SIAM J. SCI. COMPUT.
Vol. 22, No. 1, pp. 1–19
© 2000 Society for Industrial and Applied Mathematics

DISCRETIZATIONS FOR THE INCOMPRESSIBLE NAVIER–STOKES EQUATIONS BASED ON THE LATTICE BOLTZMANN METHOD*

MICHAEL JUNK† AND AXEL KLAR‡

Abstract. A discrete velocity model with spatial and velocity discretization based on a lattice Boltzmann method is considered in the low Mach number limit. A uniform numerical scheme for this model is investigated. In the limit, the scheme reduces to a finite difference scheme for the incompressible Navier–Stokes equation, which is a projection method with a second order spatial discretization on a regular grid. The discretization is analyzed and the method is compared to Chorin's original spatial discretization. Numerical results supporting the analytical statements are presented.

Key words. discrete velocity models, lattice Boltzmann method, low Mach number limit, incompressible Navier–Stokes equations, Chorin's projection scheme, finite difference method, relaxation systems

AMS subject classifications. 76P05, 76D05, 65M06, 35B25

PII. S1064827599357188

1. Introduction.

Lattice Boltzmann methods (LBMs) use discrete velocity models of kinetic equations to obtain approximate solutions of the incompressible Navier–Stokes system. The idea of the LBM rests on the observation that the kinetic and the incompressible Navier–Stokes models are equivalent in the limit of small Knudsen and Mach numbers. See [11, 3, 7] for reviews on LBMs and [28] for a review on discrete velocity models. In recent years, numerous articles on the LBM have appeared in which the method is analyzed. We refer to the references in the above cited reviews and, for example, to [1, 13, 15, 6, 33]. For connections to kinetic schemes, see [22].

A disadvantage of standard lattice Boltzmann (LB) discretizations is that the stiffness of the kinetic equation in the limit of small Knudsen and Mach numbers is not taken into account; see also [27]. Since the discretization is fully explicit, very fine time and space steps have to be used, slowing down the method considerably. To allow for larger discretization steps, the algorithm should at least be partially implicit. Such an approach has been successfully used for a large number of kinetic equations with stiff relaxation terms in fluid dynamic or diffusive limits and has led to the development of asymptotic preserving methods, see [4, 20, 18, 19, 26, 23, 24]. For an LB-type discrete velocity model with a diffusive scaling, a scheme working uniformly in the incompressible Navier–Stokes limit, using a semi-implicit time discretization and leading to a Chorin projection scheme with MAC grid, has been suggested in [25].

In this work, our starting point is the following: numerically, it has been proven by many authors that the LBM yields stable and reliable results for the incompressible Navier–Stokes equation [30, 16]. Moreover, pressure oscillations, as in the original Chorin method, are not observed although the scheme works on a regular (collocated)

*Received by the editors June 16, 1999; accepted for publication (in revised form) October 14, 1999; published electronically June 13, 2000.
http://www.siam.org/journals/sisc/22-1/35718.html
†FB Mathematik, U Kaiserslautern, 67663 Kaiserslautern, Germany (junk@mathematik.uni-kl.de).
‡FB Mathematik, TU Darmstadt, 64289 Darmstadt, Germany (klar@mathematik.tu-darmstadt.de).

grid. The aim, to extract the reason for this nice behavior, is accomplished by developing a method based on the LB spatial discretization and comparing it to Chorin's original method.

Our approach is based on the observation that the velocity-discrete kinetic equation is in one-to-one correspondence with a system of moment equations; see, for example [12, 10]. The system includes the equations of mass and momentum which yield the Navier–Stokes equations in a suitable diffusion limit (related to small Knudsen and Mach numbers). Using essentially the space-discretization of LBM, we automatically obtain a discretization of the moment system which leads to a new spatial discretization for the incompressible Navier–Stokes equations. The discretization is used together with the Chorin projection.

As has been investigated in detail, for example, in [34, 35], that the original Chorin space discretization leads to an alternating error of first order in the pressure. This type of instability is not seen in the projection method if the MAC grid is used. In this case a second order approximation of the pressure is obtained without alternating terms in the error expansion for the pressure at first order. However, one has to use staggered grids having different locations for pressure and velocity. We mention that higher order approximation or regularizing methods have been used to avoid staggered grids and alternating terms in the error expansion (see [34]). It turns out that the spatial discretization of the scheme presented here is second order for both the pressure and the velocity, not using grid staggering. However, although the error in pressure is reduced compared to Chorin's scheme the errors in velocity are larger.

The paper is organized as follows: Section 2 introduces an LB-type discrete velocity model and its associated closed moment system. In section 3 the time and space discretization of the numerical scheme for the discrete velocity model is described. Section 4 deals with the low Mach number limit of the discretized equations leading to the projection method for the incompressible Navier–Stokes equations with the new spatial discretization. Section 5 contains remarks on the treatment of the boundary conditions and an analytical investigation of the scheme following the work in [34]. In particular, the scheme is compared to Chorin's original method. Finally, section 6 presents a numerical investigation of the second order convergence for the pressure which has been found analytically. A numerical comparison with Chorin's scheme is included as well.

2. An LB-type discrete velocity model and the associated moment system. The basic kinetic model is given by the Boltzmann equation

$$\frac{\partial f}{\partial t} + \mathbf{v}\nabla f = J(f) \tag{2.1}$$

which describes the evolution of a particle density $f(\mathbf{x}, \mathbf{v}, t)$. The left-hand side of (2.1) represents free transport of the particles while the right-hand side describes interactions through collisions. The difference between continuous and discrete velocity models is the structure of the phase space $\mathcal{X} \times \mathcal{V}$. In the classical Boltzmann equation, the space part $\mathcal{X}$ is a subset of $\mathbb{R}^3$ and the velocity domain $\mathcal{V}$ is the full space $\mathbb{R}^3$. For discrete models we have

$$\mathcal{V} = \{\mathbf{c}_0, \ldots, \mathbf{c}_{N-1}\}, \qquad \mathbf{c}_i \in \mathbb{R}^d.$$

In our particular example, we consider a two-dimensional model ($d = 2$) with nine velocities ($N = 9$)

$$\mathbf{c}_1 = \left(\begin{smallmatrix}1\\0\end{smallmatrix}\right), \quad \mathbf{c}_2 = \left(\begin{smallmatrix}0\\1\end{smallmatrix}\right), \quad \mathbf{c}_3 = \left(\begin{smallmatrix}-1\\0\end{smallmatrix}\right), \quad \mathbf{c}_4 = \left(\begin{smallmatrix}0\\-1\end{smallmatrix}\right),$$
$$\mathbf{c}_5 = \left(\begin{smallmatrix}1\\1\end{smallmatrix}\right), \quad \mathbf{c}_6 = \left(\begin{smallmatrix}-1\\1\end{smallmatrix}\right), \quad \mathbf{c}_7 = \left(\begin{smallmatrix}-1\\-1\end{smallmatrix}\right), \quad \mathbf{c}_8 = \left(\begin{smallmatrix}1\\-1\end{smallmatrix}\right),$$

and $\mathbf{c}_0 = \mathbf{0}$. In the discrete case, the $\mathbf{v}$-dependence of the particle distribution $f(\mathbf{x}, \mathbf{v}, t)$ is uniquely determined through N functions

$$f_i(\mathbf{x}, t) = f(\mathbf{x}, \mathbf{c}_i, t), \qquad i = 0, \ldots, N-1.$$

Macroscopic quantities like mass–, momentum–, or energy density are obtained by taking velocity moments of f. If ψ is any $\mathbf{v}$-dependent function, we denote the discrete velocity integral by

$$\langle \psi \rangle = \sum_{i=0}^{N-1} \psi(\mathbf{c}_i).$$

Then, mass and momentum density can be written as

$$\begin{aligned} \rho(\mathbf{x}, t) &= \langle f(\mathbf{x}, \mathbf{v}, t) \rangle = \sum_{i=0}^{N-1} f_i(\mathbf{x}, t), \\ \rho \mathbf{u}(\mathbf{x}, t) &= \langle \mathbf{v} f(\mathbf{x}, \mathbf{v}, t) \rangle = \sum_{i=0}^{N-1} \mathbf{c}_i f_i(\mathbf{x}, t); \end{aligned} \tag{2.2}$$

compare, for example, [31]. In LB applications, the collision operator $J(f)$ in (2.1) is typically of BGK-type

$$J(f) = -\frac{1}{\tau}(f - f^{eq}). \tag{2.3}$$

The parameter $\tau > 0$ is called *relaxation time* and f^{eq} is the *equilibrium distribution*. In the isothermal case, f^{eq} depends on f through the parameters ρ and $\mathbf{u}$ which are calculated according to (2.2); see, for example, [13]. For the standard D2Q9-model [29] with nine velocities, we have

$$f^{eq}(\rho, \mathbf{u}; \mathbf{v}) = \rho \left(1 + 3\mathbf{u} \cdot \mathbf{v} - \frac{3}{2}|\mathbf{u}|^2 + \frac{9}{2}(\mathbf{u} \cdot \mathbf{v})^2\right) f^*(\mathbf{v}),$$

where f^* is defined by

$$f^*(\mathbf{c}_i) = \begin{cases} \frac{4}{9} & i = 0, \\ \frac{1}{9} & i = 1, \ldots, 4, \\ \frac{1}{36} & i = 5, \ldots, 8. \end{cases}$$

The equilibrium distribution is constructed in such a way that

$$\langle J(f) \rangle = 0 \qquad \text{and} \qquad \langle \mathbf{v} J(f) \rangle = \mathbf{0},$$

which reflects conservation of mass and momentum in the collision process.

In order to obtain a relation between the kinetic equation (2.1) and the incompressible Navier–Stokes system, we introduce the diffusive scaling $\mathbf{x} \to \mathbf{x}/\epsilon$, $t \to t/\epsilon^2$ together with a rescaling of velocity $\mathbf{u} \to \epsilon\mathbf{u}$. This scaling describes the small Knudsen and low Mach number limit of kinetic equations; see [32, 9, 2, 17] for details. Under these transformations, (2.1) turns into

$$\frac{\partial f}{\partial t} + \frac{1}{\epsilon}\mathcal{D}f = -\frac{1}{\epsilon^2\tau}(f - f^{eq}(\rho, \epsilon\mathbf{u})), \tag{2.4}$$

where $\mathcal{D} = \mathbf{v}.\nabla$ has been used as abbreviation for the space derivatives. In our exemplary case, (2.4) consists of nine equations for the occupation numbers $f_0, \ldots, f_8$. In order to get closer in notation to the Navier–Stokes system, we transform (2.4) into an equivalent set of moment equations (see also [25, 10] for a similar approach). Based on the $\mathbf{v}$-polynomials (compare [12])

(2.5)

$$\begin{aligned}
&P_0(\mathbf{v}) = 1,\\
&P_1(\mathbf{v}) = \frac{v_1}{\epsilon}, && P_2(\mathbf{v}) = \frac{v_2}{\epsilon},\\
&P_3(\mathbf{v}) = \frac{v_1^2}{\epsilon^2} - \frac{1}{3\epsilon^2}, && P_4(\mathbf{v}) = \frac{v_1 v_2}{\epsilon^2}, && P_5(\mathbf{v}) = \frac{v_2^2}{\epsilon^2} - \frac{1}{3\epsilon^2},\\
&P_6(\mathbf{v}) = \frac{(3|\mathbf{v}|^2 - 4)v_1}{\epsilon^3}, && P_7(\mathbf{v}) = \frac{(3|\mathbf{v}|^2 - 4)v_2}{\epsilon^3},\\
&P_8(\mathbf{v}) = \frac{9|\mathbf{v}|^4 - 15|\mathbf{v}|^2 + 2}{\epsilon^4},
\end{aligned}$$

we obtain an invertible linear mapping $f \to \mathbf{P}f$ defined by

$$\mathbf{P}f = (\langle P_0 f\rangle, \ldots, \langle P_8 f\rangle)^T.$$

Applying $\mathbf{P}$ to (2.4) results in an equivalent set of equations with a differential operator which is still linear and hyperbolic

$$\left(\frac{\partial}{\partial t} + \frac{1}{\epsilon}\mathbf{P}\mathcal{D}\mathbf{P}^{-1}\right)\mathbf{P}f = -\frac{1}{\epsilon^2\tau}\left(\mathbf{P}f - \mathbf{P}f^{eq}(\rho, \epsilon\mathbf{u})\right). \tag{2.6}$$

In order to write (2.6) in a more explicit form, we introduce names for the moments. Note that $\langle P_0 f\rangle = \rho$ and $\langle P_i f\rangle = \rho u_i$ for $i = 1, 2$. The second order moments form a symmetric tensor

$$\Theta = \begin{pmatrix}\Theta_{11} & \Theta_{12}\\ \Theta_{12} & \Theta_{22}\end{pmatrix} = \begin{pmatrix}\langle P_3 f\rangle & \langle P_4 f\rangle\\ \langle P_4 f\rangle & \langle P_5 f\rangle\end{pmatrix}$$

and for the remaining moments we set

$$\mathbf{q} = \begin{pmatrix}\langle P_6 f\rangle\\ \langle P_7 f\rangle\end{pmatrix}, \qquad s = \langle P_8 f\rangle.$$

The first two equations in (2.6) are those of mass and momentum conservation

$$\begin{aligned}
\partial_t \rho + \operatorname{div} \rho\mathbf{u} &= 0,\\
\partial_t \rho\mathbf{u} + \operatorname{div} \Theta + \frac{1}{3\epsilon^2}\nabla\rho &= 0.
\end{aligned} \tag{2.7}$$

Here, the divergence is applied to the rows of Θ. The equation for Θ is

$$\partial_t \Theta + \frac{2}{3\epsilon^2} S[\rho \mathbf{u}] + \frac{1}{3} Q = -\frac{1}{\epsilon^2 \tau}(\Theta - \rho \mathbf{u} \otimes \mathbf{u}), \tag{2.8}$$

where

$$S_{ij}[\rho \mathbf{u}] = \frac{1}{2}\left(\frac{\partial \rho u_i}{\partial x_j} + \frac{\partial \rho u_j}{\partial x_i}\right) \qquad \text{and} \qquad Q = \begin{pmatrix} \partial_{x_2} q_2 & \partial_{x_2} q_1 + \partial_{x_1} q_2 \\ \partial_{x_2} q_1 + \partial_{x_1} q_2 & \partial_{x_1} q_1 \end{pmatrix}.$$

Finally, the third and fourth order moments satisfy

$$\begin{aligned} \partial_t \mathbf{q} + \frac{1}{\epsilon^2} \operatorname{div} \begin{pmatrix} \Theta_{22} & 2\Theta_{12} \\ 2\Theta_{12} & \Theta_{11} \end{pmatrix} + \frac{1}{6} \nabla s &= -\frac{1}{\epsilon^2 \tau} \mathbf{q}, \\ \partial_t s + \frac{4}{\epsilon^2} \operatorname{div} \mathbf{q} &= -\frac{1}{\epsilon^2 \tau} s. \end{aligned} \tag{2.9}$$

Altogether, we obtain a hyperbolic system with stiff relaxation terms. The diffusion limit of the above system is straightforwardly determined. From the momentum equation in (2.7) we conclude that $\nabla \rho$ tends to zero as $\epsilon \to 0$. Hence, ρ approaches a constant $\bar{\rho}$ (which is the Boussinesq relation in the isothermal case). Writing $\rho = \bar{\rho}(1 + 3\epsilon^2 p)$, (2.7) transforms into

$$\begin{aligned} \partial_t p + \frac{1}{3\epsilon^2} \operatorname{div} \mathbf{u} &= -\operatorname{div} p\mathbf{u}, \\ \partial_t \mathbf{u} + \operatorname{div} \frac{1}{\bar{\rho}} \Theta + \nabla p &= -3\epsilon^2 \partial_t p \mathbf{u}. \end{aligned} \tag{2.10}$$

For $\epsilon \to 0$, (2.8) yields in lowest order

$$\frac{1}{\bar{\rho}} \Theta = \mathbf{u} \otimes \mathbf{u} - \frac{2\tau}{3} S[\mathbf{u}]. \tag{2.11}$$

Since (2.9) decouples completely from the other equations (in lowest order) and since $2 \operatorname{div} S[\mathbf{u}] = (\Delta + \nabla \operatorname{div})\mathbf{u}$, we obtain from (2.10) and (2.11) the incompressible Navier–Stokes equations as limiting system

$$\begin{aligned} \operatorname{div} \mathbf{u} &= 0, \\ \partial_t \mathbf{u} + \operatorname{div} \mathbf{u} \otimes \mathbf{u} + \nabla p &= \frac{\tau}{3} \Delta \mathbf{u}. \end{aligned} \tag{2.12}$$

Compare [14] for another approach. The Reynolds number is related to the relaxation time by $Re = 3/\tau$.

We remark that (2.7), (2.8), and (2.9) can be viewed as a relaxation system for (2.12). Since the LBM is a particular discretization of (2.4) and since a discretization of (2.4) automatically turns into a discretization of the moment system under the transformation $\mathbf{P}$, we conclude that the LBM can be viewed as a relaxation method for the incompressible Navier–Stokes system [21].

3. The numerical scheme: Spatial and temporal discretization. To obtain a spatial discretization of the moment system (2.7), (2.8), and (2.9), we use a first order upwind method for the operator $\mathcal{D} = \mathbf{v} \cdot \nabla$ in (2.6). This choice is motivated by the LBM where the upwind approximation is combined with an explicit

Euler discretization for the time derivative. To get a proper treatment of the stiff pressure–velocity coupling, however, we choose a semi-implicit time discretization instead of the explicit one (see also [25]). In the limit $\epsilon \to 0$, a projection scheme for the incompressible Navier–Stokes equations is obtained.

Before we describe the discretization of the moment system, let us first show how the LBM is obtained in this context; see also [5, 13, 15]. The discretization of $\mathcal{D} = \mathbf{v} \cdot \nabla$ is taken as

$$(\mathcal{D}_h f)(\mathbf{x}, \mathbf{v}, t) = \frac{1}{h}(f(\mathbf{x}, \mathbf{v}, t) - f(\mathbf{x} - h\mathbf{v}, \mathbf{v}, t)). \tag{3.1}$$

Together with an explicit Euler discretization of the time derivative and an evaluation of the collision operator at a shifted $\mathbf{x}$-position, (2.4) turns into

$$f(\mathbf{x}, \mathbf{v}, t + \Delta t) - f(\mathbf{x}, \mathbf{v}, t) + \frac{\Delta t}{\epsilon h}(f(\mathbf{x}, \mathbf{v}, t) - f(\mathbf{x} - h\mathbf{v}, \mathbf{v}, t)) \\ = -\frac{\Delta t}{\epsilon^2 \tau}(f - f^{eq}(\rho, \epsilon\mathbf{u}))(\mathbf{x} - h\mathbf{v}, \mathbf{v}, t).$$

Setting $\Delta t = \epsilon^2$, $h = \epsilon$, $\mathbf{v} = \epsilon\mathbf{w}$, and transforming the space variable $\mathbf{x} \to \mathbf{x} + \mathbf{w}\Delta t$ yields for the occupation numbers

$$f_i(\mathbf{x} + \mathbf{w}_i\Delta t, t + \Delta t) - f_i(\mathbf{x}, t) = -\frac{1}{\tau}\Big(f_i(\mathbf{x}, t) - f_i^{eq}(\rho(\mathbf{x}, t), \epsilon\mathbf{u}(\mathbf{x}, t))\Big),$$

which is the standard discretization in the LBM (see [29, 6, 1]).

To describe the effect of the discretization (3.1) on the moment system, we introduce the vectors

$$\mathbf{M} = \mathbf{P}f = (\rho, \rho u_1, \rho u_2, \Theta_{11}, \Theta_{12}, \Theta_{22}, q_1, q_2, s)^T$$

and

$$\mathbf{M}^{eq} = \mathbf{P}f^{eq}(\rho, \epsilon\mathbf{u}) = (\rho, \rho u_1, \rho u_2, \rho u_1^2, \rho u_1 u_2, \rho u_2^2, 0, 0, 0)^T.$$

Using (3.1) to replace $\mathcal{D}$ in (2.6), the discretized moment system can be written in the compact form

$$\frac{\partial \mathbf{M}}{\partial t} + \frac{1}{\epsilon}\mathbf{P}\mathcal{D}_h\mathbf{P}^{-1}\mathbf{M} = -\frac{1}{\tau\epsilon^2}(\mathbf{M} - \mathbf{M}^{eq}).$$

If $\{\mathbf{e}_i : i = 0, \ldots, 8\}$ are the standard unit vectors in $\mathbb{R}^9$, we can write the contribution of M_j to the equation for M_i as

$$\frac{1}{\epsilon}\mathbf{e}_i \cdot \Big(\mathbf{P}\mathcal{D}_h\mathbf{P}^{-1}\Big)M_j\mathbf{e}_j = \frac{1}{\epsilon}\mathbf{e}_i \cdot \Big(\mathbf{P}\frac{1}{h}(M_j(\mathbf{x}, t) - M_j(\mathbf{x} - h\mathbf{v}))\mathbf{P}^{-1}\mathbf{e}_j\Big). \tag{3.2}$$

With the abbreviation $g_j = \mathbf{P}^{-1}\mathbf{e}_j$, (3.2) can be transformed into

$$\frac{1}{\epsilon}\mathbf{e}_i \cdot \Big(\mathbf{P}\mathcal{D}_h\mathbf{P}^{-1}\Big)M_j\mathbf{e}_j = \frac{1}{\epsilon h}\Big(M_j(\mathbf{x}, t)\delta_{ij} - \langle P_i(\mathbf{v})g_j(\mathbf{v})M_j(\mathbf{x} - \mathbf{v}h)\rangle\Big) \\ = \frac{1}{\epsilon h}\Big(M_j(\mathbf{x}, t)\delta_{ij} - \sum_{k=0}^{8} P_i(\mathbf{c}_k)g_j(\mathbf{c}_k)M_j(\mathbf{x} - \mathbf{c}_k h)\Big).$$

Hence, the contribution of M_j to equation i can be written as a finite difference stencil with the general form

$$\frac{1}{\epsilon h}\begin{bmatrix} -P_i g_j(\mathbf{c}_8) & -P_i g_j(\mathbf{c}_4) & -P_i g_j(\mathbf{c}_7) \\ -P_i g_j(\mathbf{c}_1) & \delta_{ij} - P_i g_j(\mathbf{c}_0) & -P_i g_j(\mathbf{c}_3) \\ -P_i g_j(\mathbf{c}_5) & -P_i g_j(\mathbf{c}_2) & -P_i g_j(\mathbf{c}_6) \end{bmatrix}. \tag{3.3}$$

In particular, for each space derivative in the moment system (2.7), (2.8), and (2.9), a corresponding stencil is found. The discretization of x_1- and x_2-derivatives occurs in three different ways. For $\operatorname{div} \rho\mathbf{u}$ in the mass conservation equation as well as for $\nabla\rho$ in the momentum equation, the following stencils appear:

$$\frac{\partial}{\partial x_1} \approx \frac{1}{12h}\begin{bmatrix} -1 & 0 & 1 \\ -4 & 0 & 4 \\ -1 & 0 & 1 \end{bmatrix}, \qquad \frac{\partial}{\partial x_2} \approx \frac{1}{12h}\begin{bmatrix} 1 & 4 & 1 \\ 0 & 0 & 0 \\ -1 & -4 & -1 \end{bmatrix}. \tag{3.4}$$

They also occur in the momentum equation to approximate $\partial_{x_1}\Theta_{11}, \partial_{x_2}\Theta_{22}$ and in (2.8) for $\partial_{x_1}\rho u_1, \partial_{x_2}\rho u_2$. The derivatives of Θ_{12} as well as $\rho u_1 u_2$ are always discretized by

$$\frac{\partial}{\partial x_1} \approx \frac{1}{4h}\begin{bmatrix} -1 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}, \qquad \frac{\partial}{\partial x_2} \approx \frac{1}{4h}\begin{bmatrix} 1 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & -1 \end{bmatrix}. \tag{3.5}$$

These stencils also appear in (2.8) for $\partial_{x_1}\rho u_2, \partial_{x_2}\rho u_2$ and $\partial_{x_1} q_2, \partial_{x_2} q_1$. The remaining derivatives in the moment system turn out to be approximated by

$$\frac{\partial}{\partial x_1} \approx \frac{1}{6h}\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}, \qquad \frac{\partial}{\partial x_2} \approx \frac{1}{6h}\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}. \tag{3.6}$$

We remark that each stencil is second order accurate. In fact, the stencils can be viewed as convex combinations of standard central difference approximations. For example, the ∂_{x_1}-stencil in (3.4) can be written as

$$\frac{1}{12h}\begin{bmatrix} -1 & 0 & 1 \\ -4 & 0 & 4 \\ -1 & 0 & 1 \end{bmatrix} = \frac{1}{6}\left(\frac{1}{2h}\begin{bmatrix} -1 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} + \frac{4}{2h}\begin{bmatrix} 0 & 0 & 0 \\ -1 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} + \frac{1}{2h}\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix} \right).$$

Up to now, we have investigated the approximations of the space derivatives appearing in the moment system. However, if we consider, for example, the discretized mass conservation equation, we find that apart from $M_1 = \rho u_1$ and $M_2 = \rho u_2$, the other variables $M_0, M_3, \ldots, M_8$ contribute also. Of course, this contribution is of higher order in the discretization parameter h and the appearance of such terms is not surprising since we started with a first order upwind approximation (3.1). As an example, we mention the contribution of ρ to the mass conservation equation (i.e., $i = j = 0$ in 3.3). Evaluating (3.3), we find up to a constant factor

$$\frac{1}{6h}\begin{bmatrix} -1 & -4 & -1 \\ -4 & 20 & -4 \\ -1 & -4 & -1 \end{bmatrix} = -h\Delta + \mathcal{O}(h^3),$$

which is the standard nine-point stencil for the Laplacian. If we keep all terms which appear in that way and which are of first order in h, we obtain the *modified equation* of the finite difference approximation (i.e., the equation which is approximated to second order accuracy). The modified equation corresponding to (2.7) is

$$\begin{aligned} \partial_t \rho + \operatorname{div} \rho \mathbf{u} &= \left(\frac{1}{6\epsilon} \Delta \rho + \frac{\epsilon}{2} \operatorname{div} \operatorname{div} \Theta \right) h, \\ \partial_t \rho \mathbf{u} + \operatorname{div} \Theta + \frac{1}{3\epsilon^2} \nabla \rho &= \left(\frac{1}{6\epsilon} (\Delta + 2\nabla \operatorname{div}) \rho \mathbf{u} + \frac{\epsilon}{6} \operatorname{div} Q \right) h. \end{aligned} \tag{3.7}$$

Similarly, equations are obtained for $\Theta, \mathbf{q}$ and s. Note that the additional terms have a stabilizing effect because they act as artificial viscosity. In the original LB approach, this viscosity is partly compensated by a negative viscosity due to the explicit Euler discretization of the time derivative and partly it is combined with the physical viscosity in the Navier–Stokes equation (which is possible since $(\Delta + 2\nabla \operatorname{div})\rho \mathbf{u}$ has the correct structure). In this way, second order accuracy of the LBM is obtained. Since we want to avoid the explicit time discretization but still obtain a second order accurate scheme, we neglect all the stencils described by (3.3) which do not have counterparts in the moment system. With this step, we focus only on the interplay between the stencils (3.4), (3.5), and (3.6). Another possibility is to start with a second order upwind discretization of $\mathcal{D}$ instead of the first order approach (3.1) (compare here, for example, [20]).

To introduce the time discretization, we will not replace the space derivatives by their finite difference approximations (3.4), (3.5), and (3.6) but restrict to the spatially continuous case to avoid confusion. Of course, the complete scheme is obtained by combining both space and time discretizations.

Introducing the momentum vector $\mathbf{m}^k(\mathbf{x}) = \rho \mathbf{u}(\mathbf{x}, k\Delta t)$ and the pressure variable p^k by $\rho(\mathbf{x}, k\Delta t) = \bar{\rho}(1 + 3\epsilon^2 p^k(\mathbf{x}))$, we define p^{k+1} and $\mathbf{m}^{k+1}$ based on a semi-implicit discretization of (2.7)

$$\begin{aligned} p^{k+1} &= p^k - \frac{\Delta t}{3\bar{\rho}\epsilon^2} \operatorname{div} \mathbf{m}^{k+1}, \\ \mathbf{m}^{k+1} &= \mathbf{m}^k - \Delta t (\operatorname{div} \Theta^k + \bar{\rho} \nabla p^{k+1}). \end{aligned} \tag{3.8}$$

Inserting $\mathbf{m}^{k+1}$ into the pressure equation yields a Helmholtz problem

$$\left(\Delta - \frac{3\epsilon^2}{\Delta t^2} \right) p^{k+1} = \frac{1}{\bar{\rho} \Delta t} \operatorname{div} \mathbf{m}^k - \operatorname{div} \operatorname{div} \frac{1}{\bar{\rho}} \Theta^k - \frac{3\epsilon^2}{\Delta t^2} p^k. \tag{3.9}$$

We remark that the discretization of the second order operators Δ and div div is automatically given by a composition of the discretizations (3.4), (3.5), and (3.6). As time discretization of (2.8), we choose

$$\Theta^{k+1} = \Theta^k - \frac{2\Delta t}{3\epsilon^2} S[\mathbf{m}^{k+1}] + \frac{\Delta t}{3} Q^k - \frac{\Delta t}{\epsilon^2 \tau} \left(\Theta^{k+1} - \left(\frac{\mathbf{m} \otimes \mathbf{m}}{\rho} \right)^{k+1} \right). \tag{3.10}$$

Finally, (2.9) is treated according to

$$\begin{aligned} \mathbf{q}^{k+1} &= \mathbf{q}^k - \frac{\Delta t}{\epsilon^2} \operatorname{div} \begin{pmatrix} \Theta_{22} & 2\Theta_{12} \\ 2\Theta_{12} & \Theta_{11} \end{pmatrix}^k - \frac{\Delta t}{6} \nabla s^{k+1} - \frac{\Delta t}{\epsilon^2 \tau} \mathbf{q}^{k+1}, \\ s^{k+1} &= s^k - \frac{4\Delta t}{\epsilon^2} \operatorname{div} \mathbf{q}^{k+1} - \frac{\Delta t}{\epsilon^2 \tau} s^{k+1}, \end{aligned}$$

which again leads to an elliptic problem (with $\alpha = \tau/\Delta t$)

$$\left(1 - \frac{4\tau^2\epsilon^2}{6(1+\epsilon^2\alpha)^2}\Delta\right) s^{k+1} = \frac{\epsilon^2\alpha}{1+\epsilon^2\alpha} s^k - \frac{4\tau\epsilon^2\alpha}{(1+\epsilon^2\alpha)^2} \operatorname{div} \mathbf{q}^k + \frac{4\tau^2}{(1+\epsilon^2\alpha)^2} \operatorname{div} \operatorname{div} \begin{pmatrix} \Theta_{22} & 2\Theta_{12} \\ 2\Theta_{12} & \Theta_{11} \end{pmatrix}^k.$$

The elliptic problems for p and s, which result from the semi-implicit treatment, can be solved by an iterative procedure. The remaining equations for $\mathbf{m}, \Theta$, and $\mathbf{q}$ are explicit.

4. The discretization of the incompressible Navier–Stokes equations. The discretization of the moment system described in the last section tends, as $\epsilon \to 0$, to a Chorin-type projection method for the incompressible Navier–Stokes equations. Since $\rho \to \bar{\rho}$ in that limit, we have $\mathbf{m} = \rho\mathbf{u} \to \bar{\rho}\mathbf{u}$ and the momentum equation in (3.8) yields

$$\mathbf{u}^{k+1} = \mathbf{u}^k - \Delta t \operatorname{div} \frac{1}{\bar{\rho}}\Theta^k - \Delta t \nabla p^{k+1}. \tag{4.1}$$

The Helmholtz equation (3.9) for the pressure turns into

$$\Delta p^{k+1} = \frac{1}{\Delta t} \operatorname{div} \mathbf{u}^k - \operatorname{div} \operatorname{div} \frac{1}{\bar{\rho}}\Theta^k. \tag{4.2}$$

Both (4.1) and (4.2) are combined with the limit of (3.10)

$$\frac{1}{\bar{\rho}}\Theta^{k+1} = \mathbf{u}^{k+1} \otimes \mathbf{u}^{k+1} - \frac{2\tau}{3} S[\mathbf{u}^{k+1}]. \tag{4.3}$$

For (4.1), this implies

$$\mathbf{u}^{k+1} = \mathbf{u}^k - \Delta t \operatorname{div} \mathbf{u}^k \otimes \mathbf{u}^k - \Delta t \nabla p^{k+1} + \frac{2\tau\Delta t}{3} \operatorname{div} S[\mathbf{u}^k]. \tag{4.4}$$

By construction, each differential operator in (4.4) is composed of derivatives from the original moment system for which we have derived the discretizations (3.4), (3.5), and (3.6). If $\bar{D}_1, \bar{D}_2$ denote the approximations in (3.4) and $\tilde{D}_1, \tilde{D}_2$ denote those in (3.5), we find for the pressure gradient and the convective term in (4.4)

$$\nabla p \leftrightarrow \bar{G}p = \begin{pmatrix} \bar{D}_1 p \\ \bar{D}_2 p \end{pmatrix}, \qquad \operatorname{div} \mathbf{u} \otimes \mathbf{u} \leftrightarrow \hat{D}\mathbf{u}^k \otimes \mathbf{u}^k = \begin{pmatrix} \bar{D}_1 u_1^2 + \tilde{D}_2 u_1 u_2 \\ \tilde{D}_1 u_1 u_2 + \bar{D}_2 u_2^2 \end{pmatrix}. \tag{4.5}$$

The viscous term is of the form

$$2 \operatorname{div} S[\mathbf{u}] \leftrightarrow \hat{L}\mathbf{u} = \begin{pmatrix} 2\bar{D}_1^2 u_1 + \tilde{D}_2^2 u_1 + \tilde{D}_2 \tilde{D}_1 u_2 \\ \tilde{D}_1 \tilde{D}_2 u_1 + \tilde{D}_1^2 u_2 + 2\bar{D}_2^2 u_2 \end{pmatrix}. \tag{4.6}$$

Under discrete divergence-free condition, $\bar{D}\mathbf{u} = 0$, a reorganization yields

$$\begin{pmatrix} 2\bar{D}_1^2 u_1 + \tilde{D}_2^2 u_1 + \tilde{D}_2 \tilde{D}_1 u_2 \\ \tilde{D}_1 \tilde{D}_2 u_1 + \tilde{D}_1^2 u_2 + 2\bar{D}_2^2 u_2 \end{pmatrix} = \begin{pmatrix} \bar{D}_1^2 u_1 + \tilde{D}_2^2 u_1 \\ \tilde{D}_1^2 u_2 + \bar{D}_2^2 u_2 \end{pmatrix} + \left[\tilde{D}_1 \tilde{D}_2, \bar{D}_1 \bar{D}_2\right] \mathbf{u},$$

where $[\cdot,\cdot]$ denotes the commutator. Since both $\tilde{D}_1\tilde{D}_2$ and $\bar{D}_1\bar{D}_2$ are second order accurate approximations of the mixed derivative $\partial_{x_1}\partial_{x_2}$, the commutator vanishes in the order of accuracy of the method

$$\left[\tilde{D}_1\tilde{D}_2, \bar{D}_1\bar{D}_2\right]\mathbf{u} = \mathcal{O}(h^2).$$

Thus, $\hat{L}$ is a second order approximation of the Laplacian. Finally, $\operatorname{div}\mathbf{u}^k$ and the Laplacian in (4.2) turn out to be discretized by

$$\operatorname{div}\mathbf{u} \leftrightarrow \bar{D}\mathbf{u} = \bar{D}_1 u_1 + \bar{D}_2 u_2,$$
$$\Delta p \leftrightarrow \bar{L}p = \bar{D}\bar{G}p = \left(\bar{D}_1^2 + \bar{D}_2^2\right)p.$$

According to (4.3), the double divergence in (4.2) splits into two contributions. The convective part $\operatorname{div}\operatorname{div}\mathbf{u}\otimes\mathbf{u}$ is discretized by $\bar{D}\hat{D}\mathbf{u}\otimes\mathbf{u}$ where $\hat{D}\mathbf{u}\otimes\mathbf{u}$ is given in (4.5). The part involving third order derivatives is treated similarly, discretizing $\operatorname{div}\operatorname{div} S[\mathbf{u}]$ by $\bar{D}\hat{D}S[\mathbf{u}]$. Altogether, the discrete versions of (4.2) and (4.4) can be written as

$$\bar{L}p^{k+1} = \frac{1}{\Delta t}\bar{D}\mathbf{u}^k - \bar{D}\hat{D}\mathbf{u}^k \otimes \mathbf{u}^k + \frac{\tau}{3}\bar{D}\hat{L}\mathbf{u}^k$$

and

$$\mathbf{u}^{k+1} = \mathbf{u}^k - \Delta t\hat{D}\mathbf{u}^k \otimes \mathbf{u}^k - \Delta t\bar{G}p^{k+1} + \frac{\tau\Delta t}{3}\hat{L}\mathbf{u}^k,$$

which is a Chorin-type projection method with spatial discretization induced by the LB approach.

5. Boundary conditions and remarks on alternating errors. In this section appropriate boundary conditions for our method are introduced and oscillating pressure error terms are discussed. The following investigations are motivated by the detailed analysis of finite difference schemes for the incompressible Navier–Stokes equations in [34].

In Chorin's method [8], the pressure is only first order accurate (with respect to the spatial discretization parameter h) despite the fact that all finite difference approximations are second order consistent. As shown in [34], this behavior results from an interaction of the discretized viscous term in the Navier–Stokes equations and the pressure Poisson equation. More precisely, the wide stencil for the Laplacian in the pressure equation essentially decouples the problem into two Poisson problems on separate subgrids. Since the subproblems are subject to boundary conditions which differ at third order, alternating errors at third order are introduced (i.e., error terms are present on one subgrid and absent on the other). If the Laplacian in the viscous term is discretized with the standard five-point stencil, as in Chorin's method, the application to an alternating function yields contributions of order $1/h^2$. In this way, the third order oscillating error is brought down to first order. To avoid this phenomenon, several approaches have been proposed and some of them are analyzed in [34]. In the LB-induced discretization of the Navier–Stokes equations described in section 4, the accuracy of the pressure remains second order. In contrast to the methods investigated in [34], an amplification of the alternating error at third order is avoided due to the wide stencil $\hat{L}$ for the Laplacian in the viscous term.

To explain the basic principles which lead to the alternating error terms in Chorin's method, we consider the Stokes equation in a periodic channel as a model

problem. The extension of the boundary treatment to the new discretization will then be straight forward.

By neglecting the nonlinear terms in the Navier–Stokes equations, we obtain the Stokes equation

$$\begin{aligned} \partial_t \mathbf{u} + \nabla p &= \nu \Delta \mathbf{u}, \\ \operatorname{div} \mathbf{u} = \mathbf{0} \quad \text{in } \Omega, \qquad \mathbf{u}|_{t=0} &= \mathbf{u}_0. \end{aligned} \tag{5.1}$$

As in [34], we choose $\Omega = (0,1) \times (0,1)$ with no-slip conditions $\mathbf{u} = \mathbf{0}$ on the top and bottom boundaries and periodic conditions in horizontal direction. The pressure p is determined by the constraint $\operatorname{div} \mathbf{u} = 0$; i.e., p has to be chosen in such a way that the gradient part of the vector field

$$\mathbf{a} = \nu \Delta \mathbf{u}$$

is exactly compensated. Following the notation in [34], the divergence-free projection of $\mathbf{a}$ is given by

$$\mathcal{P}\mathbf{a} = \mathbf{a} - \nabla q,$$

where q solves the Neumann–Poisson problem

$$\Delta q = \operatorname{div} \mathbf{a} \quad \text{in } \Omega, \qquad \frac{\partial q}{\partial \mathbf{n}} = \mathbf{n} \cdot \mathbf{a} \quad \text{on } \partial\Omega$$

with a normalization condition $\int_\Omega q \, dx = 0$. To get a discrete representation of the projection operator, we introduce discretizations G and D of the gradient and the divergence. Then, the discrete divergence-free part $\mathcal{P}_h \mathbf{a}$ of $\mathbf{a}$ is obtained by solving

$$DGq = D\mathbf{a} \tag{5.2}$$

with suitably discretized Neumann and normalization conditions. Indeed, setting

$$\mathcal{P}_h \mathbf{a} = \mathbf{a} - Gq$$

we find, by construction, $D\mathcal{P}_h \mathbf{a} = 0$.

In Chorin's method, G and D are based on standard central difference approximations on the regular square grid $\Omega_h = \{x_{ij} = (ih, jh) : i, j = 0, \dots, N\}$, where $h = 1/N$. With

$$D_1 = \frac{1}{2h} \begin{bmatrix} -1 & 0 & 1 \end{bmatrix} \qquad \text{and} \qquad D_2 = \frac{1}{2h} \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix},$$

the discrete gradient acting on a scalar function q is given by

$$(Gq)_{ij} = \begin{pmatrix} (D_1 q)_{ij} \\ (D_2 q)_{ij} \end{pmatrix} = \frac{1}{2h} \begin{pmatrix} q_{i+1,j} - q_{i-1,j} \\ q_{i,j+1} - q_{i,j-1} \end{pmatrix}.$$

(The weight in the center of the stencil always refers to the point where the stencil is applied.) At points x_{ij} well inside the domain ($2 \le j \le N-2$), the divergence D acts on a vector field $\mathbf{u}$ according to

$$D\mathbf{u} = D_1 u_1 + D_2 u_2.$$

The iterated operator $L = DG$ appearing in (5.2) is then the wide Laplacian

$$L = \frac{1}{4h^2}\begin{bmatrix} & & 1 & & \\ & & 0 & & \\ 1 & 0 & -4 & 0 & 1 \\ & & 0 & & \\ & & 1 & & \end{bmatrix}. \tag{5.3}$$

At points x_{ij} next to the boundary ($j = 1$ or $j = N-1$), the original operator D_2 is, according to Chorin's approach, replaced by

$$D_2^{(1)} = \frac{1}{2h}\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \qquad \text{and} \qquad D_2^{(N-1)} = \frac{1}{2h}\begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix}.$$

For the velocity field $\mathbf{u}$, the discrete operator

$$(D^{(j)}\mathbf{u})_{ij} = (D_1 u_1)_{ij} + (D_2^{(j)} u_2)_{ij}, \qquad j \in \{1, N-1\},$$

still approximates the divergence since $\mathbf{u} = \mathbf{0}$ on the boundary so that $(D^{(j)}\mathbf{u})_{ij} = (D\mathbf{u})_{ij}$. However,

$$L^{(1)} = D^{(1)}G = D_1 D_1 + D_2^{(1)} D_2 = \frac{1}{4h^2}\begin{bmatrix} & & 1 & & \\ & & 0 & & \\ 1 & 0 & -3 & 0 & 1 \\ & & 0 & & \\ & & 0 & & \end{bmatrix}$$

now has the interpretation as a discrete Laplacian with an incorporated boundary condition. Indeed, $(L^{(1)}q)_{i1} = (D^{(1)}\mathbf{a})_{i1}$ can be rewritten in terms of the undisturbed operators $(Lq)_{i1} = (D\mathbf{a})_{i1}$ together with the condition

$$(D_2 q)_{i0} = (a_2)_{i0}$$

which is a second order accurate approximation of the Neumann boundary condition $\frac{\partial q}{\partial \mathbf{n}} = \mathbf{a} \cdot \mathbf{n}$ at the lower boundary. (Similar considerations apply to the upper boundary.)

Finally, in boundary points, the x_1-derivative of the divergence is deleted completely ($D_1^{(j)} = 0$ for $j \in \{0, N\}$) and the x_2-derivative is obtained from the half-sided difference

$$D_{2,hs} = \frac{1}{2h}\begin{bmatrix} -1 \\ 4 \\ -3 \\ 0 \\ 0 \end{bmatrix} \tag{5.4}$$

by deleting the central weight. The equation $(L^{(0)}q)_{i0} = (D^{(0)}\mathbf{a})_{i0}$ based on the iterated operator

$$L^{(0)} = D^{(0)}G = D_1^{(0)} D_1 + D_2^{(0)} D_2$$

can again be interpreted as $(Lq)_{i0} = (D\mathbf{a})_{i0}$ but now with the boundary condition

$$(BD_2q - Ba_2)_{i,0} = \frac{h}{3}(D_1D_1q - D_1a_1)_{i,0}.$$

Here, B is the linear interpolation operator

$$B = \frac{1}{3}\begin{bmatrix} -1 \\ 3 \\ 0 \\ 1 \\ 0 \end{bmatrix} = 1 + \mathcal{O}(h^3) \tag{5.5}$$

so that $B(D_2q - a_2)$ also approximates the Neumann condition.

Before we apply the construction of the boundary stencils to the discretization induced by the LB, let us briefly comment on the mechanism which introduces alternating errors (see also [34]). An important observation is that the wide Laplacian (5.3) involves either even layers ($j = 0, 2, \ldots, N$), or odd layers ($j = 1, 3, \ldots, N-1$) but does not mix in between them. (We assume for simplicity that N is even.) Since the boundary stencils $L^{(1)}$ and $L^{(N-1)}$ respect this separation, the Poisson problem on the odd subgrid decouples completely. (Here, we assume for simplicity that the integral normalization is discretized separately on each subgrid. Moreover, the accuracy of the integration rule is assumed to be at least fourth order, which again simplifies the argument.) To analyze the solution of

$$\begin{aligned} (Lq_{\text{odd}} - D\mathbf{a})_{ij} &= 0, \qquad j = 1, 3, \ldots, N-1, \\ (D_2q_{\text{odd}} - a_2)_{ij} &= 0, \qquad j \in \{1, N-1\}, \end{aligned} \tag{5.6}$$

with the discretized integral normalization, we assume an expansion of the form

$$q_{\text{odd}} = q^{(0)}_{\text{odd}} + hq^{(1)}_{\text{odd}} + h^2q^{(2)}_{\text{odd}} + h^3q^{(3)}_{\text{odd}} + \mathcal{O}(h^4). \tag{5.7}$$

Inserting (5.7) into (5.6), performing a Taylor expansion, and considering equal orders in h, we find equations satisfied by the coefficients. In lowest order, the original problem

$$\Delta q^{(0)} = \operatorname{div}\mathbf{a}, \qquad \frac{\partial q^{(0)}}{\partial x_2} = a_2 \quad \text{on } \partial\Omega, \qquad \int_\Omega q^{(0)}\,dx = 0 \tag{5.8}$$

is recovered. Due to the second order accuracy of the stencils in (5.6), the equations for $q^{(1)}_{\text{odd}}$ and $q^{(3)}_{\text{odd}}$ are homogeneous so that $q^{(1)}_{\text{odd}} = q^{(3)}_{\text{odd}} = 0$ because of unique solvability. In second order, we find $q^{(2)}_{\text{odd}}$ as solution of

$$\begin{aligned} \Delta q^{(2)} &= \frac{1}{6}\left(\frac{\partial^3 a_1}{\partial x_1^3} + \frac{\partial^3 a_2}{\partial x_2^3}\right) - \frac{1}{3}\left(\frac{\partial^4}{\partial x_1^4} + \frac{\partial^4}{\partial x_2^4}\right) q^{(0)}, \\ \frac{\partial q^{(2)}}{\partial x_2} &= -\frac{1}{6}\frac{\partial^3 q^{(0)}}{\partial x_2^3} \quad \text{on } \partial\Omega, \qquad \int_\Omega q^{(2)}\,dx = 0. \end{aligned} \tag{5.9}$$

Altogether, we obtain the expansion

$$q_{\text{odd}} = q^{(0)}_{\text{odd}} + h^2q^{(2)}_{\text{odd}} + \mathcal{O}(h^4). \tag{5.10}$$

In the next step we can solve the problem on the even subgrid which uses q_{odd} in the boundary conditions

$$
\begin{aligned}
(Lq_{\text{even}} - D\mathbf{a})_{ij} &= 0 \qquad j = 0, 2, \dots, N, \\
\frac{1}{3}\Big[(D_2 q_{\text{even}} - a_2)_{i,-1} + 3(D_2 q_{\text{even}} - a_2)_{i,1}\Big] & \\
&= \frac{1}{3}(D_2 q_{\text{odd}} - a_2)_{i,2} + \frac{h}{3}(D_1 D_1 q_{\text{even}} - D_1 a_1)_{i,0}.
\end{aligned} \tag{5.11}
$$

(A corresponding condition applies on the upper boundary.) The coefficients in

$$q_{\text{even}} = q^{(0)}_{\text{even}} + h q^{(1)}_{\text{even}} + h^2 q^{(2)}_{\text{even}} + h^3 q^{(3)}_{\text{even}} + \mathcal{O}(h^4)$$

are determined with the same procedure as above. It turns out that $q^{(0)}_{\text{even}} = q^{(0)}_{\text{odd}}$ is the solution of (5.8) and $q^{(1)}_{\text{even}}$ solves

$$\Delta q^{(1)} = 0, \qquad \frac{\partial q^{(1)}}{\partial x_2} = \frac{1}{4}\left(\frac{\partial^2 q^{(0)}}{\partial x_1^2} - \frac{\partial a_1}{\partial x_1}\right) \quad \text{on } \partial\Omega, \qquad \int_\Omega q^{(1)}\, dx = 0.$$

In general, the solution to this problem is nontrivial. However, if $\mathbf{a} = \nu\Delta\mathbf{u}$ and $q^{(0)} = p$ are related to a smooth solution of the Stokes equation, a compatibility condition assures that $\partial^2 q^{(0)}/\partial x_1^2 = \partial a_1/\partial x_1$ so that, in fact, $q^{(1)}_{\text{even}} = 0$. (Basically, the Stokes equation implies $\nabla p = \nu\Delta\mathbf{u}$ on the boundary so that the compatibility relation follows by applying an additional x_1-derivative.)

For the coefficient $q^{(2)}_{\text{even}}$, we recover (5.9) with boundary condition

$$\frac{\partial q^{(2)}_{\text{even}}}{\partial x_2} = \frac{\partial q^{(2)}_{\text{odd}}}{\partial x_2}$$

so that also $q^{(2)}_{\text{even}} = q^{(2)}_{\text{odd}}$. Only in third order, we observe a difference between the solutions on the odd and even subgrids. While $q^{(3)}_{\text{odd}} = 0$, the coefficient $q^{(3)}_{\text{even}}$ solves

$$
\begin{aligned}
\Delta q^{(3)} = 0, \qquad \int_\Omega q^{(3)}\, dx = 0, & \\
\frac{\partial q^{(3)}}{\partial x_2} = \frac{1}{12}\left(\frac{\partial^4 q^{(0)}}{\partial x_2^4} - \frac{\partial^3 a_2}{\partial x_2^3}\right) - \frac{1}{4}\left(\frac{\partial^2 q^{(2)}}{\partial x_2^2} + \frac{1}{6}\frac{\partial^4 q^{(0)}}{\partial x_2^4}\right) & \quad \text{on } \partial\Omega.
\end{aligned}
$$

Combining the solutions on the subgrids we find the expansion

$$q_{ij} = q^{(0)}_{ij} + h^2 q^{(2)}_{ij} + h^3 q^{(3)}_{ij} + (-1)^j h^3 \hat{q}^{(3)}_{ij} + \mathcal{O}(h^4),$$

where $q^{(0)}, q^{(2)}$ solve (5.8) and (5.9) and

$$q^{(3)} = \frac{1}{2}(q^{(3)}_{\text{even}} + q^{(3)}_{\text{odd}}), \qquad \hat{q}^{(3)} = \frac{1}{2}(q^{(3)}_{\text{even}} - q^{(3)}_{\text{odd}}).$$

In the case of the Stokes (or Navier–Stokes) equation, the Poisson problem is coupled with the evolution equation for the velocity field $\mathbf{u}$, giving rise to a similar expansion

$$\mathbf{u}_{ij} = \mathbf{u}^{(0)}_{ij} + h^2 \mathbf{u}^{(2)}_{ij} + h^3 \mathbf{u}^{(3)}_{ij} + (-1)^j h^3 \hat{\mathbf{u}}^{(3)}_{ij} + \mathcal{O}(h^4). \tag{5.12}$$

The input $\mathbf{a}$ to the Poisson problem is given by $\nu\Delta\mathbf{u}$ where, in Chorin's method, the Laplacian is discretized with the usual five-point stencil. Applying this stencil to (5.12), the term $(-1)^j h^3 \hat{\mathbf{u}}_{ij}^{(3)}$ yields the alternating contribution

$$(-1)^j \left(h \begin{bmatrix} & -1 & \\ 1 & -4 & 1 \\ & -1 & \end{bmatrix} \hat{\mathbf{u}}^{(3)} \right)_{ij}.$$

Since

$$h \begin{bmatrix} & -1 & \\ 1 & -4 & 1 \\ & -1 & \end{bmatrix} = -4h + h^3 \left(\frac{\partial^2}{\partial x_1^2} - \frac{\partial^2}{\partial x_2^2} \right) + \mathcal{O}(h^5),$$

the source term $\mathbf{a}$ in the Poisson equation contains an alternating, first order contribution so that the problem for $q^{(1)}$ now has a nontrivial alternating source. This implies that the approximate pressure p in the Stokes problem is only first order accurate with an alternating error. (For a more detailed analysis, we refer again to [34].)

Let us now define boundary stencils for the new discretization derived in section 4. The discrete gradient $\bar{G}$ is always given by $\bar{D}_1$, $\bar{D}_2$ defined in (3.4). The divergence $\bar{D}$ is also composed of these stencils for points inside the domain ($j = 2, \ldots, N-2$). The corresponding Laplacian has the form

$$\bar{L} = \bar{D}\bar{G} = \frac{1}{72} \begin{bmatrix} 1 & 4 & 8 & 4 & 1 \\ 4 & & -8 & & 4 \\ 8 & -8 & -36 & -8 & 8 \\ 4 & & -8 & & 4 \\ 1 & 4 & 8 & 4 & 1 \end{bmatrix}. \tag{5.13}$$

At points next to the boundary, we modify $\bar{D}_1$ and $\bar{D}_2$ by deleting boundary weights giving rise to $\bar{D}^{(j)}$ and $\bar{L}^{(j)} = \bar{D}^{(j)}\bar{G}$ for $j = 1, N-1$. On the boundary itself, we replace $\bar{D}_1$ by zero (as in Chorin's method) and choose half-sided differences for $\bar{D}_2$. Although these half-sided stencils do not follow directly from the LB approach, there is a natural choice based on the observation that $\bar{D}_2$ can be written as convex combination of central differences

$$\bar{D}_2 = \bar{C}D_2 \qquad \text{with} \qquad \bar{C} = \frac{1}{6}\begin{bmatrix} 1 & 4 & 1 \end{bmatrix}.$$

As in Chorin's method, where D_2 is replaced by $D_2^{(0)}$ (a second order, half-sided approximation with deleted central weight) we set

$$\bar{D}_2^{(0)} = \bar{C}D_2^{(0)} = \frac{1}{12h} \begin{bmatrix} -1 & -4 & -1 \\ 4 & 16 & 4 \\ & 0 & \\ & 0 & \\ & 0 & \end{bmatrix}.$$

A similar choice at the upper boundary yields $\bar{D}_2^{(N)}$ and finally gives rise to $\bar{L}^{(j)} = \bar{D}^{(j)}\bar{G}$ for $j = 0, N$.

In contrast to the wide Laplacian (5.3), the stencil (5.13) does not decouple odd and even subgrids. However, the coupling is not very strong which can be seen if $\bar{L}$ is

applied to a function which is constant in x_1 direction. Then, the stencil has the same effect as (5.3) leading to the decoupling known from Chorin's method. Rewriting the boundary stencils in terms of the interior discretization, we again find two different realizations of the Neumann boundary condition (which is an important structural feature for the occurrence of alternating terms). At the lower boundary, we have

$$\left[\bar{C}(\bar{D}_2 q - a_2) - \frac{h}{3}(\bar{D}_1\bar{D}_1 q - D_1 a_1)\right]_{i,0} = 0,$$
$$\left[B(\bar{D}_2 q - a_2) - \frac{2h}{3}(\bar{D}_1\bar{D}_1 q - D_1 a_1)\right]_{i,0} = 0,$$

where B is defined in (5.5). If the compatibility condition $\partial^2 q/\partial x_1^2 = \partial a_1/\partial x_1$ is satisfied, both conditions differ only in third order, exactly as in the case of Chorin's method.

The fundamental difference compared to Chorin's approach is the structure of the Laplacian $\hat{L}$ (4.6) which is used to discretize the viscosity term $\nu\Delta\mathbf{u}$. We neglect second order terms in the definition of $\hat{L}$ and consider

$$\hat{L}\mathbf{u} = \begin{pmatrix} \bar{D}_1^2 u_1 + \tilde{D}_2^2 u_1 \\ \tilde{D}_1^2 u_2 + \bar{D}_2^2 u_2 \end{pmatrix}.$$

Granting that $\mathbf{u}$ has an expansion as in (5.12), the Laplacian $\hat{L}$ does not amplify the alternating term as the standard five-point stencil. In fact, applying $\hat{L}$ to $h^3 A\hat{\mathbf{u}}^{(3)}$ (where A is an alternating function with $A_{ij} = (-1)^j$) leads to an alternating term of the same order

$$h^3\hat{L}A\hat{\mathbf{u}}^{(3)} = h^3 A \begin{pmatrix} \frac{1}{9}\frac{\partial \hat{u}_1^{(3)}}{\partial x_1^2} + \frac{\partial \hat{u}_1^{(3)}}{\partial x_2^2} \\ \Delta \hat{u}_2^{(3)} \end{pmatrix} + \mathcal{O}(h^5).$$

In contrast to the five-point Laplacian, the wide stencil $\hat{L}$ requires a special treatment at points next to the boundary. As for the boundary divergence, we consider the participating stencils as convex combinations of usual central differences which are replaced by half-sided approximations if necessary. For example, $\tilde{D}_2$ defined in (3.5) can be written as $\tilde{D}_2 = \tilde{C}D_2$ with $\tilde{C} = \frac{1}{2}\begin{bmatrix}1 & 0 & 1\end{bmatrix}$. Close to the boundary, we replace D_2 by $D_{2,hs}$ defined in (5.4). Altogether, we set

$$(\hat{L}^{(j)}\mathbf{u})_{ij} = \begin{pmatrix} \bar{D}_1 D_1 u_1 + \tilde{D}_2\tilde{C}D_{2,hs}u_1 \\ \tilde{D}_1 D_1 u_2 + \bar{D}_2\tilde{C}D_{2,hs}u_2 \end{pmatrix}_{ij}, \qquad j \in \{1, N-1\}.$$

This modified stencil now again amplifies an alternating term but only by one order

$$\hat{L}^{(j)}A\hat{\mathbf{u}}^{(3)} = \frac{4}{h}\frac{\partial}{\partial x_2}\hat{\mathbf{u}}^{(3)} + \mathcal{O}(1).$$

Consequently, a second order alternating contribution appears in the calculated pressure and adds to the second order error already present giving rise to a second order accurate scheme. (The amplification can even be avoided altogether by using $D_{2,hs}$ based on twice the grid size. Then, the approximate second x_2-derivative operates only on the even subgrid and thus is unaffected by alternating terms.)

Motivated by the analysis of the LB-induced discretization, we can set up other schemes with a similar behavior but with simpler stencils. We just use the feature of

TABLE 1
Absolute pressure errors ($*10^{-3}$) (and estimated convergence rates).

N	Chorin	Chorin (filtered)	LB-based
16	1.2321	0.6655	0.2473
32	0.5694 (1.11)	0.1604 (2.05)	0.0650 (1.93)
64	0.2703 (1.07)	0.0359 (2.16)	0.0158 (2.04)
128	0.1311 (1.04)	0.0065 (2.47)	0.0031 (2.34)

TABLE 2
Absolute velocity errors ($*10^{-3}$) (and estimated convergence rates).

N	Chorin	LB-based
16	0.8561	3.6406
32	0.2141 (2.00)	0.9249 (1.98)
64	0.0582 (1.88)	0.2215 (2.06)
128	0.0125 (2.21)	0.0444 (2.32)

a wide Laplacian in the discretization of $\nu\Delta\mathbf{u}$ to avoid amplifications of high-order alternating terms. For example, to simplify the numerical effort, one may use, instead of $\hat{L}$, the stencil $\bar{L}$—with appropriate boundary modifications—to approximate the viscous term. Another possibility is to use the wide five point stencil (5.3). As with the LB-induced discretization, $h^3 A\hat{\mathbf{u}}^{(3)}$ is not amplified inside the domain and is amplified at most by one order next to the boundary so that second order schemes in velocity and pressure are obtained.

6. Numerical investigations. Our numerical test problem is taken from [34]. The periodic channel problem (5.1) is initialized with the velocity field

$$u_1(x_1, x_2) = 6x_2(1 - x_2) + 16(2x_2 - 6x_2^2 + 4x_2^3)\sin(2\pi x_1)/2\pi,$$
$$u_2(x_1, x_2) = -16(x_2^2 - 2x_2^3 + x_2^4)\cos(2\pi x_1),$$

which relaxes to plane Poiseuille flow for $t \to \infty$. Errors due to time integration are kept small by using very small time steps. The results obtained with Chorin's method and the LB-induced discretization are presented at $t = 1$. As mentioned at the end of the previous section, we consider a simplification of the method described above, using the stencil defined by (5.3) instead of $\hat{L}$. (With $\bar{L}$ instead of $\hat{L}$, the same behavior is observed.) The orders of convergence of the pressure for the different methods are given in Table 1.

First order convergence for the pressure is found with Chorin's method in contrast to the second order convergence with the LB-based method. This is in accordance with the analytical considerations, since wide Laplacians are used to discretize the viscous terms in the velocity equation. We mention that second order convergence for the pressure can also be obtained with Chorin's method if filtering of the pressure is used; see [34]. However, filtering introduces additional errors so that Chorin's scheme including filtering shows an error larger than the LB-type method.

The convergence of velocity is second order in both cases, as shown in Table 2.

Here, the absolute errors of the LBM exceed those of Chorin's method by a factor of four. This loss of resolution is due to the fact that a wide stencil of size $2h$ is used to approximate the viscous term instead of a usual five-point stencil as in Chorin's method.

7. Conclusions. Starting from an LB-type discrete velocity model with the diffusion scaling, a relaxation system for an equivalent set of velocity moments is

derived. A simple upwind discretization of the kinetic equation, similar to the one used in the original LB scheme, gives rise to a spatial discretization of the moment system. A semi-implicit time discretization which respects the stiffness of the problem then leads to a Chorin-type projection method for the incompressible Navier–Stokes equations as limiting system. In contrast to Chorin's original method, second order convergence of the pressure is observed. This improvement is related to the use of wide stencils in the velocity equation. On the other hand, wide stencils reduce resolution so that absolute errors in velocity are larger compared to Chorin's method although both schemes are second order accurate in velocity. The analytical results are based on an analysis of alternating error terms and are supported by numerical investigations.

REFERENCES

[1] T. Abe, *Derivation of the Lattice Boltzmann method by means of the discrete ordinate method for the Boltzmann equation*, J. Comput. Phys., 131 (1997), pp. 241–246.

[2] C. Bardos, F. Golse, and D. Levermore, *Fluid dynamic limits of kinetic equations: Formal derivations*, J. Statist. Phys., 63 (1991), pp. 323–344.

[3] R. Benzi, S. Succi, and M. Vergassola, *The Lattice-Boltzmann equation: Theory and applications*, Phys. Rep., 222 (1992), pp. 145–197.

[4] R. E. Caflisch, S. Jin, and G. Russo, *Uniformly accurate schemes for hyperbolic systems with relaxation*, SIAM J. Numer. Anal., 34 (1997), pp. 246–281.

[5] N. Cao, S. Chen, S. Jin, and D. Martinez, *Physical symmetry and lattice symmetry in Lattice Boltzmann methods*, Phys. Rev. E, 55 (1997), p. 21.

[6] H. Chen, S. Chen, and W. Matthaeus, *Recovery of the Navier-Stokes equations using a Lattice-gas Boltzmann method*, Phys. Rev. A, 45 (1992), pp. 5339–5342.

[7] S. Chen and G. Doolen, *Lattice Boltzmann method for fluid flows*, Ann. Rev. Fluid Mech. 30 Annual Reviews, Palo Alto, CA, 1998, pp. 329–364.

[8] A. Chorin, *Numerical solution of the Navier-Stokes equations*, Math. Comp., 22 (1968), pp. 745–762.

[9] A. De Masi, R. Esposito, and J. Lebowitz, *Incompressible Navier Stokes and Euler limits of the Boltzmann equation*, Comm. Pure Appl. Math., 42 (1989), p. 1189.

[10] D. d'Humières, *Generalized Lattice-Boltzmann Equations*, in AIAA Rarefied Gas Dynamics: Theory and Applications, Progress in Astronautics and Aeoronautics 159, Academic Press, New York, 1992, pp. 450–458.

[11] U. Frisch, D. d'Humieres, B. Hasslacher, P. Lallemand, Y. Pomeau, and J. Rivet, *Lattice gas hydrodynamics in two and three dimensions*, Complex Systems, 1 (1987), pp. 649–707.

[12] L. Giraud, D. d'Humieres, and P. Lallemand, *A lattice Boltzmann model for Jeffreys viscoelastic fluid*, Europhys. Lett., 42 (1998), pp. 625–630.

[13] X. He and L. Luo, *A priori derivation of the lattice Boltzmann equation*, Phys. Rev. E, 55 (1997), pp. 6333–6336.

[14] X. He and L. Luo, *Lattice Boltzmann model for the incompressible Navier-Stokes equation*, J. Statist. Phys., 88 (1997), pp. 927–944.

[15] X. He and L. Luo, *Theory of the lattice Boltzmann method: From the Boltzmann equation to the lattice Boltzmann equation*, Phys. Rev. E, 56 (1997), pp. 6811–6817.

[16] S. Hou, Q. Zou, S. Chen, G. Doolen, and A. Cogley, *Simulation of cavity flow by the lattice Boltzmann method*, J. Comput. Phys., 118 (1995), p. 329.

[17] T. Inamuro, M. Yoshino, and F. Ogino, *Accuracy of the lattice Boltzmann method for small Knudsen number with finite Reynolds number*, Phys. Fluids, 9 (1997), pp. 3535–3542.

[18] S. Jin and D. Levermore, *Numerical schemes for hyperbolic conservation laws with stiff relaxation terms*, J. Comput. Phys., 126 (1996), p. 449.

[19] S. Jin, L. Pareschi, and G. Toscani, *Diffusive relaxation schemes for discrete-velocity kinetic equations*, SIAM J. Numer. Anal., 35 (1998), pp. 2405–2439.

[20] S. Jin and Z. Xin, *The relaxation schemes for systems of conservation laws in arbitrary space dimensions*, Comm. Pure Appl. Math., 48 (1995), pp. 235–276.

[21] M. Junk, *A Finite Difference Interpretation of the Lattice Boltzmann method*, Universität Kaiserslautern, Kaiserslautern, Germany, 1999, preprint.

[22] M. Junk, *Kinetic schemes in the case of low Mach numbers*, J. Comput. Phys., 151 (1999), pp. 947–968.

[23] A. Klar, *An asymptotic-induced scheme for nonstationary transport equations in the diffusive limit*, SIAM J. Numer. Anal., 35 (1998), pp. 1073–1094.
[24] A. Klar, *An asymptotic-induced numerical scheme for kinetic equations in the low mach number limit*, SIAM J. Numer. Anal., 36 (1999), pp. 1507–1527.
[25] A. Klar, *Relaxation schemes for a Lattice Boltzmann type discrete velocity model and numerical Navier-Stokes limit*, J. Comput. Phys., 148 (1999), pp. 1–17.
[26] E. Larsen and J. Morel, *Asymptotic solution of numerical transport problems in optically thick, diffusive regimes* II, J. Comput. Phys., 83 (1989), pp. 212–236.
[27] R. Mei and W. Shyy, *On the finite difference-based lattice Boltzmann method in curvilinear coordinates*, J. Comput. Phys., 143 (1998), pp. 426–448.
[28] T. Platkowski and R. Illner, *Discrete velocity models of the Boltzmann equations: A survey on the mathematical aspects of the theory*, SIAM Rev., 30 (1988), pp. 213–255.
[29] Y. Qian, D. d'Humieres, and P. Lallemand, *Lattice BGK models for the Navier–Stokes equation*, Europhys. Lett., 17 (1992), pp. 479–484.
[30] M. Reider and J. Sterling, *Accuracy of discrete velocity BGK models for the simulation of the incompressible Navier-Stokes equations*, Comput. & Fluids, 24 (1995), pp. 459–467.
[31] X. Shan and X. He, *Discretization of the velocity space in the solution of the Boltzmann equation*, Phys. Rev. Lett., 80 (1998), pp. 65–68.
[32] Y. Sone, *Asymptotic theory of a steady flow of a rarefied gas past bodies for small Knudsen numbers*, in Advances in Kinetic Theory and Continuum Mechanics, Proceedings of a Symposium Held in Honour of Henri Cabannes, R. Gatignol and Soubbaramayer, eds., Springer-Verlag, Paris, 1990, pp. 19–31.
[33] J. Sterling and S. Chen, *Stability analysis of Lattice Boltzmann methods*, J. Comput. Phys., 123 (1996), pp. 196–206.
[34] B. R. Wetton, *Analysis of the spatial error for a class of finite difference methods for viscous incompressible flow*, SIAM J. Numer. Anal., 34 (1997), pp. 723–755.
[35] B. R. Wetton, *Error analysis for Chorin's original fully discrete projection method and regularizations in space and time*, SIAM J. Numer. Anal., 34 (1997), pp. 1683–1697.

SIAM J. SCI. COMPUT.
Vol. 22, No. 1, pp. 20–38
© 2000 Society for Industrial and Applied Mathematics

RUNGE–KUTTA SOLUTIONS OF A HYPERBOLIC CONSERVATION LAW WITH SOURCE TERM*

MARK A. AVES†, DAVID F. GRIFFITHS†, AND DESMOND J. HIGHAM‡

Abstract. Spurious long-term solutions of a finite-difference method for a hyperbolic conservation law with a general nonlinear source term are studied. Results are contrasted with those that have been established for nonlinear ordinary differential equations. Various types of spurious behavior are examined, including spatially uniform equilibria that exist for arbitrarily small time-steps, nonsmooth steady states with profiles that jump between fixed levels, and solutions with oscillations that arise from nonnormality and exist only in finite precision arithmetic. It appears that spurious behavior is associated in general with insufficient spatial resolution. The potential for curbing spuriosity by using adaptivity in space or time is also considered.

Key words. adaptivity, finite difference, nonnormality, spurious solution, steady state

AMS subject classifications. 65M06, 65M50

PII. S106482759833601X

1. Introduction.

1.1. Background. There is a growing interest in the analysis of time-stepping methods for the long-term solution of nonlinear evolutionary equations. In particular, the propensity for widely used methods to generate spurious steady states is receiving much attention. In the context of ordinary differential equations (ODEs), many examples of spurious behavior have been identified [6, 12, 24]. Theoretical work in this area includes the study of spurious fixed points for small time-steps [10], the identification or construction of methods that guarantee to avoid spurious fixed points [7, 12], and the use of bifurcation theory to study routes to spuriosity [13, 20]. In [1] the potential for spurious behavior with an adaptive ODE algorithm was investigated, and both positive and negative results concerning the effectiveness of error control were derived.

Our aim in this work is to analyze spurious behavior arising from a discretized hyperbolic partial differential equation (PDE) of the form

$$u_t + (f(u))_x = g(u), \quad 0 < x < 1, \quad t > 0, \tag{1}$$

where the flux f and the source term g are nonlinear. Motivating this work is our desire to highlight new features that arise from the introduction of a spatial derivative, and hence to contrast the results with those that have been established for ODEs.

Difficulties arise in the numerical solution of (1) when the source term is "stiff"; that is, the time scale associated with reaction is very much shorter than that associated with advection. One of the most common problems addressed in the literature is that of incorrect speed of propagation of waves; see, for example, [3, 5, 15, 23]. This

*Received by the editors March 20, 1998; accepted for publication (in revised form) November 9, 1999; published electronically June 13, 2000. This research was supported by the Engineering and Physical Sciences Research Council of the UK under grants GR/K80228 and GR/M42206.
http://www.siam.org/journals/sisc/22-1/33601.html
†Department of Mathematics, University of Dundee, Dundee, DD1 4HN, Scotland (maves@mcs.dund.ac.uk, dfg@mcs.dund.ac.uk).
‡Department of Mathematics, University of Strathclyde, Glasgow, G1 1XH, Scotland (djh@maths.strath.ac.uk).

has lead to the development of methods, often implicit or semi-implicit, for time-accurate solutions. In this work we are concerned with the amplitude of the waves, rather than with the speed.

1.2. Model problem. To simplify the presentation, we focus on the case of linear advection, so that the model problem becomes

$$u_t + au_x = g(u), \quad 0 < x < 1, \quad t > 0, \tag{2}$$

where $a > 0$ is constant and the source term g is nonlinear. In section 6 we indicate how our results carry through to the more general nonlinear flux case (1). For the purpose of illustration we will frequently refer to the logistic source term

$$g(u) = \alpha u(1 - u), \tag{3}$$

where $\alpha > 0$ is constant. This prototypical nonlinear function has been used in many studies of the dynamics of numerical methods.

Making the change of coordinates $s = x - at, \tau = t$ transforms (2) to

$$\frac{du}{d\tau} = g(u). \tag{4}$$

Hence, along the characteristics, where $x-at$ is constant, solutions solve the ODE (4). (The underlying ODE (4), will, of course, also play an important role in the analysis of numerical methods for (2).) For the logistic function (3), given $u_{\tau=0} = u_0$, (4) has the solution

$$u(\tau) = \frac{u_0}{u_0[1 - \exp(-\alpha\tau)] + \exp(-\alpha\tau)}.$$

Hence, when the initial condition is positive the solution tends to the stable fixed point $u \equiv 1$ as $\tau \to \infty$.

We consider both the periodic initial value problem (PIVP) and the initial boundary value problem (IBVP) for (2). In the PIVP case, we are given $u(x, 0)$ for $0 < x < 1$ and u is assumed to be periodic in space: $u(1+x, t) = u(x, t)$. It is clear from the discussion above that with g given by (3) and positive initial data, the solution converges to the fixed point $u \equiv 1$ as $t \to \infty$.

In the IBVP case we are given $u(x, 0)$ for $0 < x < 1$ and $u(0, t)$ for $t > 0$. For simplicity, we will assume a constant boundary condition, $u(0, t) \equiv u^0$. With the logistic source term (3), if the initial and boundary data are positive, then the solution tends to a (generally) nonuniform steady state with profile

$$u(x, t) = \frac{u^0}{u^0[1 - \exp(-\alpha x/a)] + \exp(-\alpha x/a)}. \tag{5}$$

1.3. Numerical method. If the spatial derivative in (2) is approximated using first order upwind differences [21], then we obtain an ODE system of the general form

$$\mathbf{U}_t = -\tfrac{a}{\Delta x} A\mathbf{U} + \mathbf{g}(\mathbf{U}) + \tfrac{a}{\Delta x}\mathbf{b} =: S(\mathbf{U}), \tag{6}$$

where Δx is a spatial meshsize and $U_j(t) \approx u(j\Delta x, t)$. Here, $\mathbf{g}(\mathbf{U})_j = g(U_j)$. We let $N = 1/\Delta x$, so that $\mathbf{U}(t) \in \mathbb{R}^N$. For the PIVP we have

$$A = \begin{bmatrix} 1 & & & -1 \\ -1 & 1 & & \\ & \ddots & \ddots & \\ & & -1 & 1 \end{bmatrix}, \quad \mathbf{b} \equiv \mathbf{0}, \tag{7}$$

while for the IBVP

$$A = \begin{bmatrix} 1 & & & \\ -1 & 1 & & \\ & \ddots & \ddots & \\ & & -1 & 1 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} u^0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}. \tag{8}$$

To approximate the semidiscrete system (6) we consider a general explicit, two-stage, second order Runge–Kutta (RK) formula [14, p. 154] with tableau

$$\begin{array}{c|cc} 0 & 0 & \\ 1/(2\theta) & 1/(2\theta) & 0 \\ \hline & 1-\theta & \theta \end{array}.$$

Here $\theta \neq 0$ is a free parameter and we always assume that $0 < \theta \leq 1$. Common choices are $\theta = \frac{1}{2}$ and $\theta = 1$, giving the improved Euler and modified Euler methods, respectively. These time-stepping methods are widely used in the finite-difference discretization of advection problems; see, for example, [11].

Letting Δt denote the time-step, the RK method applied to (6) produces approximations $\mathbf{U}^n \in \mathbb{R}^N$ with $U_j^n \approx U(j\Delta x, n\Delta t)$ according to

$$\mathbf{U}^{n+1} = \mathbf{U}^n + \Delta t \left[(1-\theta)S(\mathbf{U}^n) + \theta S(\mathbf{U}^n + \tfrac{\Delta t}{2\theta} S(\mathbf{U}^n))\right]. \tag{9}$$

2. Fixed points of the semidiscrete problem. Although we are mainly concerned with the numerical solutions generated by the overall algorithm, a useful starting point is to study the fixed points of the system (6).

2.1. Initial BVP. For the IBVP (6) has a fixed point when

$$U_{j-1} = U_j - \tfrac{\Delta x}{a} g(U_j), \qquad j = 1, 2, \ldots, N, \tag{10}$$

with $U_0 = u^0$. Maps of this type have received considerable attention in the literature; see, for example, [16]. Generically, there are parameter ranges for which periodic cycles of any period may be generated.

The following results give insight into the behavior of the map (10) around a zero of g.

THEOREM 2.1. *Suppose $g \in C^1$ with $g(\beta) = 0$ and $g'(\beta) < 0$ for some $\beta \in \mathbb{R}$. Let $I \subseteq \mathbb{R}$ be an open, connected interval containing β such that $g'(u) < 0$ for all $u \in I$. Then, if $U_0 \in I$, there exists a solution sequence $\{U_j\}_{j=0}^N$ of (10) in which the components U_j approach β monotonically as j increases. Furthermore, this fixed point of the semidiscrete IBVP (6) is linearly stable.*

Proof. Consider the general case where $U_{j-1} \in I$ with $U_{j-1} \neq \beta$. Define $h_{j-1} : \mathbb{R} \mapsto \mathbb{R}$ by

$$h_{j-1}(u) = u - \frac{\Delta x}{a} g(u) - U_{j-1}. \tag{11}$$

Then, using the mean value theorem,

$$h_{j-1}(U_{j-1}) = -\frac{\Delta x}{a} g(U_{j-1}) = -\frac{\Delta x}{a}(g(U_{j-1}) - g(\beta)) = -\frac{\Delta x}{a} g'(\xi_{j-1})(U_{j-1} - \beta),$$

where $\xi_{j-1} \in I$. Hence, $h_{j-1}(\beta)h_{j-1}(U_{j-1}) = (\Delta x/a)g'(\xi_{j-1})(U_{j-1} - \beta))^2 < 0$; so there is a zero U_j of h_{j-1} between U_{j-1} and β.

The Jacobian of the ODE (6) at this fixed point is lower triangular with strictly negative diagonal entries. Hence, the fixed point is linearly stable. ∎

Theorem 2.1 gives the reassuring result that there are stable, smooth, fixed points of (6) that mimic those of the underlying PDE. The next result shows that unstable, oscillatory, fixed points may also exist if the spatial resolution is inadequate.

THEOREM 2.2. *Suppose $g \in C^1$ with $g(\beta) = 0$ and $g'(\beta) > 2a/\Delta x$ for some $\beta \in \mathbb{R}$. Let $I \subseteq \mathbb{R}$ be an open, connected interval containing β such that $g'(u) > 2a/\Delta x$ for all $u \in I$. Then if $U_0 \in I$ and $U_0 - \Delta x g(U_0)/a \in I$ there exists a solution sequence $\{U_j\}_{j=0}^N$ of (10) in which the components U_j approach β as j increases, with successive components lying on opposite sides of β. Furthermore, this fixed point of the semidiscrete IBVP (6) is linearly unstable.*

Proof. Consider the general case where $U_{j-1} \in I$ with $U_{j-1} \neq \beta$, and let $W_{j-1} := U_{j-1} - \Delta x g(U_{j-1})/a \in I$. From the mean value theorem,

$$(W_{j-1} - \beta)(U_{j-1} - \beta) = (U_{j-1} - \beta)^2(1 - \Delta x g'(\xi_{j-1})/a) < 0,$$

where $\xi_{j-1} \in I$. Hence, U_{j-1} and W_{j-1} lie on opposite sides of β. Let h_{j-1} be defined by (11). Then $h'_{j-1}(u) < -1$ for all $u \in I$.

Now, consider the case where $U_{j-1} < \beta$. Note that $h_{j-1}(\beta) > 0$. For any $u \in I$ with $u > U_{j-1}$ we have

$$h_{j-1}(u) = h_{j-1}(U_{j-1}) + \int_{U_{j-1}}^{u} h'_{j-1}(u)\, du < h_{j-1}(U_{j-1}) - u + U_{j-1}.$$

Putting $u = W_{j-1}$ gives $h_{j-1}(W_{j-1}) < 0$. Hence, h_{j-1} has a root $U_j \in (\beta, W_{j-1})$. Also, we note that $W_j := U_j - \Delta x g(U_j)/a = U_{j-1} \in I$.

In the case where $U_{j-1} > \beta$, a similar argument shows that h_{j-1} has a root $U_j \in (W_{j-1}, \beta)$.

Linear instability follows by observing that the relevant Jacobian of the ODE (6) is lower triangular with strictly positive diagonal entries. ∎

For the logistic source term (3), these results lead to the following corollary.

COROLLARY 2.3. *Consider the semidiscrete IBVP (6) with logistic source term (3).*

If $U_0 \in (1/2, \infty)$, then there exists a linearly stable fixed point $\{U_j\}_{j=0}^N$ of (6) for which the components U_j approach 1 monotonically as j increases.

If $0 < \psi < 1/2$ and $(1 - \psi - \sqrt{1 - 3\psi^2})/2 < U_0 < (1/2) - \psi$, where $\psi := a/(\alpha \Delta x)$, then there exists a linearly unstable fixed point $\{U_j\}_{j=0}^N$ of (6) for which the components U_j approach zero as j increases with successive components lying on opposite sides of β.

A spatially uniform fixed point (SUFP) has the form $\mathbf{U} = \beta\mathbf{e}$, where $\mathbf{e} = [1, 1, \ldots, 1]^T$. It is clear from the form of (6) that such a fixed point exists only if $g(\beta) = 0$ and $u^0 = \beta$. In this case, $\beta\mathbf{e}$ is a discrete analogue of a fixed point of (4), and hence of the problem (2). (Note that strict spatial uniformity requires the appropriate boundary condition to be specified. However, as illustrated by Theorems 2.1 and 2.2, fixed points that are almost spatially uniform arise generically.)

The Jacobian of the ODE (6) at a point $\mathbf{U} = \beta\mathbf{e}$ is $g'(\beta)I - (a/\Delta x)A$. This matrix is lower bidiagonal, and hence, it has the single eigenvalue $\lambda = g'(\beta) - a/\Delta x$. This implies that a fixed point $\mathbf{U} = \beta\mathbf{e}$ is stable for

$$g'(\beta) - \tfrac{a}{\Delta x} < 0, \tag{12}$$

a condition that is automatically satisfied if $g'(\beta) \leq 0$ and $a > 0$. However, the Jacobian may be highly nonnormal, leading to extreme transient growth of perturbations in (6) and a negligible basin of attraction for the fixed point. In this case it is more profitable to study the pseudospectrum of the Jacobian [17]. Omitting the details, the pseudospectrum of this matrix in an appropriate limit is given by (see [18])

$$g'(\beta) - \tfrac{a}{\Delta x}(1 - z), \qquad z \in \mathbb{C}, \quad |z| \leq 1.$$

In order to have stability in a practical sense when N is large, it is necessary that the pseudospectrum lie in the left half of the complex plane. This reduces to the condition $g'(\beta) < 0$, which is identical to the corresponding constraint for the PIVP (see section 2.2).

2.2. Periodic IVP. The PIVP version of (6) has a fixed point when (10) holds, with the interpretation that $U_0 \equiv U_N$. As in the IBVP case, we may study fixed points that are close to a zero of g. The arguments used in Theorems 2.1 and 2.2 remain valid, and, because of the "wrap-around" effect, the following stronger result holds.

THEOREM 2.4. *Suppose $g \in C^1$ with $g(\beta) = 0$ for some $\beta \in \mathbb{R}$.*

If $g'(\beta) < 0$, let $I \subseteq \mathbb{R}$ be the largest open, connected interval containing β such that $g'(u) < 0$ for all $u \in I$. Then the only fixed point of the semidiscrete PIVP (6) *for which $U_0 \in I$ is the linearly stable SUFP $\beta\mathbf{e}$.*

If $g'(\beta) > 2a/\Delta x$, let $I \subseteq \mathbb{R}$ be the largest open, connected interval containing β such that $g'(u) > 2a/\Delta x$ for all $u \in I$. Then the only fixed point of the semidiscrete PIVP (6) *for which $U_0 \in I$ and $U_0 - \Delta x g(U_0)/a \in I$ is the linearly unstable SUFP $\beta\mathbf{e}$.*

3. Fixed points of the overall algorithm.

3.1. Time-stepping on the underlying ODE. We begin this section by examining fixed points of the underlying RK(θ) method on the scalar ODE $u_t = g(u)$. Introducing z to represent an intermediate stage value, a fixed point β of the RK(θ) scheme must satisfy

$$\beta + \tfrac{\Delta t}{2\theta} g(\beta) = z, \tag{13}$$

$$(1 - \theta)g(\beta) + \theta g(z) = 0. \tag{14}$$

It is clear that if $g(\beta) = 0$, then $z = \beta$ is a solution—so that fixed points of the ODE are inherited by the RK scheme. (This is true for all RK schemes [12].) However, for general nonlinear g, spurious fixed points, where $g(\beta) \neq 0$, may be admitted.

It is possible to make some general comments about spurious fixed points in (13)–(14). (Recall our assumption that $0 < \theta \leq 1$.) First, from (14), there must be at least one zero of g between z and β. Moreover, if u is such that $g(u) = 0$, $g'(u) \neq 0$, and $\beta = u + c_1\epsilon + c_2\epsilon^2 + \cdots$ for some small parameter ϵ and constants $c_1, c_2, \ldots$, then

$$(1 - \theta)\beta + \theta z = u + O(\epsilon^2).$$

This tells us that z is better than β as an approximation to u when $\theta \in (1/2, 1]$ and ϵ is small.

Fixed points arising with the logistic nonlinearity (3) have been studied by many authors. For any $0 < \theta \leq 1$, the fixed point $u = 1$ is stable for $0 < \Delta t < 2$. Letting $r := \alpha\Delta t$, the following results about spurious fixed points are known for the popular choices of $\theta = 1/2$ and $\theta = 1$, [6].

$\theta = 1$: spurious fixed points $u = 1 + 2/r$ and $u = 2/r$ exist for all r. The former is stable for $0 < r < -1 + \sqrt{5}$, and the latter is stable for $2 < r < 1 + \sqrt{5}$.

$\theta = \frac{1}{2}$: spurious fixed points $u = [2 + r \pm \sqrt{r^2 - 4}]/(2r)$ exist for $r > 2$ and are stable for $2 < r < \sqrt{8}$.

Two points are worth emphasizing.

- In these examples, spurious fixed points either cease to exist or become arbitrarily large in modulus as $\Delta t \to 0$. Humphries [10] proved a general result in this vein—for any RK formula and any locally Lipschitz g, a spurious fixed point that persists for small Δt must blow up as $\Delta t \to 0$.
- The $\theta = 1$ case above shows that although spurious fixed points must blow up as $\Delta t \to 0$, they may remain stable for arbitrarily small Δt.

3.2. Spatially uniform fixed points of the PIVP. For the PIVP it is straightforward to confirm that $\beta\mathbf{e}$ is a fixed point of the overall algorithm if and only if β is a fixed point for the RK method on the scalar ODE $u_t = g(u)$. Hence, the question of *existence* of SUFPs has been answered in the previous subsection. *Stability*, however, is a separate issue.

Suppose first that $\beta\mathbf{e}$ is a nonspurious SUFP, so that $g(\beta) = 0$. Then we know from [12, Theorem 3] that the appropriate condition for linear stability is $|R(z)| < 1$ for every $z = \Delta t\lambda$, where λ is an eigenvalue of $-aA/\Delta x + g'(\beta)I$ and $R(z) = 1 + z + z^2/2$. The matrix A has eigenvalues $1 - \exp(i\varphi)$, where $\varphi \equiv \varphi_j = 2\pi j/N$, $j = 1, 2, \ldots, N$. Treating $\varphi \in [0, 2\pi)$ as a continuous variable (which may be justified when N is large), it follows that the spectrum of A lies on a circle centered on the negative real axis. It may then be shown that the real eigenvalues of A always dominate the stability condition and the required constraint may be written

$$0 < -\Delta t g'(\beta) < 2 - 2c. \tag{15}$$

For the logistic source term (3) with $\beta = 1$, this becomes $0 < r < 2 - 2c$, where $r = \alpha\Delta t$. This region is shown in light gray in the plots of Figure 1.

Generally, writing the map (9) as $\mathbf{U}^{n+1} = G(\mathbf{U}^n)$, after some manipulation the Jacobian of G at a point $\mathbf{U} = \beta\mathbf{e}$ may be written in the form

$$G'(\beta\mathbf{e}) = (1 + \rho - \tfrac{1}{2}\gamma^2)I + \tfrac{1}{2}(cA - \gamma I)^2, \tag{16}$$

where

$$\gamma = 1 + \tfrac{1}{2}\Delta t(g'(\beta) + g'(z)), \tag{17}$$

$$\rho = \Delta t((1 - \theta)g'(\beta) + \theta g'(z) + \tfrac{1}{2}\Delta t g'(\beta)g'(z)), \tag{18}$$

with z defined in (13). We use $c := a\Delta t/\Delta x$ to denote the Courant number [21]. For the logistic source term it is possible to determine the stable parameter ranges of the spurious fixed points corresponding to each value of θ. This leads to "almost triangular" regions with a vertical edge determined by the eigenvalue $\lambda(A) = 0$ and the "hypotenuse" by $\lambda(A) = 2$, $(\varphi = \pi)$. The remaining eigenvalues reduce the regions by rounding the apex in such a way as to create a C^1 smooth boundary. The details are omitted in view of their complexity. These regions are shown in darker gray in Figure 1 for $\theta = 1/2, 3/4, 1$.

Two key points arise from Figure 1. First, for $\theta > \frac{1}{2}$ it is possible to choose parameter values where both the correct and spurious fixed points are stable. In such cases, the choice of initial data will determine which, if either, of the two steady states

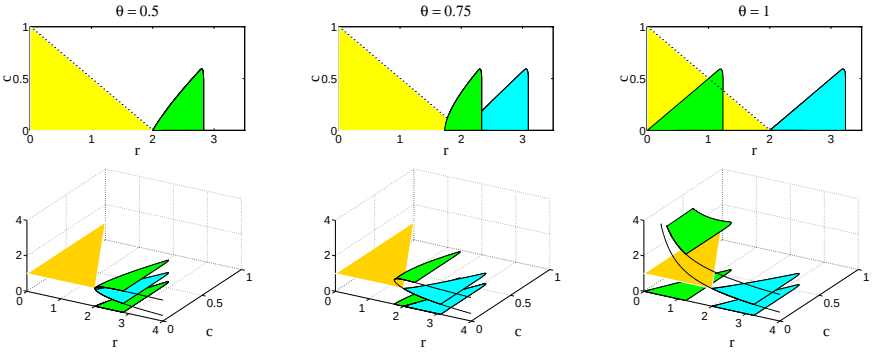


FIG. 1. *Stable parameter ranges in the (r, c)-plane for steady states of the RK(θ) methods with $\theta = 0.5, 0.75, 1.0$ on the PIVP with* (3)*. On the bottom row the vertical axis shows the magnitude of the steady state solution.*

is computed. A numerical illustration of this effect is given in section 3.4. The second point concerns the stability of spurious fixed points for small Δt. We know from section 3.1 that for $\theta = 1$ on the scalar ODE the spurious fixed point $u = 1 + 2/r$ is stable for $0 < r < -1 + \sqrt{5} \approx 1.2$. This stability interval is seen along the $c = 0$ axis at the base of the dark shaded $\theta = 1$ region in Figure 1. However, it is clear that if we increase c beyond zero, moving from the ODE to the PDE, then the spurious fixed point is always unstable for small r, that is, small Δt. We now develop some theory to show that this behavior is generic.

The lemma below concerns the scalar ODE.

LEMMA 3.1. *Consider the RK(θ) method* (9) *(with $0 < \theta \leq 1$) applied to the scalar ODE* (4)*. Suppose that $g : \mathbb{R} \mapsto \mathbb{R}$ is bounded and differentiable on every bounded interval, the zeros of g are separated, and $g(u)$ is monotonic for all $|u|$ sufficiently large. Suppose further that for sufficiently small Δt there is a spurious fixed point $\beta = \beta(\Delta t)$ with β depending C^1 continuously upon Δt and with $g(\beta)$ bounded away from zero for small Δt.*

(1) *For $\theta = 1$, we have $g'(\beta) < -2/\Delta t$ for sufficiently small Δt.*
(2) *For $0 < \theta < 1$, the fixed point β of the method* (9) *is linearly unstable for sufficiently small Δt.*

Proof. Let $z = z(\Delta t)$ be defined by (13). Note that if β is bounded as $\Delta t \to 0$, then $\Delta t g(\beta) \to 0$, so that $z \to \beta$ in (13). In this case, from (14), $g(\beta) \to 0$. This contradicts the assumption that $g(\beta)$ is bounded away from zero for small Δt, and hence, we must have $|\beta| \to \infty$ as $\Delta t \to 0$.

For the remainder of the proof we assume without comment that Δt is sufficiently small for our arguments to hold. We use a dot to represent differentiation with respect to Δt so that, for example, $\dot{\beta}$ denotes $d\beta/d\Delta t$.

For $\theta = 1$ we have $g(z) = 0$ from (14). Hence, z is fixed as $\Delta t \to 0$. It follows from (13) that $\beta g(\beta) < 0$ and hence, from the monotonicity of g, that $g'(\beta) < 0$. Differentiating (13) with respect to Δt gives

$$\beta\dot{\beta}(2 + \Delta t g'(\beta)) = -\beta g(\beta) > 0. \tag{19}$$

Now β^2 increases as $\Delta t \to 0$, so $d(\beta^2)/d\Delta t < 0$; that is, $2\beta\dot{\beta} < 0$. Using this in (19) gives $g'(\beta) < -2/\Delta t$.

For $0 < \theta < 1$ it follows from (14) that $g(\beta)$ and $g(z)$ are of opposing sign.

From the boundedness and monotonicity assumptions on g, we must have $|z| \to \infty$ as $\Delta t \to 0$, with $z\beta < 0$. In (13) we then find that

$$\beta^2 + \frac{\Delta t \beta g(\beta)}{2\theta} = z\beta < 0,$$

and hence, $\beta g(\beta) < 0$. The monotonicity of g then forces $g'(\beta) < 0$.

Now (13) and (14) give

$$2(1-\theta)z + \Delta t g(z) = 2(1-\theta)\beta. \tag{20}$$

Differentiating (13) and (20) with respect to Δt gives

$$(2\theta + \Delta t g'(\beta))\dot{\beta} + g(\beta) = 2\theta \dot{z},$$
$$(2 - 2\theta + \Delta t g'(z))\dot{z} + g(z) = 2(1-\theta)\dot{\beta}.$$

This may be written

$$\begin{bmatrix} 2\theta + \Delta t g'(\beta) & -2\theta \\ -2(1-\theta) & (2 - 2\theta + \Delta t g'(z)) \end{bmatrix} \begin{bmatrix} \dot{\beta} \\ \dot{z} \end{bmatrix} = - \begin{bmatrix} g(\beta) \\ g(z) \end{bmatrix}. \tag{21}$$

The determinant of the two-by-two matrix in (21) is 2ρ, where ρ is defined in (18). Hence,

$$2\rho \begin{bmatrix} \dot{\beta} \\ \dot{z} \end{bmatrix} = - \begin{bmatrix} (2 - 2\theta + \Delta t g'(z))g(\beta) + 2\theta g(z) \\ (2-2\theta)g(\beta) + (2\theta + \Delta t g'(\beta))g(z) \end{bmatrix}.$$

Using (14), this simplifies to

$$2\rho \begin{bmatrix} \dot{\beta} \\ \dot{z} \end{bmatrix} = - \begin{bmatrix} \Delta t g(\beta) g'(z) \\ \Delta t g(z) g'(\beta) \end{bmatrix}.$$

Thus, multiplying the first component by β gives

$$\rho \frac{d\beta^2}{d\Delta t} = -\Delta t \beta g(\beta) g'(z). \tag{22}$$

Our earlier arguments showed that $\beta g(\beta) < 0$, and since z and β tend to $\pm\infty$ in opposite directions, it follows from the monotonicity of g that $g'(z) < 0$. Hence, the right-hand side of (22) is negative. Since $d(\beta^2)/d\Delta t < 0$, we deduce that $\rho > 0$. This gives $G'(\beta) > 1$ in (16) (with the dimension set to $N = 1$ and with $A = 0$), and therefore, the spurious fixed point is unstable. ∎

We now use Lemma 3.1 to establish a result about the stability of spurious SUFPs of the semidiscrete PDE. The theorem below applies to a general class of semidiscrete problems, for which the PIVP (6) is a special case. (Adding this level of generality makes the theorem more widely applicable and does not complicate the proof.) The result shows that stable, spurious SUFPs occur only where there is a lack of spatial resolution.

THEOREM 3.2. *Consider the RK(θ) method* (9) *(with* $0 < \theta \leq 1$*) applied to an ODE system of the form*

$$\mathbf{U}_t = -\tfrac{a}{\Delta x^m} A\mathbf{U} + \mathbf{g}(\mathbf{U}), \tag{23}$$

where $a, \Delta x > 0$ *and* $\mathbf{g}(\mathbf{U})_i \equiv g(U_i)$ *with* g *satisfying the assumptions in Lemma* 3.1. *Suppose that* $A \in \mathbb{R}^{N\times N}$ *satisfies* $A\mathbf{e} = \mathbf{0}$ *and has an eigenvalue* $0 < \lambda \in \mathbb{R}$. *Consider a spurious SUFP* $\beta\mathbf{e}$, *where* $\beta = \beta(\Delta t)$ *depends* C^1 *continuously upon* Δt, *with* $g(\beta)$ *bounded away from zero for small* Δt.

(1) *For $\theta = 1$, such a SUFP cannot be stable for small Δt if*

$$\Delta x^m < \frac{a\lambda}{g'(z)}, \tag{24}$$

where the intermediate stage value z is defined in (13).

(2) *For $0 < \theta < 1$, such a SUFP cannot be stable for small Δt.*

Proof. First, note that since $A\mathbf{e} = 0$, if $\beta\mathbf{e}$ is a SUFP for the RK(θ) method on (23), then β is a fixed point for the same method on the scalar problem (4). Hence, we may appeal to Lemma 3.1.

Stability of the SUFP $\beta\mathbf{e}$ is determined by the spectrum of the matrix $G'(\beta\mathbf{e})$ in (16)–(18), with $c := a\Delta t/\Delta x^m$. Since A has an eigenvalue $\lambda \in \mathbb{R}$, $G'(\beta\mathbf{e})$ has a real eigenvalue of the form

$$\mu = 1 + \rho + \tfrac{1}{2}c^2\lambda^2 - \gamma c\lambda.$$

Instability of the SUFP $\beta\mathbf{e}$ follows if we can show that $\mu - 1 > 0$.

Consider first the case $\theta = 1$. It is straightforward to verify that

$$\mu - 1 = \tfrac{1}{2}(\lambda c - \Delta t g'(\beta) - 2)(\lambda c - \Delta t g'(z)),$$

where, from Lemma 3.1, the first factor on the right-hand side is positive for small Δt. Consequently, $\mu - 1 > 0$ if $\lambda c > \Delta t g'(z)$, from which, using $c = a\Delta t/\Delta x^m$, we obtain (24).

The result for $0 < \theta < 1$ is almost immediate from Lemma 3.1. Since $\mu > 1$ when $c = 0$, it follows by continuity that $\mu > 1$ for small c. ∎

For the picture in the top right-hand corner of Figure 1 we have $\theta = 1$, $\lambda = 2$, $m = 1$, $\beta = 1 + 2/(\alpha\Delta t)$, $z = 0$, $g'(0) = \alpha > 0$. So the condition in Theorem 3.2 for no stable spurious SUFPs when Δt is small becomes $\Delta x < 2a/\alpha$; that is, $c > r/2$. Figure 1 shows this bound to be sharp.

3.3. Spatially uniform fixed points of the IBVP. For the IBVP the PDE has a SUFP if and only if $g(u^0) = 0$. In this case, the map (9) inherits the fixed point $\beta\mathbf{e}$, with $\beta = u^0$. The Jacobian of the map at this point is lower triangular with Toeplitz form. The matrix has a single eigenvalue, which is real, and restricting this to lie in the interval $(-1, 1)$ produces the stability condition

$$0 < c - \Delta t g'(\beta) < 2. \tag{25}$$

The left-hand inequality requires $\Delta x g'(\beta) < a$, which forces Δx to be sufficiently small when $g'(\beta) > 0$, but imposes no restriction when $g'(\beta) < 0$. The right-hand inequality then requires

$$\Delta t < \frac{2\Delta x}{a - \Delta x g'(\beta)}. \tag{26}$$

However, since the Jacobian is (potentially) highly nonnormal, eigenvalues may be misleading indicators of stability. This phenomenon is well known in the analysis of linear stability of PDE methods [17] and corresponds to the classical result [19] that stability of the PIVP is a necessary condition for stability of the IBVP.

We now show the nonexistence of spurious SUFPs.

THEOREM 3.3. *For the IBVP, when $N \geq 3$, the numerical method* (9) *does not admit a spatially uniform, spurious fixed point.*

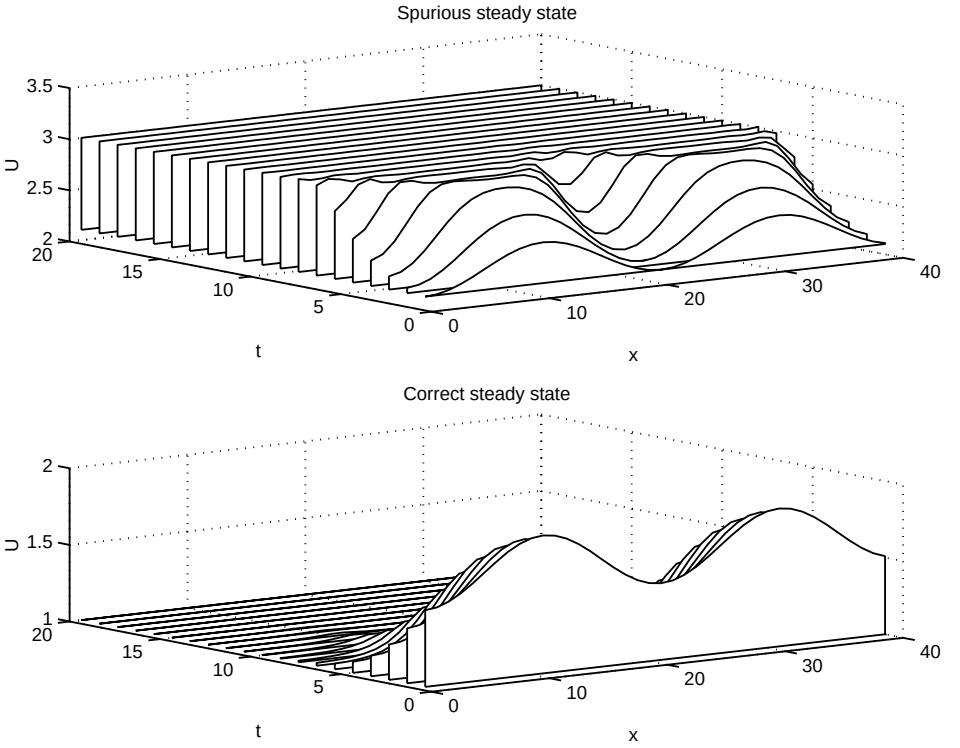


FIG. 2. *Spurious and correct steady states for the PIVP.*

Proof. If $\beta\mathbf{e}$ is a SUFP, then

$$(1-\theta)S(\beta\mathbf{e}) + \theta S\left(\beta\mathbf{e} + \frac{\Delta t}{2\theta}S(\beta\mathbf{e})\right) = 0. \tag{27}$$

From the form of S in (6), considering the jth component of (27), where $j > 2$ leads to the condition

$$(1-\theta)g(\beta) + \theta g\left(\beta + \frac{\Delta t}{2\theta}g(\beta)\right) = 0. \tag{28}$$

This shows that β must be a fixed point for the RK(θ) method applied to the underlying scalar ODE. Now, taking the second component in (27) and using (28) gives

$$u^0 - \beta = 0. \tag{29}$$

Finally, using (28) and (29) in the first component of (27) shows that

$$\frac{a\Delta t}{2\Delta x}g(\beta) = 0.$$

Hence, $g(\beta) = 0$, and the fixed point is not spurious. ∎

From the proof of Theorem 3.3 it is clear that if β is a spurious fixed point for the RK(θ) method applied to the underlying scalar ODE, then $\beta\mathbf{e}$ satisfies the conditions required for a fixed point on the IBVP problem, except near the boundary $j = 0$. We will see in the next section that it is possible for the method to settle to a fixed point that is close to β except near the left-hand boundary.

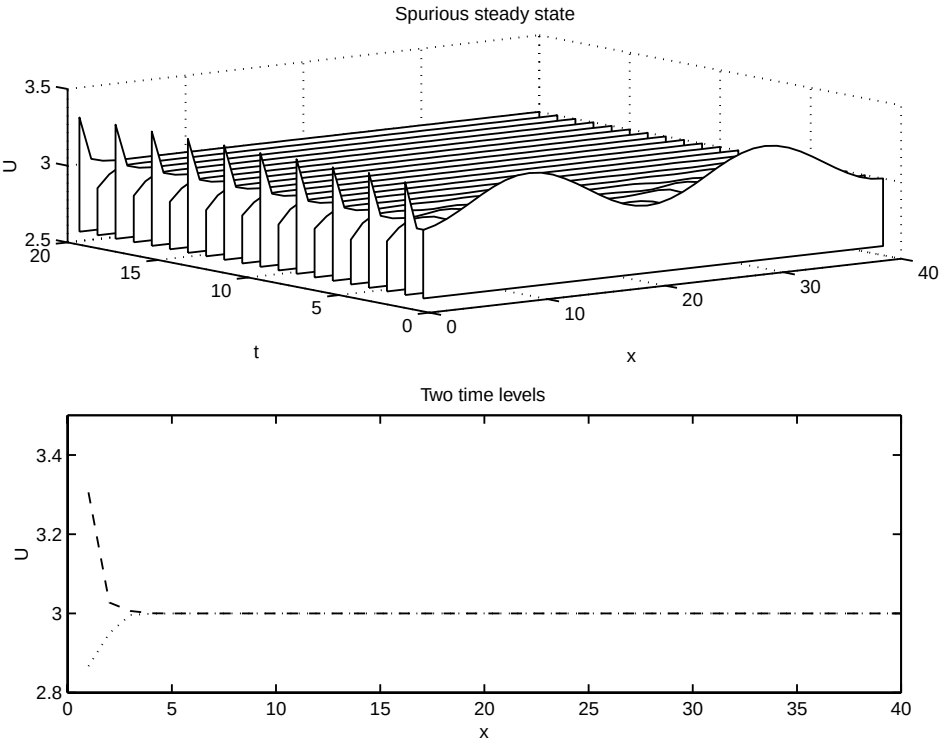


FIG. 3. *Spurious nonuniform steady state for the IBVP.*

3.4. Numerical examples and further analysis. We now present some numerical results. All computations used $a = 1$ in (2) and $\theta = 1$ in (9). We begin by illustrating that stable spurious and stable correct fixed points may coexist. Figure 2 shows the first 20 time-levels for the PIVP using $\Delta x = 1/N = 1/40$, $\Delta t = 0.1\Delta x$ and with $\alpha = 400$ in the logistic source term (3). This gives $r = 1$ and $c = 0.1$, and we see from the top right-hand picture in Figure 1 that the fixed points $\mathbf{U} = \beta\mathbf{e}$ with $\beta = \beta_{\text{corr}} := 1$ (correct) and $\beta = \beta_{\text{spur}} := 1 + 2/r = 3$ (spurious) are stable. The upper picture in Figure 2 was generated using initial data $u(x,0) = 0.7\beta_{\text{spur}} + 0.4\sin(2\pi x)^2$, and the solution is attracted to the spurious level. The lower picture used $u(x,0) = 0.5\beta_{\text{spur}} + 0.4\sin(2\pi x)^2$, which is seen to be in the basin of attraction of the true fixed point.

The upper picture in Figure 3 gives the result when the same parameter values as above are used on the IBVP with initial data $u(x,0) = \beta_{\text{spur}} + 0.3\sin(2\pi x)^2$ and a boundary condition of $u^0 = \beta_{\text{spur}}$. In this case, most components have settled close to the spurious level β_{spur}, but there are oscillations at the left-hand boundary. (For clarity, the boundary value is not plotted.) The overall solution is period two in time. The lower picture in Figure 3 gives the two profiles between which the solution oscillates. Note that Theorem 3.3 shows that a spatially uniform spurious fixed point cannot be computed for the IBVP.

In the course of our experiments, we found that, on both the IBVP and PIVP, it was common for the solution to settle down to a stable steady state that consisted of disjoint "locally uniform" patches that mix together spurious and correct levels. Figure 4 gives an example. This stable steady state arose on the PIVP with the same

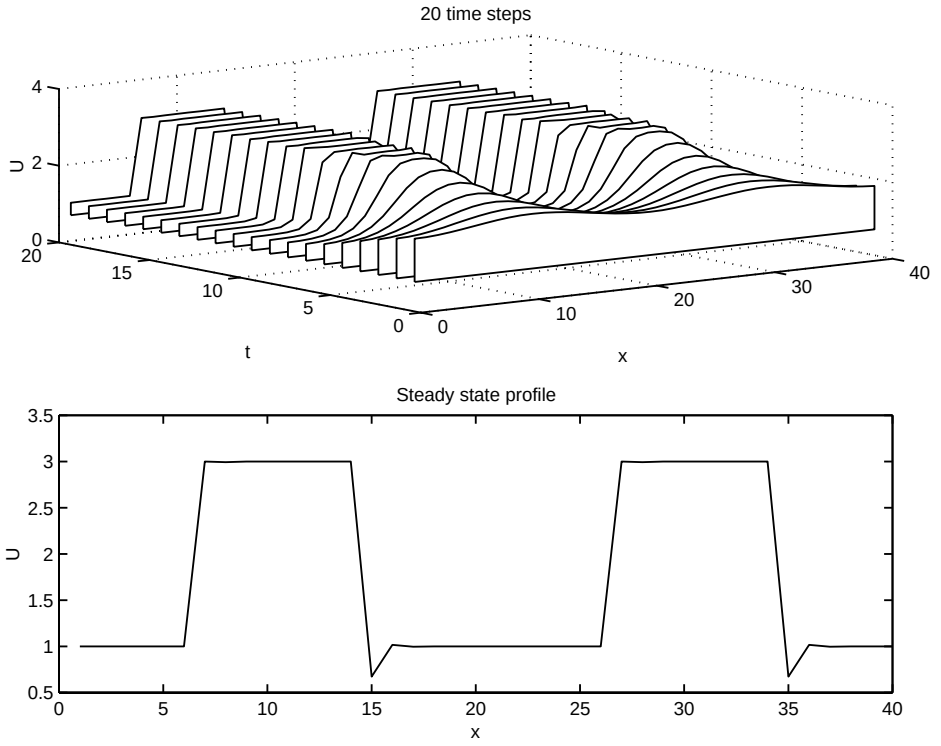


FIG. 4. *Stable steady state with jumps.*

parameters as above; we simply changed the initial condition to $u(x,0) = 0.6\beta_{\text{spur}} + 0.3\sin(2\pi x)^2$. In this example, some initial data has been attracted to the spurious level β_{spur} and some to the correct level β_{corr}. The lower picture in Figure 4 gives the profile of the steady state.

The profile shown in Figure 4 can be analyzed further, as follows. Writing the map (9) as $\mathbf{U}^{n+1} = G(\mathbf{U}^n)$, a fixed point must satisfy $\mathbf{U} = G(\mathbf{U})$, which reduces to a set of scalar equations for the components:

$$-\frac{a}{\Delta x}[U_j - U_{j-1}] + \frac{a^2\Delta t}{2\Delta x^2}[U_j - 2U_{j-1} + U_{j-2}] - \frac{a\Delta t}{2\Delta x}[g(U_j) - g(U_{j-1})] + g\left(U_j - \frac{a\Delta t}{2\Delta x}(U_j - U_{j-1}) + \frac{\Delta t}{2}g(U_j)\right) = 0,$$

for $j = 2, 3, \ldots$. Hence, the sequence $\{U_j\}_{j=0}^N$ satisfies a two-step recurrence that we may write as

$$q(U_{j-2}, U_{j-1}, U_j) = 0. \tag{30}$$

Note that as an equation for U_j (given U_{j-2} and U_{j-1}), (30) is implicit. On the other hand, if we regard (30) as an equation for U_{j-2} (given U_j and U_{j-1}), then the map is explicit.

With the logistic source term, to explain the upward jumps in Figure 4 we look for a solution sequence for (30) of the form

$$\ldots, 1, 1, 1 + \frac{2}{r} + c^2 V_0, 1 + \frac{2}{r} + c^2 V_1, 1 + \frac{2}{r} + c^2 V_2, \ldots. \tag{31}$$

Since $q(1,1,1+2/r)=0$, we take $V_0=0$. The equation $q(1,1+2/r,1+\frac{2}{r}+c^2V_1)=0$ has a solution $V_1=-2/(r^2(r+2))+O(c)$. Generally, assuming that the $\{V_j\}$ are small and linearizing produces the relation

$$V_j\,(r+2+c)\,(r-c)+2cV_{j-1}\,(1+c)-c^2V_{j-2}=0. \tag{32}$$

The characteristic polynomial of this recurrence has roots $-c/(r-c)$ and $c/(r+2+c)$. Since $r>0$ and $c>0$, it follows that $V_j\to 0$ as $j\to\infty$ in (32) if $r>c$, with very rapid convergence when $r\gg c$.

Next we look for a downward jump solution of the form

$$\ldots,1+\frac{2}{r},1+\frac{2}{r},1+V_0,1+V_1,1+V_2,\ldots. \tag{33}$$

Setting $q(1+2/r,1+2/r,1+V_0)=0$ gives the quadratic

$$V_0^2r^2+V_0^2r\,(r+c-2)-4c.$$

We will take the smaller root, for which $V_0=\frac{4c}{r(r-2)}+O(c^2)$. Linearizing $q(1+2/r,1+V_0,1+V_1)$ gives

$$V_1\approx 2c\frac{r^2V_0+rcV_0-c-V_0r}{(r+c-2)\,r\,(r+rcV_0+c)}.$$

Generally, linearizing the equation $g(1+V_{j-2},1+V_{j-1},1+Vj)=0$ gives the recurrence

$$V_j\,(r+c)\,(r+c-2)-2cV_{j-1}\,(c-1+r)+c^2V_{j-2}=0, \tag{34}$$

whose characteristic polynomial has the roots $c/(r+c),c/(r+c-2)$. It follows that a sufficient condition for $V_j\to 0$ as $j\to\infty$ in (34) is $r<2-c$.

The existence of the c^2 factor in the sequence (31) suggests that upward jumps settle to the new level more rapidly than downward jumps. This agrees with the profile in Figure 4.

Next, we show that for small c it is generic to have steady states profiles of the type pictured in Figure 4, that is, profiles that jump between levels, with each level corresponding to a stable (correct or spurious) fixed point for the RK map on the underlying scalar ODE. Note that this result formalizes an argument given in [4].

THEOREM 3.4. *Suppose $g\in C^1$. Regard Δt and Δx as fixed and allow $c=a\Delta t/\Delta x$ to vary. Suppose $\mathbf{U}^*\in\mathbb{R}^N$ is such that every component of $\mathbf{U}^*$ is a stable (correct or spurious) fixed point of the RK(θ) method on the underlying scalar ODE* (4). *Then there exists $c^*>0$ such that for all $0\le c\le c^*$ there is a stable fixed point $\mathbf{U}=\mathbf{U}(c)$ of the RK method on the PIVP* (6), *where $\mathbf{U}(c)$ depends continuously upon c and $\mathbf{U}(0)=\mathbf{U}^*$.*

Proof. Write the RK method on the PIVP (6) as

$$\mathbf{U}^{n+1}=\mathbf{U}^n+\Delta tH(c,U^n). \tag{35}$$

We know that $H(0,\mathbf{U}^*)=0$. At $c=0$ the map (35) uncouples; the Jacobian $\partial H(0,\mathbf{U})/\partial\mathbf{U}$ is diagonal. Since each component of $\mathbf{U}^*$ represents a *stable* fixed point on the underlying scalar ODE, we have

$$\left|1+\Delta t\left(\frac{\partial H(0,\mathbf{U}^*)}{\partial\mathbf{U}}\right)_{i,i}\right|<1,\quad 0\le i\le N.$$

It follows that each element on the diagonal of $\partial H(0, \mathbf{U}^*)/\partial \mathbf{U}$ is nonzero, and hence, the Jacobian is nonsingular. The implicit function theorem may therefore be invoked, to show that there exists $c^* > 0$ such that for all $0 \le c \le c^*$ there is a solution $\mathbf{U}(c)$ of the nonlinear system $H(c, \mathbf{U}(c)) = 0$, where $\mathbf{U}(c)$ depends continuously upon c and $\mathbf{U}(0) = \mathbf{U}^*$. Since the eigenvalues of a matrix depend continuously upon its entries, it follows that by further reduction of c^*, if necessary, we can ensure that the eigenvalues of $I + \Delta t \partial H(c, \mathbf{U}(c))/\partial \mathbf{U}$ remain inside the unit disc for all $0 \le c \le c^*$. □

4. Nonnormality in the IBVP. We now describe a type of spurious behavior that arises on the IBVP in the presence of finite precision and nonnormality. The effect is linear, in the sense that the same behavior can be seen when the logistic source term is replaced by $\alpha(1 - u)$.

Figure 5 shows the results of a computation on the IBVP with the logistic source term (3) using $\alpha = 400$, $\Delta x = 1/N = 1/40$, and $\Delta t = .97\Delta t_{\lim}$. Here, $\Delta t_{\lim} := 2\Delta x/(1 + \alpha \Delta x)$ is the eigenvalue stability limit, that is, the time-step beyond which the correct steady state $\mathbf{U}^n = \mathbf{e}$ becomes linearly unstable. We took $u(x, 0) = 1 + 10^{-8}$ for the initial data and $u^0 = 1$ as the boundary condition. The upper picture in Figure 5 shows the solution evolving over the first 120 time-steps (with every fourth time level plotted). In this regime, small oscillations are growing in magnitude and moving to the right. The lower picture in Figure 5 shows time-levels $n = 4010$ to $n = 4040$. This illustrates the typical long-term behavior: there are $O(1)$ oscillations around the right-hand boundary that persist for all time.

This type of behavior arises only when (a) the time step is chosen near the linear stability limit, and (b) the initial data and boundary condition are close to the true steady state. In such circumstances the numerical solution was observed either to blow up or tend to a state of the type illustrated in Figure 5.

Spurious behavior of this form was first described on a slightly different model PDE in [9], and the following simple argument explains the broad details.

After linearizing the problem, the iteration (9) has the form $\mathbf{E}^{n+1} = B\mathbf{E}^n$, where $\mathbf{E}^n = \mathbf{U}^n - \mathbf{e}$ and $B = G'(\mathbf{e})$. The matrix B has a spectral radius of .94, and hence, in exact arithmetic we have $\mathbf{E}^n \to 0$ as $n \to \infty$. However, in the presence of rounding errors the appropriate model is $\mathbf{E}^{n+1} = \widehat{B}^n \mathbf{E}^n$, where $\widehat{B}^n$ is a perturbation of B. In our case, B is highly nonnormal and so the eigenvalues of $\widehat{B}^n$ may be very different from those of B. If $\widehat{B}^n$ has an eigenvalue close to the boundary of the unit disc, then $\mathbf{E}^n$ will not decay, and hence, $\mathbf{U}^n$ will not approach the steady state $\mathbf{e}$. In our example B has a single eigenvalue .94, but there is a perturbation of Euclidean norm 10^{-15} that produces an eigenvalue with real part beyond +1. What we observe in the lower picture in Figure 5 are *pseudoeigenvectors* of B; that is, eigenvectors of perturbed matrices $\widehat{B}^n$.

5. Adaptivity. So far, we have analyzed a finite difference scheme on a uniform grid. In this section we ask whether adaption in either space or time is inimical to spuriosity.

5.1. Adaptivity in space. First, we consider a "static" spatial equidistribution algorithm of the type described in [2, 22]. The algorithm proceeds as follows. Suppose that at time-level n we have an approximation $\mathbf{U}^n$ on a spatial mesh given by $\Delta x_i^n := x_{i+1}^n - x_i^n$. Then the finite difference scheme (9) (with appropriate modifications of the spatial differences to accommodate the nonuniform mesh) is used to produce an

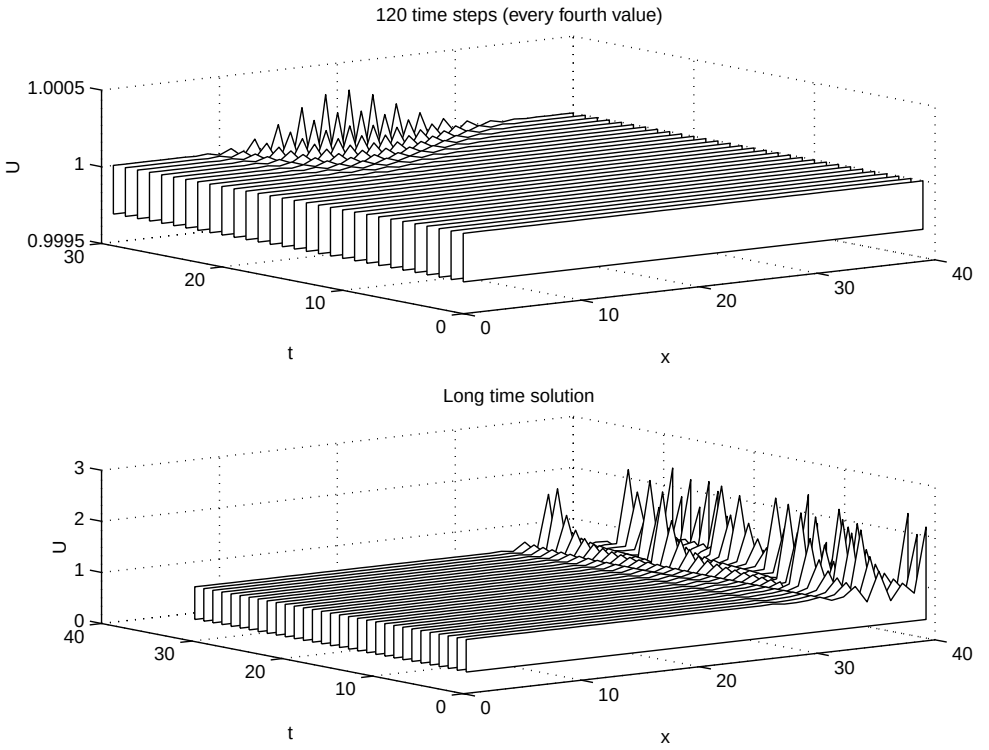


FIG. 5. *Spurious long-term behavior on the IBVP caused by nonnormality.*

approximation $\widehat{\mathbf{U}}^{n+1}$ at time-level $n+1$. We then form the weights

$$w_i^{n+1} := \sqrt{1 - \mu + \mu(d_i^{n+1})^2},$$

which involve the gradient approximations

$$d_i^{n+1} := \frac{\widehat{U}_{j+1}^{n+1} - \widehat{U}_j^{n+1}}{\Delta x_i^n}$$

and the parameter $\mu \in [0, 1]$. The new mesh $\{\Delta x_i^{n+1}\}$ is found by setting $w_i \Delta x_i^{n+1}$ equal for all i. This represents a linear system to be solved for the new mesh. The final numerical solution $\mathbf{U}^{n+1}$ is obtained by piecewise cubic interpolation from the original data $\{x_i^n, \widehat{\mathbf{U}}^{n+1}\}$ to the new mesh $\{x_i^{n+1}\}$. The motivation for this algorithm is that the weights measure the "sharpness" of the solution, and so the sharpness per unit interval is fixed. In this way, the algorithm aims to place more meshpoints in regions of x where there is rapid spatial change in the solution. The choice $\mu = \frac{1}{2}$ equidistributes a discretization of the arclength, while $\mu = 0$ gives a uniform mesh.

In experimenting with this algorithm, we fixed $a = 1$ and at each time-level chose a value $0.1 \min_i\{\Delta x_i^n\}$ for the time-step. In this way the Courant number is bounded above by 0.1.

It is immediately clear that the equidistribution algorithm described above cannot eliminate spatially uniform, spurious fixed points. (Inserting a SUFP $\mathbf{U}^n = \beta\mathbf{e}$, with $\Delta x_i^n \equiv 1/N$, we see that the solution, the spatial mesh, and the time-step are

unchanged at the next time-level.) Analyzing the *stability* of such a fixed point is a complicated process, since the solution, the spatial mesh, and the time-step may be perturbed. Also, the stability potentially depends upon the fine-details of the algorithm, such as the choice of μ and the type of interpolation process used.

In our tests, we found that some of the spatially uniform spurious fixed points that we identified for a fixed spatial mesh in section 3.2 were stable as solutions for the equidistribution algorithm. Hence, the algorithm can be made to compute spurious fixed points. However, because of the Courant number control, this behavior only arose when the initial data was close to the uniform spurious level. For other types of initial data, spatial variation of the solution caused a reduction in $\min_i\{\Delta x_i^n\}$, and hence, in the time-step, and this in turn forced the numerical solution away from the spurious level.

We also found, as we intuitively expected, that the equidistribution algorithm eliminated spurious solutions with "jumps" of the form illustrated in Figures 3 and 4.

In summary, although the spatial mesh movement offered an improvement, it did not completely eliminate the potential for spatially uniform spurious fixed points.

5.2. Adaptivity in time. We now consider adaptation of the time-step as a means to eliminate spurious behavior. In the context of ODE solvers, this issue has been looked at in [1], and positive results have been obtained about the benefit of error control.

We will analyze the standard approach of using an embedded pair of formulas for the system (6). We assume that the spatial mesh remains uniform. The second order RK formula in (9) is coupled with Euler's method, which is first order, to give

$$\mathbf{U}^{n+1} = \mathbf{U}^n + \Delta t^n \left[(1-\theta) S(\mathbf{U}^n) + \theta S\left(\mathbf{U}^n + \frac{\Delta t^n}{2\theta} S(\mathbf{U}^n) \right) \right], \tag{36}$$

$$\mathbf{U}_{\mathrm{E}}^{n+1} = \mathbf{U}^n + \Delta t^n S(\mathbf{U}^n). \tag{37}$$

The secondary formula (37) is used only in the estimation of the error and selection of a new time-step.

The step is regarded as acceptable if

$$\frac{\| \mathbf{U}^{n+1} - \mathbf{U}_{\mathrm{E}}^{n+1} \|_\infty}{\Delta t^n} \le \tau, \tag{38}$$

where $\tau > 0$ is the error tolerance. If the error criterion (38) is not satisfied, then the step is retaken with a smaller Δt^n. The details of how the time-step is altered after successful or rejected steps will not affect our conclusions. Also, we mention that changing the norm in (38) or using the error-per-step alternative $\| \mathbf{U}^{n+1} - \mathbf{U}_{\mathrm{E}}^{n+1} \|_\infty \le \tau$ would not impact the results greatly.

We first prove a result that holds when the RK pair (36)–(37) is used to solve a general ODE system. We show that any fixed point cannot be genuinely spurious, in the sense that the residual is bounded by the error tolerance.

LEMMA 5.1. *Suppose that the pair* (36)–(37) *subject to the error criterion* (38) *is used to solve a general ODE system* $\mathbf{U}_t = S(\mathbf{U})$. *If* $\mathbf{U}^n = \mathbf{U}^{n+1} = \mathbf{U}^*$, *then* $\| S(\mathbf{U}^*) \|_\infty \le \tau$.

Proof. If $\mathbf{U}^n = \mathbf{U}^{n+1} = \mathbf{U}^*$, then from (36)

$$S(\mathbf{U}^*) = \theta \left[S(\mathbf{U}^*) - S\left(\mathbf{U}^* + \frac{\Delta t^n}{2\theta} S(\mathbf{U}^*) \right) \right]. \tag{39}$$

However, the error criterion (38) ensures that

$$\left\| \theta \left[S(\mathbf{U}^*) - S\left(\mathbf{U}^* + \frac{\Delta t^n}{2\theta} S(\mathbf{U}^*) \right) \right] \right\|_\infty \leq \tau. \tag{40}$$

Combining (39) and (40) gives the result. □

This result is closely related to those proved for regular RK pairs in [8]. In fact, (36)–(37) is a very special case of a regular pair and the result is therefore stronger than those in [8]. (In particular the residual bound does not involve a Lipschitz constant of S.)

Lemma 5.1 translates immediately into a result about SUFPs of the PIVP. However, the following result is more widely applicable.

THEOREM 5.2. *Suppose that the pair* (36)–(37) *subject to the error criterion* (38) *is used for either the IBVP or PIVP. If, for some* $j \geq 2$, $U_{j-1}^n = U_j^n = U_j^{n+1} = \beta$, *then* $|g(\beta)| \leq \tau$.

Proof. Since $U_j^{n+1} = U_j^n$, we have

$$(1-\theta) S(\mathbf{U}^n)_j + \theta S\left(\mathbf{U}^n + \frac{\Delta t^n}{2\theta} S(\mathbf{U}^n) \right)_j = 0.$$

Hence,

$$S(\mathbf{U}^n)_j = \theta \left[S(\mathbf{U}^n)_j + S\left(\mathbf{U}^n + \frac{\Delta t^n}{2\theta} S(\mathbf{U}^n) \right)_j \right].$$

So the error criterion (38) ensures that

$$| S(\mathbf{U}^n)_j | \leq \tau. \tag{41}$$

Since $U_j^n = U_{j-1}^n = \beta$ and $j \geq 2$, using the form of S we find

$$S(\mathbf{U}^n)_j = g(\beta). \tag{42}$$

The result follows from (41) and (42). □

Note that this result requires only a very weak type of spatial uniformity and spuriosity: a single component must be constant locally across space and time. Hence, in addition to ruling out SUFPs on the PIVP, this type of error control is also guaranteed to eliminate "jagged" spurious fixed points of the type illustrated in Figure 4.

In our experiments with the tolerance set to $\tau = \Delta x$, no spurious fixed points were encountered.

We remark that the design of error control strategies for time-dependent PDEs involves many considerations, in particular with regard to the balance of spatial and temporal errors. Theorem 5.2 illustrates that a simple ODE-based approach offers immediate advantages from a spuriosity perspective. We are currently investigating the use of more sophisticated adaptive procedures.

6. Nonlinear flux. We briefly discuss how the results obtained in the previous sections extend to the more general conservation law (1) with nonlinear flux. The semidiscrete system (6) takes the form

$$\mathbf{U}_t = -\tfrac{1}{\Delta x} A\, \mathbf{f}(\mathbf{U}) + \mathbf{g}(\mathbf{U}) + \tfrac{1}{\Delta x} \mathbf{b},$$

where $\mathbf{f}(\mathbf{U})_j = f(U_j)$. The matrix A given by (7) or (8) is based on upwind differencing and it is therefore appropriate to assume that

$$\infty > f'(u) \geq a > 0.$$

The assumption that $f'(u)$ is bounded strictly away from zero is imposed to avoid degeneracy at fixed points β, where $f'(\beta) = 0$. The results of section 2 on existence of SUFPs may then be generalized by replacing g by the composition $ag \circ f^{-1}$ (so that g' becomes ag'/f').

The Jacobian of the fully discrete system may, in this more general setting, be defined by

$$G'(\beta\mathbf{e}) = I + \Delta t\left((1-\theta)F'(\beta) + \theta F'(z)\right) + \frac{1}{2}\Delta t^2 F'(\beta)F'(z),$$

where $F'(u) = g'(u)I - (1/\Delta x)f'(u)A$. This reduces to (16) when $f(u) = au$. Our main stability result, Theorem 3.2, then extends naturally and the condition (24) is replaced by

$$\Delta x^m < \frac{\lambda f'(z)}{g'(z)}.$$

7. Summary. This work concerns spurious behavior of finite difference schemes, with emphasis on the new features that arise on moving from an ODE to a hyperbolic PDE model.

For the case of uniform meshes, the main results may be summarized as follows:

- Theorem 3.2 for the PIVP shows that, unlike in the ODE case, as the mesh is refined in a natural manner, spatially uniform spurious fixed points are generically unstable.
- The computations and analysis in section 3.4 show that nonuniform spurious fixed points arise naturally. In particular, Theorem 3.4 and the accompanying analysis highlights the existence of stable nonsmooth steady states that jump between levels.
- The discussion in section 4 shows how a different type of spurious solution arises on the IBVP, as a consequence of nonnormality. This behavior exists only in the presence of finite precision arithmetic.

Section 5 gives results for adaptive algorithms, where the mesh is varied to take account of the solution. An equidistribution algorithm for the spatial grid offers some help in the avoidance of spurious behavior but does not completely eliminate spatially uniform spurious fixed points. In the case of a standard ODE-based time-step control, Theorem 5.2 provides a rigorous bound on the level of spuriosity.

REFERENCES

[1] M. A. Aves, D. F. Griffiths, and D. J. Higham, *Does error control suppress spuriosity?*, SIAM J. Numer. Anal., 34 (1997), pp. 756–778.

[2] C. J. Budd, G. P. Koomullil, and A. M. Stuart, *On the solution of convection-diffusion boundary value problems using equidistributed grids*, SIAM J. Sci. Comput., 20 (1998), pp. 591–618.

[3] B. Engquist and B. Sjögreen, *Robust Difference Approximation of Stiff Inviscid Detonation Waves*, Tech. Rep. CAM 91-03, Department of Mathematics, UCLA, Los Angeles, CA, 1991.

[4] D. F. Griffiths and A. R. Mitchell, *Stable periodic bifurcations of an explicit discretization of a nonlinear partial differential equation in reaction diffusion*, IMA J. Numer. Anal., 8 (1988), pp. 435–454.
[5] D. F. Griffiths, A. M. Stuart, and H. C. Yee, *Numerical wave propagation in an advection equation with a nonlinear source term*, SIAM J. Numer. Anal., 29 (1992), pp. 1244–1260.
[6] D. F. Griffiths, P. K. Sweby, and H. C. Yee, *On spurious asymptotic numerical solutions of explicit Runge-Kutta methods*, IMA J. Numer. Anal., 12 (1992), pp. 319–338.
[7] E. Hairer, A. Iserles, and J. M. Sanz-Serna, *Equilibria of Runge-Kutta methods*, Numer. Math., 58 (1990), pp. 243–254.
[8] D. J. Higham, *Regular Runge–Kutta pairs*, Appl. Numer. Math., 25 (1997), pp. 229–241.
[9] D. J. Higham and B. Owren, *Non-normality effects in a discretised, nonlinear, reaction-convection-diffusion equation*, J. Comput. Phys., 124 (1996), pp. 309–323.
[10] A. R. Humphries, *Spurious solutions of numerical methods for initial value problems*, IMA J. Numer. Anal., 13 (1993), pp. 263–290.
[11] W. Hundsdorfer, B. Koren, M. van Loon, and J. G. Verwer, *A positive finite-difference advection scheme*, J. Comput. Phys., 117 (1995), pp. 35–46.
[12] A. Iserles, *Stability and dynamics of numerical methods for nonlinear ordinary differential equations*, IMA J. Numer. Anal., 10 (1990), pp. 1–30.
[13] A. Iserles, A. T. Peplow, and A. M. Stuart, *A unified approach to spurious solutions introduced by time discretisation. Part* I: *Basic theory*, SIAM J. Numer. Anal., 28 (1991), pp. 1723–1751.
[14] J. D. Lambert, *Numerical Methods for Ordinary Differential Systems*, Wiley, New York, 1991.
[15] R. J. Leveque and H. C. Yee, *A study of numerical methods for hyperbolic conservation laws with stiff source terms*, J. Comput. Phys., 86 (1990), pp. 187–210.
[16] T.-Y. Li and J. A. Yorke, *Period three implies chaos*, Amer. Math. Monthly, 82 (1975), pp. 985–992.
[17] S. C. Reddy and L. N. Trefethen, *Stability of the method of lines*, Numer. Math., 62 (1992), pp. 235–267.
[18] L. Reichel and L. N. Trefethen, *Eigenvalues and pseudo-eigenvalues of Toeplitz matrices*, Linear Algebra Appl., 162–164 (1992), pp. 153–185.
[19] R. D. Richtmyer and K. W. Morton, *Difference Methods for Initial Value Problems*, 2nd ed., Wiley, New York, 1967.
[20] O. Stein, *Bifurcations of hyperbolic fixed points for explicit Runge–Kutta methods*, IMA J. Numer. Anal., 17 (1997), pp. 151–175.
[21] J. C. Strikwerda, *Finite Difference Schemes and Partial Differential Equations*, Wadsworth and Brooks/Cole, Belmont, CA, 1989.
[22] P. K. Sweby and H. C. Yee, *On the dynamics of some grid-adaption schemes*, in Proceedings of the Fourth International Conference on Numerical Grid Generation in CFD and Related Fields, Swansea, Wales, 1994, pp. 467–478.
[23] H. C. Yee, *A Class of High-Resolution Explicit and Implicit Shock-Capturing Methods*, NASA Technical Memorandum 101088, NASA Ames Research Center, Moffett Field, CA, 1989.
[24] H. C. Yee, P. K. Sweby, and D. F. Griffiths, *Dynamical systems approach study of spurious steady state numerical solutions of nonlinear differential equations.* 1. *The ODE connection and its implications for algorithm developments in computational fluid dynamics*, J. Comput. Phys., 97 (1991), pp. 249–310.

SIAM J. SCI. COMPUT.
Vol. 22, No. 1, pp. 39–62
© 2000 Society for Industrial and Applied Mathematics

ON EFFICIENT SOLUTIONS TO THE CONTINUOUS SENSITIVITY EQUATION USING AUTOMATIC DIFFERENTIATION*

JEFF BORGGAARD† AND ARUN VERMA‡

Abstract. Shape sensitivity analysis is a tool that provides quantitative information about the influence of shape parameter changes on the solution of a partial differential equation (PDE). These shape sensitivities are described by a continuous sensitivity equation (CSE). Automatic differentiation (AD) can be used to perform this sensitivity analysis without writing any additional code to solve the sensitivity equation. The approximate solution of the PDE uses a spatial discretization (mesh) that often depends on the shape parameters. Therefore, the straightforward application of AD introduces derivatives of the mesh. There are two drawbacks to this approach. First, extra computational effort (especially memory) is used in these calculations due to mesh sensitivities. Second, this mesh sensitivity information needs to be computed in order to obtain accurate results. In this work, we provide a methodology that avoids mesh sensitivities (and their drawbacks) by defining a modified PDE on a fixed domain (i.e., independent of the shape parameter) such that AD provides the desired approximation of the CSE. Using two examples, we demonstrate significant improvement in the computational effort, both in terms of floating point operations and memory requirements. We explain how these code modifications can be applied to a wide variety of practical problems with minimal changes to the original code. These changes are negligible when compared to the complexity of writing a separate solver for the sensitivity equation.

Key words. small shape optimization, automatic differentiation, continuous sensitivity equation, shape sensitivity, mesh sensitivity, boundary conditions, ADMAT, ADMIT

AMS subject classifications. 65K10, 65Y20, 35J99, 65N45

PII. S1064827599352136

1. Introduction.

Shape optimization problems frequently take the form of finding parameters that describe the shape of an object or a region in order to minimize a given design objective (such as weight or drag). In many practical situations, the behavior of *states* in the system can be modeled as the solution to a partial differential equation (PDE). Calculating the dependence of the state solution on these shape design parameters (the so-called *state sensitivity*) is of interest to design engineers who want to gain more insight into their problem. Furthermore, these sensitivity variables can be used to evaluate the gradient of the objective function (and other constraint functions) at a given design, which can be readily coupled with an optimization algorithm in an attempt to find the optimal parameter values.

The state sensitivity variables satisfy the continuous sensitivity equation (CSE), which can be derived formally by implicit differentiation of the state PDE and the corresponding boundary conditions [4]. This sensitivity equation is always *linear* and *shares structure* with the state PDE. In particular, these coupled PDEs share the same linearization and boundary condition type (Dirichlet, Neumann, etc.). Nearly all problems of interest require numerical techniques in order to approximate the

*Received by the editors February 23, 1999; accepted for publication (in revised form) December 3, 1999; published electronically June 13, 2000. This work was supported in part by the Air Force Office of Scientific Research under grant F49620-96-1-0329, the National Science Foundation under grants ASC-9704685, DMS-9508773, and DMS-9704509, and the Department of Energy under grant DE-FG02-90ER25013.A000.

http://www.siam.org/journals/sisc/22-1/35213.html

†Interdisciplinary Center for Applied Mathematics, Department of Mathematics, Virginia Tech, Blacksburg, VA 24061 (jborggaard@vt.edu).

‡Cornell Theory Center and Department of Computer Science, Cornell University, Ithaca, NY 14850 (verma@cs.cornell.edu).

solutions to these equations. Because these equations share the same structure, many computations which would be used to solve the state PDE alone can be reused in the solution of the sensitivity equation. Thus, obtaining the sensitivity variables can be performed for a fraction of the cost of computing the state variables.

In many cases, the software to solve the state PDE is already available, representing years of development and testing. This software needs to be modified in order to solve the coupled system. This is often impractical for a variety of reasons. Design engineers are often not experts on the simulation software, which may contain "legacy code" (where the software developer is no longer available to consult on code modifications) or "spaghetti code" (where the structure of the code is fragile or poor), making the necessary modification very difficult. In addition, the resources required to modify and debug an existing code may not be available. Automatic differentiation (AD) tools have been developed to simplify this process as they read in the original code and produce new software to solve for the state and sensitivity variables simultaneously [1, 17].

We point out that we have alluded to two fundamentally different approaches for computing the sensitivity variables. In the first, we derive the sensitivity equation and then apply solution techniques, i.e., we *differentiate-then-approximate*. In the second, we consider the traditional application of automatic differentiation. Essentially, we *approximate-then-differentiate* the state PDE. The operations of differentiation and approximation commute in many situations. For example, this fact is often exploited when developing algorithms for nonlinear problems [18]. However, when the shape of the boundary is parameter dependent, the discretization of the domain (finite difference mesh points, finite element nodes, etc.) depends upon these parameters. Derivatives of these discretization quantities, the so-called *mesh sensitivities*, appear when using the approximate-then-differentiate approach; however, they do not arise when differentiation occurs first. The result is that traditional application of AD in these problems introduces a number of "chain-rule"-like terms that contain the mesh sensitivities. Not only is there an increased overhead in calculating the mesh sensitivities themselves, but computing all of the additional terms that contain them adds significant computational effort. Some related issues in the ODE setting are discussed in [14].

Recent studies in the use of AD [11] have shown that the exploitation of the structure of the underlying algorithms can greatly enhance the performance of these tools and produce, in certain instances, code that approaches the performance of hand-coded algorithms. This requires that the design engineer expose the structure of the problem to the AD tools, and this may require minor code modifications. Finding ways to exploit the algorithm structure is currently an area of active research [11, 9, 10, 21].

In this work, we present a technique for exploiting structure at the problem level rather than at the algorithm level. This technique usually requires only minor code modification, specifically in how the boundary conditions and forcing functions are implemented. The modification is performed *before* handing the software over to AD tools. These changes essentially lead to approximating the CSE and avoid altogether the mesh derivatives and the chain-rule terms which contain them. The result is improved efficiency and a substantial reduction in memory requirements for the final differentiated code. Furthermore, eliminating mesh derivatives allows straightforward coupling of this technique with any adaptive mesh refinement strategy, a useful feature in automatic design optimization algorithms [5, 6, 8].

The remainder of this work is organized as follows. In the next section, we provide background on different techniques for computing the state sensitivities and the relationships between them. To facilitate this discussion, we introduce two example problems. The first is a nonlinear two-point boundary value problem solved by a finite element method and the second is the Laplace equation in two-dimensional coordinates solved by a finite difference method. These allow us to describe our present AD methodology in section 3. In section 4, this methodology is applied and the results demonstrate its effectiveness on these example problems. Finally, we present our conclusions and extensions in section 5.

2. Background.

2.1. Example: Nonlinear two-point boundary value problem. As our first example, we consider a nonlinear two-point boundary value problem in the interval $(0, a)$, $a > 1$,

$$\mathcal{A}_a(u(x;a)) \equiv \frac{\partial^2}{\partial x^2} u(x;a) + \frac{1}{8}\left[\frac{\partial}{\partial x} u(x;a)\right]^3 = 0 \tag{1}$$

with boundary conditions

$$u(x=0;a) = 0 \qquad \text{and} \qquad u(x=a;a) = 4.$$

The subscript on $\mathcal{A}_a$ indicates dependence of the operator on the parameter a, which occurs either through the boundary conditions or, by introducing a mapping function, through variable coefficients in the mapped operator (the operator obtained by mapping the problem to a fixed domain, e.g., (0,1) [7]).

This example was used in [7, 19] since it has sufficient complexity (nonlinear with a parameter-dependent domain) as well as a closed form solution:

$$u_{\text{ex}}(x;a) = 4\sqrt{x + \frac{(a-1)^2}{4}} - 2(a-1).$$

CSE. We define the state sensitivity for the problem (1) to be $s = \frac{\partial u}{\partial a}$. This quantity describes the behavior of the solution as the right end point (parameterized by a) is changed. As shown in [7], s solves the CSE

$$\mathcal{L}_a(u)s \equiv \frac{\partial^2}{\partial x^2} s(x;a) + \frac{3}{8}\left[\frac{\partial}{\partial x} u(x;a)\right]^2 s(x;a) = 0 \tag{2}$$

with boundary conditions

$$s(x=0;a) = 0 \qquad \text{and} \qquad s(x=a;a) = -\frac{\partial}{\partial x} u(x=a;a).$$

The term $\mathcal{L}_a(u)s$ is the linearization of $\mathcal{A}_a$ at u applied to s.

Boundary conditions for (2) are derived by setting the total (or material) derivative of u with respect to a to zero,

$$\frac{Du}{Da}(x;a) = \frac{\partial u}{\partial a}(x;a) + \frac{\partial u}{\partial x}(x;a)\Pi_a(x;a) = s(x;a) + \frac{\partial u}{\partial x}(x;a)\Pi_a(x;a) = 0,$$

where Π_a represents the derivative of the spatial location with respect to a. This term arises since the location where the boundary condition is evaluated is parameter dependent. For this problem $\Pi_a(x = 0; a) = 0$ and $\Pi_a(x = a; a) = 1$, leading to the boundary conditions in (2). An important observation is that Π_a only needs to be defined and evaluated on the boundary. Since we specify the parameterization of the shape, this quantity is well defined. It can be verified that differentiating u_{ex} with respect to a provides a closed form solution to (2) and satisfies the boundary conditions (using u_{ex}).

Note that the sensitivity equation is a linear PDE. For the case above where $\mathcal{A}_a$ is nonlinear, this equation requires the solution u in order to define $\mathcal{L}_a$. Additionally, for shape sensitivity problems ($\Pi_a \neq 0$), u is required to compute the boundary conditions. Therefore, the solution to (1) needs to be obtained *before* the sensitivity equation can be solved.

Finite element algorithm. To illustrate our AD methodology in section 3, we consider a finite element approximation to the above equations. The domain $(0, a)$ is subdivided into N elements which allow for either a piecewise linear or piecewise quadratic representation of the solution. The standard Galerkin procedure leads to a system of nonlinear algebraic equations which are solved using Newton's method. Convergence is declared when the ℓ_2-norm of the residual drops below 10^{-8}. Note that (1) and (2) share the same linearization and boundary condition types. Therefore, using a Newton method to solve the nonlinear finite element system for (1) produces the same system matrix as the finite element solution for (2) (using the previous iterate for u). The solution to the second linear system can be performed efficiently by solving the Newton system with an LU factorization and reusing this factorization to solve the second system. However, there are often computational advantages in using other linear system solvers, such as the conjugate gradient method. Thus, in many cases, code modifications to solve both linear systems may not be able to take advantage of this structure.

These algorithms are implemented in MATLAB. In particular, the Jacobian matrix is stored in the `sparse` data structure in order to minimize storage and CPU requirements. MATLAB vectorization is used in the Jacobian build to create a faster code. MATLAB features such as the sparse data structures and vectorization are exploited in the AD tools.

2.2. Example: Laplace equation in two dimensions. This example features the finite difference solution to Laplace's equation in a parameter-dependent region. The region resembles a unit square with the exception of the bottom edge, which is a multiple of the sine function (see Figure 1). Thus, we define the parameter-dependent portion of the boundary by the parametric curve

$$\Gamma_a = \{(x, a\sin(\pi x)) \mid x \in (0, 1)\}.$$

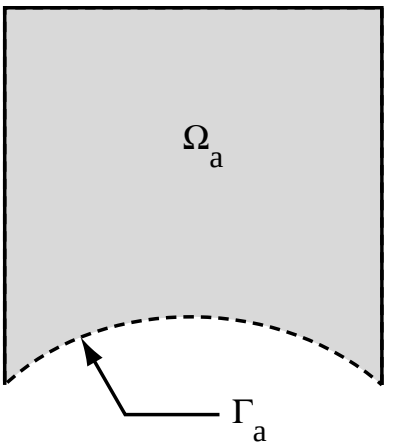


FIG. 1. *Parameter-dependent region.*

Let $\mathcal{A}_a : \mathrm{H}^1(\Omega_a) \to \mathrm{H}^{-1}(\Omega_a)$ be the Laplacian operator, $f \in \mathrm{H}^{-1}(\Omega_a)$ be a prescribed forcing function, and u be the solution to

$$\mathcal{A}_a(u(x,y;a)) \equiv \frac{\partial^2}{\partial x^2}u(x,y;a) + \frac{\partial^2}{\partial y^2}u(x,y;a) = f(x,y;a) \tag{3}$$

subject to

$$\begin{aligned} u(x=0,y;a) = u(x=1,y;a) &= 0, && y \in (0,1), \\ u(x,y=1;a) &= 0, && x \in (0,1), \quad \text{and} \\ u(x,y=a\sin(\pi x);a) &= \sin(\pi x) && x \in (0,1). \end{aligned}$$

Note that although $\mathcal{A}_a$ is linear, we use the notation $\mathcal{A}_a(u)$ to parallel the example in the previous section.

For this example, we define the function f such that u has the closed form solution

$$u_{\mathrm{ex}}(x,y;a) = \frac{1-y}{1-a\sin(\pi x)}\sin(\pi x).$$

Thus, f will be a nontrivial function of a in this example.

CSE. The CSE, which describes the state sensitivity, can be derived by performing implicit differentiation of (3) with respect to a and interchanging orders of differentiation (assuming necessary smoothness). We make an additional assumption that $\frac{\partial f}{\partial a} \in \mathrm{H}^{-1}(\Omega_a)$ so that the solution s has the same regularity as u (as we effectively solve this equation with the same approximation scheme as problem (3)).

$$\mathcal{L}_a(u)s(x,y;a) \equiv \frac{\partial^2}{\partial x^2}s(x,y;a) + \frac{\partial^2}{\partial y^2}s(x,y;a) = \frac{\partial}{\partial a}f(x,y;a) \tag{4}$$

subject to

$$\begin{aligned} s(0,y;a) = s(1,y;a) &= 0, && y \in (0,1), \\ s(x,1;a) &= 0, && x \in (0,1), \end{aligned}$$

and

$$s(x, y = a\sin(\pi x); a) = -\frac{\partial}{\partial y}u(x, y = a\sin(\pi x); a)\Pi_a(x, y = a\sin(\pi x); a), \quad x \in (0,1).$$

For this example, the function describing the influence of the parameter on the boundary, Π_a, is trivial on all sides except for Γ_a. On this side, $\Pi_a(x, y = a\sin(\pi x); a) = \sin(\pi x)$ and appears in the boundary condition above.

Note that $\mathcal{L}_a$ is a differential operator which represents the linearization of $\mathcal{A}_a$ with respect to u. As above, even though $\mathcal{A}_a$ is linear, we'll maintain this notation throughout the paper for easy generalization to nonlinear problems (as in section 2.1).

Finite difference algorithm. For purposes of applying finite differences to solve problem (3), we introduce the mapping $\mathcal{M}_a$, which is a bijection of the form

$$\mathcal{M}_a : \Omega_a \ni (x,y) \to (\xi,\eta) \in (0,1)\times(0,1) \equiv \tilde{\Omega}.$$

Transformation to the square simplifies the construction of high order difference stencils, which can now be set up on a lattice. This mapping procedure also covers the case where finite difference points are clustered in regions where more accurate differences are needed. To treat a wider class of problems with more irregularly shaped domains, domain decomposition can be used to create subdomains, each of which can be mapped in this way. In practice, algorithms for calculating this mapping may require the solution of another PDE [20].

The differential equation in Ω_a is then transformed to $\tilde{\Omega}$ by representing functions in the new coordinates, i.e.,

$$f(x,y;a) = \tilde{f}(\mathcal{M}_a(x,y;a);a) = \tilde{f}(\xi(x,y;a),\eta(x,y;a);a) = \tilde{f}(\xi,\eta;a),$$

and by replacing the differential operators in the obvious way, e.g.,

$$\frac{\partial u}{\partial x} = \frac{\partial \tilde{u}}{\partial \xi}\frac{\partial \xi}{\partial x} + \frac{\partial \tilde{u}}{\partial \eta}\frac{\partial \eta}{\partial x}.$$

Transforming problem (3) in this way leads to the equation

$$\tilde{\mathcal{A}}_a\left(\tilde{u}(\xi,\eta;a)\right) = \tilde{f}(\xi,\eta;a)$$

or

$$\begin{aligned} &\left(\left(\frac{\partial \xi}{\partial x}\right)^2 + \left(\frac{\partial \xi}{\partial y}\right)^2\right)\frac{\partial^2 \tilde{u}}{\partial \xi^2} + 2\left(\frac{\partial \xi}{\partial x}\frac{\partial \eta}{\partial x} + \frac{\partial \xi}{\partial y}\frac{\partial \eta}{\partial y}\right)\frac{\partial^2 \tilde{u}}{\partial \xi \partial \eta} + \left(\left(\frac{\partial \eta}{\partial x}\right)^2 + \left(\frac{\partial \eta}{\partial y}\right)^2\right)\frac{\partial^2 \tilde{u}}{\partial \eta^2} \\ &\qquad + \left(\frac{\partial^2 \xi}{\partial x^2} + \frac{\partial^2 \xi}{\partial y^2}\right)\frac{\partial \tilde{u}}{\partial \xi} + \left(\frac{\partial^2 \eta}{\partial x^2} + \frac{\partial^2 \eta}{\partial y^2}\right)\frac{\partial \tilde{u}}{\partial \eta} = f\left(\mathcal{M}_a^{-1}(\xi,\eta);a\right), \end{aligned} \tag{5}$$

subject to

$$\begin{aligned} \tilde{u}(\xi=0,\eta;a) = \tilde{u}(\xi=1,\eta;a) &= 0, && \eta \in (0,1), \\ \tilde{u}(\xi,\eta=1;a) &= 0, && \xi \in (0,1), \quad \text{and} \\ \tilde{u}(\xi,\eta=0;a) &= \sin\left(\pi \mathcal{M}_a^{-1}(\xi,\eta=0)\right), && \xi \in (0,1). \end{aligned}$$

Once the mapping is defined, the PDE in $\tilde{\Omega}$ can be readily approximated since finite difference approximations in $\tilde{\Omega}$ are performed on a lattice,

$$\frac{\partial \tilde{u}}{\partial \xi}(\xi,\eta;a) \approx \frac{u(\xi+\Delta\xi,\eta;a) - u(\xi-\Delta\xi,\eta;a)}{2\Delta\xi}, \text{ etc.}$$

A finite difference method is applied to (5) in the usual manner, considering second order central differences for all of the derivative terms. The resulting numerical approximation to the PDE in $\tilde{\Omega}$ is denoted by $\tilde{u}^N$, representing $\tilde{u}$ at the discrete points of the lattice. The finite difference approximation defines an algebraic equation

$$\tilde{A}_a(\tilde{u}^N) = \tilde{b}. \tag{6}$$

Note that $\tilde{b}$ contains point evaluations of the forcing function and the nonhomogeneous values of the boundary conditions. The solution to (6) can be found by solving the linear system and this solution can be mapped back to Ω_a ($u^N(x,y;a) = \tilde{u}^N(\mathcal{M}(x,y;a);a)$).

2.3. Discrete sensitivity equations and mesh sensitivities. In this section, we introduce the concept of mesh sensitivities and emphasize the differences and similarities between the differentiate-then-approximate and approximate-then-differentiate approaches to obtain state sensitivities. To simplify this discussion, we restrict our attention to the finite difference algorithm from the previous section.

Rather than deriving the sensitivity equation and finding an approximation for it, many practitioners apply implicit differentiation to the discrete form of the equations, e.g., (6), in order to determine the dependence of the solution on the parameter a. Straightforward differentiation leads to

$$\nabla_u \tilde{A}_a(\tilde{u}^N) \frac{\mathrm{D}\tilde{u}^N}{\mathrm{D}a} = \nabla_a \tilde{b} - \tilde{B}_a(\tilde{u}^N). \tag{7}$$

The Jacobian matrix on the left-hand side of the equation above is the linearization of $\tilde{A}_a$ and may be available in factored form if a direct solver is used to solve (6). The term $\nabla_a \tilde{b}$ contains terms arising from differentiating the boundary conditions and forcing functions in $\tilde{b}$. The expression $\tilde{B}_a$ contains terms which involve derivatives of the mapping $\mathcal{M}_a$ with respect to a. In other words, these are terms from the differentiation of the variable coefficients in $\tilde{A}$ that include

$$\frac{\partial \xi}{\partial a}, \ldots, \frac{\partial^2 \xi}{\partial a \partial x}, \ldots, \frac{\partial^3 \xi}{\partial a \partial x^2}, \ldots, \text{etc.}$$

All of the above terms are collectively referred to as the mesh sensitivities, since they represent how the discrete points in Ω (or the mesh) depend on a. It is these quantities and the evaluation of $\tilde{B}_a$ that we wish to avoid altogether.

Of course, it is also possible to apply the finite difference technique directly to the sensitivity equation (4). Thus, we transform the sensitivity equation to $\tilde{\Omega}$ as for (3): s to $\tilde{s}$, the differential operator $\mathcal{L}_a$ to $\tilde{\mathcal{L}}_a$, etc. Linearization of $\mathcal{A}_a$ with respect to u

commutes with the approximation scheme (as alluded to earlier, this is the premise of deriving many Newton algorithms). Thus, $\tilde{L}_a(\tilde{u}^N) = \nabla_u \tilde{A}_a(\tilde{u}^N)$ and the linear system for $\tilde{s}^N$ is

$$\nabla_u \tilde{A}_a(\tilde{u}^N)\tilde{s}^N = \tilde{c}. \tag{8}$$

The right-hand side vector contains the forcing function and boundary conditions (analogous to the construction of $\tilde{b}$).

Note that the linear systems for $\frac{\mathrm{D}\tilde{u}^N}{\mathrm{D}a}$ in (7) and for $\tilde{s}^N$ in (8) share the same system matrix. The differences in the right-hand sides reflect differences in the meaning of $\frac{\mathrm{D}\tilde{u}^N}{\mathrm{D}a}$ and $\tilde{s}^N$. As implied by our notation "D" in (7), when differentiation is performed on the discretized equations, the total (or material) derivative of $\tilde{u}^N$ is computed. Thus, these sensitivities capture the "convective" behavior of the solution as a is varied. This behavior is represented by the first two terms on the right-hand side below,

$$\frac{\mathrm{D}\tilde{u}^N}{\mathrm{D}a} = \frac{\partial \tilde{u}^N}{\partial \xi}\frac{\partial \xi}{\partial a} + \frac{\partial \tilde{u}^N}{\partial \eta}\frac{\partial \eta}{\partial a} + \frac{\partial \tilde{u}^N}{\partial a}.$$

This should be compared to $\tilde{s}^N$, which is the approximation $(\widetilde{\frac{\partial u}{\partial a}})^N$ (the operations on u here are differentiation, then transformation, then approximation). By comparing (7) and (8), we see that differentiating the approximation introduces terms to the calculations from the following identity:

$$\nabla_u \tilde{A}(\tilde{u}^N)\left[\frac{\partial \tilde{u}^N}{\partial \xi}\frac{\partial \xi}{\partial a} + \frac{\partial \tilde{u}^N}{\partial \eta}\frac{\partial \eta}{\partial a}\right] = \nabla_a \tilde{b} - \tilde{c} - \tilde{B}(\tilde{u}^N).$$

In the next section, we outline a methodology that bypasses the above calculations resulting in more efficient sensitivity calculations.

We point out that this methodology will not produce exactly the same end result that would be obtained by performing AD in the traditional fashion. In other words,

$$\left(\widetilde{\frac{\partial u}{\partial a}}\right)^N \neq \frac{\partial \tilde{u}^N}{\partial a},$$

as the first expression contains truncation errors from approximating the sensitivity equation, while the second expression contains the derivative of the truncation errors from approximating the state PDE. These errors will not be the same. However, in many cases, they both vanish as the approximations are refined. This observation motivates the notion of asymptotic consistency [2, 3], justifying the use of the CSE to obtain gradients in an optimal design algorithm.

2.4. AD strategies. AD is a chain-rule-based technique for evaluating the derivatives of functions defined by a high-level language computer program. AD relies on the fact that all computer programs, no matter how complicated, use a finite set of *elementary functions*. The `function` computed by the program is simply a composition of these elementary functions. The partial derivatives of these elementary functions are known, and thus derivatives of the `function` can be computed by propagating these derivatives via the chain rule. An introduction to AD can be found in [16].

Abstractly, the program to evaluate the solution u as a function of a (generally an m-vector) has the form

$$\begin{aligned} a &\equiv (a_1, a_2, \ldots, a_m), \\ &\downarrow \\ z &\equiv (z_1, z_2, \ldots, z_p), \quad p >> m + n, \\ &\downarrow \\ u &\equiv (u_1, u_2, \ldots, u_n), \end{aligned}$$

where the intermediate variables z are related through a series of these elementary functions which may be unary,

$$z_k = f^k_{\text{elem}}(z_i), \qquad i < k,$$

consisting of operations such as ($-$, pow$(\cdot)$, sin$(\cdot)$, $\ldots$), or binary,

$$z_k = f^k_{\text{elem}}(z_i, z_j), \qquad i < k, \quad j < k,$$

such as ($+$, $/$, $\ldots$).

AD has two basic modes of operation, the forward mode and the reverse mode. In the forward mode the derivatives are propagated along with the computation, e.g., in the elementary step $z_k = f^k_{\text{elem}}(z_i, z_j)$, the intermediate derivative, $\frac{dz_k}{da}$, can be propagated in the forward mode as

$$\frac{dz_k}{da} = \frac{\partial f^k_{\text{elem}}}{\partial z_i}\frac{dz_i}{da} + \frac{\partial f^k_{\text{elem}}}{\partial z_j}\frac{dz_j}{da}.$$

This propagation is done for all the intermediate variables z and for the output variables u. This process eventually produces the desired derivative $\frac{du}{da}$.

The reverse mode propagates the derivatives $\frac{du}{dz_k}$ for all intermediate variables backward (i.e., in the reverse order) through the computation. For example, in the elementary step $z_k = f^k_{\text{elem}}(z_i, z_j)$, the derivatives are propagated as

$$\frac{du}{dz_i} = \frac{\partial f^k_{\text{elem}}}{\partial z_i}\frac{du}{dz_k} \qquad \text{and} \qquad \frac{du}{dz_j} = \frac{\partial f^k_{\text{elem}}}{\partial z_j}\frac{du}{dz_k}. \tag{9}$$

At the end of this procedure we obtain the derivative $\frac{du}{da}$. However, the reverse mode requires saving the entire function computation, since the propagation is done backward through the computation. Hence, the partials $\frac{\partial f^k_{\text{elem}}}{\partial z_j}, \frac{\partial f^k_{\text{elem}}}{\partial z_i}$ need to be stored for the derivative computation in (9) making the reverse mode potentially prohibitive due to memory requirements.[1]

An application of AD to the discretized PDE solution results in the computation of additional terms involving the mesh. In the reverse mode these terms are stored and subsequently used for the derivative computation. This adds to the computational and memory requirements and adversely affects the performance in the reverse mode. These extra memory requirements are not necessary in the forward mode. The usage of reverse mode of AD is important for calculating gradients; the *cheap gradient*

[1]There are techniques that make the reverse mode efficient; e.g., a checkpointing technique due to Griewank [15] dramatically reduces the memory requirements with little increase in the computational requirements.

theorem [16] tells us that the gradient of any nonlinear function can be calculated for less than the cost of five function evaluations in the reverse mode of AD.

ADMAT (automatic differentiation for MATLAB) is an AD tool that can be applied to functions written for MATLAB [12]. ADMIT-1 is a MATLAB toolbox, which uses a generic AD plug-in tool (e.g., ADMAT or ADOL-C [17]) to compute the gradient and the (possibly sparse) Jacobian and Hessian matrices. The requirements from the users are minimal: the user is simply required to supply the code for the function computation and identify the variables. For complete information on using ADMIT-1, we refer the interested reader to the ADMIT-1 user manual [13]. ADMAT and ADMIT-1 are used to obtain the numerical results in section 4.

3. Methodology: Differentiation of an equivalent problem. In this section, we consider an AD methodology that avoids the need for mesh sensitivities. To facilitate this, we introduce a simple example: Let $f(a)$ be a function that is defined for $a \in \mathcal{P}$. Furthermore, assume f is differentiable over the set $\mathcal{P}$ and define a new function g with independent variables $a \in \mathcal{P}$ and $\alpha \in (-\epsilon, \epsilon)$ as

$$g(a, \alpha) = f(a) + \alpha \frac{df}{da}(a).$$

The following statements are obviously true:

$$\begin{array}{rcll} g(a, \alpha = 0) & = & f(a) & \forall a \in \mathcal{P}, \\ \frac{\partial g}{\partial a}(a, \alpha = 0) & = & \frac{df}{da}(a) & \forall a \in \mathcal{P}. \end{array}$$

Therefore, we have shifted the task of finding the derivative of f with respect to a to finding the derivative of g with respect to α at $\alpha = 0$.

In the sections below, we introduce a new parameter α and define a modified equivalent problem to aid our sensitivity analysis. While the example function g seems contrived, since we need to find $\frac{df}{da}(a)$ in order to construct g in the first place, it motivates how we can construct an equivalent problem to compute the derivatives we desire. This idea is used to construct a modified problem (PDE). Typically, we need only to define modified forcing functions and boundary conditions using the technique above. Since the modified problem is defined at the abstract level before any approximations are considered, it may be possible to perform minor modifications to an existing algorithm to produce a solution to this modified problem. Moreover, unlike the simple contrived example above, this problem definition does not require the sensitivity solution a priori but rather leads to the sensitivity solution. AD of this modified algorithm with respect to the parameter α produces an algorithm for approximating the CSE. The advantages of obviating differentiation of the algorithm with respect to a is that mesh sensitivities are not required. This has significant computational advantages as we demonstrate in section 4.

The procedure is illustrated in Figure 2. The typical application of AD follows the upper right path; first the PDE is approximated and then AD is applied. This is different from the continuous sensitivity approach along the lower left path, which first derives a linear PDE (the CSE) for the sensitivity variables and then performs approximation. The methodology below creates a modified PDE such that the approximation of this modified PDE yields the same result as approximating the original PDE. However, when AD is applied, it produces an approximation to the CSE (and inherits all of the advantages and disadvantages of doing so).

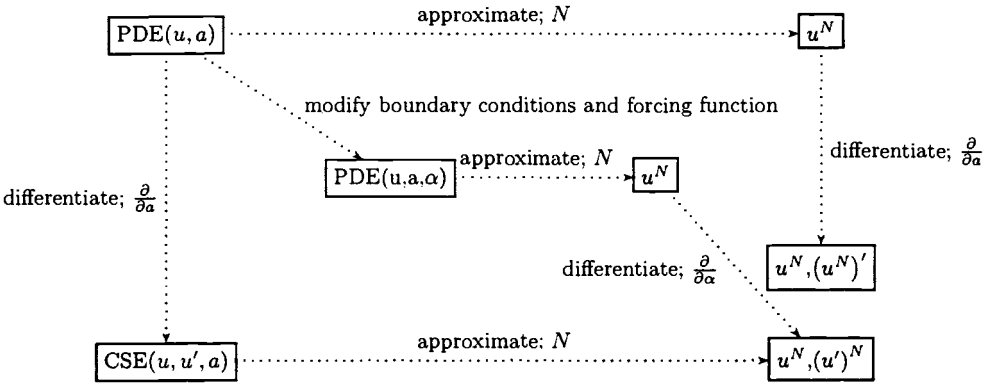


FIG. 2. *Schematic of proposed differentiation methodology.*

3.1. Differentiation methodology. The general form of the two examples from section 2 is to seek a solution u that satisfies

$$\mathcal{A}_a(u) = f \qquad \text{on } \Omega_a \tag{10}$$

subject to the boundary conditions

$$u = \hat{u} \quad \text{on } \Gamma_{\mathrm{d}} \qquad \text{and} \qquad \nabla u \cdot \hat{n} = \hat{q} \quad \text{on } \Gamma_{\mathrm{n}}$$

with $\Gamma_{\mathrm{d}} \cup \Gamma_{\mathrm{n}} = \partial\Omega_a$ and $\Gamma_{\mathrm{d}} \cap \Gamma_{\mathrm{n}} = \emptyset$. The general form of the sensitivity equation is

$$\mathcal{L}_a(u)s = g(u) \qquad \text{on } \Omega_a \tag{11}$$

subject to

$$s = \hat{s}(u) \quad \text{on } \Gamma_{\mathrm{d}} \qquad \text{and} \qquad \nabla s \cdot \hat{n} = \hat{q}_s(u) \quad \text{on } \Gamma_{\mathrm{n}}.$$

Note that the operator $\mathcal{A}_a(u)$ may depend on the parameter a. If this is the case, it results in adding a forcing function to the CSE in addition to $\frac{\partial f}{\partial a}$. This additional forcing function may also be a function of u, and the combined forcing function $g(u)$ reflects this dependence.

Next, we use the parameter α and introduce a differential equation for $v = v(x, y; a, \alpha)$ as follows. First of all, to construct a forcing function, we add α times the forcing function in (11) to the forcing function in (10). Next, we add α times the boundary conditions of the sensitivity equation (11) to the corresponding boundary conditions in the PDE (10). This leads to the following PDE with a solution v:

$$\mathcal{A}_a(v) = f + \alpha g \qquad \text{on } \Omega_a$$

with

$$v = \hat{u} + \alpha\hat{s}(v) \quad \text{on } \Gamma_{\mathrm{d}} \qquad \text{and} \qquad \nabla v \cdot \hat{n} = \hat{q} + \alpha\hat{q}_s(v) \quad \text{on } \Gamma_{\mathrm{n}}.$$

Thus, if we consider applying this to the example in section 2.2, we obtain the PDE

$$\mathcal{A}_a(v) = f + \alpha \frac{\partial f}{\partial a} \tag{12}$$

with

$$\begin{aligned} v(0,y;a,\alpha) = v(1,y;a,\alpha) &= 0, && y \in (0,1), \\ v(x,1;a,\alpha) &= 0, && x \in (0,1), \end{aligned}$$

and

$$v(x, y = a\sin(\pi x); a, \alpha) = \sin(\pi x) - \alpha \tfrac{\partial}{\partial y} v(x, y = a\sin(\pi x); a, \alpha)\sin(\pi x), \quad x \in (0,1).$$

From our assumptions on f and $\frac{\partial f}{\partial a}$, the right-hand side of (12) is in $\mathcal{H}^{-1}$. Observe the following fact.

THEOREM 3.1. *The derivative of the solution of* (3) *with respect to* a *is the same function as the derivative of the solution of* (12) *with respect to* α *at* $\alpha = 0$. *In other words,*

$$\frac{\partial u}{\partial a}(\cdot,\cdot;\cdot) = \frac{\partial v}{\partial \alpha}(\cdot,\cdot;\cdot,\alpha = 0).$$

Proof. Note that when $\alpha = 0$, (12) reduces to (3) (the same operator on the same domain with the same boundary conditions) so they have the same solution, i.e.,

$$u(\cdot,\cdot;\cdot) = v(\cdot,\cdot;\cdot,\alpha = 0).$$

Consider the sensitivity equation for problem (12) with parameter α; then $r = \frac{\partial v}{\partial \alpha}$ satisfies the following PDE:

$$\mathcal{L}_a(u) r = \frac{\partial f}{\partial a} \tag{13}$$

with

$$\begin{aligned} r(0,y;a,\alpha) = r(1,y;a,\alpha) &= 0, && y \in (0,1), \\ r(x,1;a,\alpha) &= 0, && x \in (0,1), \quad \text{and} \\ r(x, y = a\sin(\pi x); a, \alpha) &= -\tfrac{\partial}{\partial y} v(x, y = a\sin(\pi x); a, \alpha)\sin(\pi x), && x \in (0,1). \end{aligned}$$

We now show that the sensitivity of v in (12) with respect to α (our r above) evaluated at $\alpha = 0$ is the same as s in (4). It is easy to see that the derivative of the forcing function in (12) with respect to α gives the forcing function in (4). We now consider the boundary condition expressions in (12). Differentiating the Dirichlet boundary condition in (12) with respect to α leads to

$$r(x,y;a,\alpha) = -\frac{\partial}{\partial y} v(x,y;a,\alpha)\sin(\pi x) - \alpha \frac{\partial}{\partial y} r(x,y;a,\alpha)\sin(\pi x).$$

When $\alpha = 0$ this is precisely the boundary condition that appears in (4) (since $u = v$ as noted above):

$$r(x,y;a,\alpha = 0) = -\frac{\partial}{\partial y} v(x,y;a,\alpha = 0)\sin(\pi x) = -\frac{\partial}{\partial y} u(x,y;a)\sin(\pi x).$$

A similar argument holds for the Neumann boundary condition. □

It is important to point out that the parameter α has no influence on the geometry of the problem, only on the forcing function and the boundary conditions that are applied. Therefore, we propose modifying the software so that it approximates (12) instead of approximating (3). Then, application of AD tools results in software that produces an approximation to the CSE (11) and all of the unnecessary calculations associated with the mesh sensitivities are avoided.

3.2. Implementation details. We now discuss the implementation of this AD methodology for the two examples in section 2. Each example requires that we modify the approximation scheme for the PDE so that it approximates an augmented PDE. Fortunately, these modifications are only necessary in the routines that calculate forcing functions and boundary conditions. These routines typically account for a small percentage of the code.

The modification of the forcing function is straightforward. If we can explicitly differentiate the forcing function with respect to a, then this function can be modified to accept another argument, α, and return the value of $f + \alpha(\frac{\partial f}{\partial a})$. In the event that the code to evaluate f is complicated, AD can be used to compute $\frac{\partial f}{\partial a}$. A new routine, say `f_new`, can be created to call `f_original` (for f) and the differentiated version of this function (to compute $\frac{\partial f}{\partial a}$), then calculate the appropriate linear combination. For example, with the ADMIT toolbox, AD can be performed in a nested fashion to carry this out:

```
function f_new(x,y,a,alpha)
   return f_original(x,y,a) + alpha*f_original(x,y,D(a));
```

where `f_original(x,y,D(a))` is the AD of the function `f_original` with respect to `a`.

The treatment of the boundary conditions is more complex and will be specified for each example shortly. For now, we describe where this complexity arises. If we consider a boundary condition of the form

$$r = \frac{\partial \hat{u}}{\partial a} - \nabla v \cdot \Pi_a$$

(as appears in (13)), this cannot be implemented by differentiating a program with the boundary condition

$$v = \hat{u} + \alpha \left(\frac{\partial \hat{u}}{\partial a} - \nabla v \cdot \Pi_a \right).$$

Since most differentiated code solves for v and r simultaneously, we can't expect v (and ∇v in particular) to be available before we calculate r. Current AD tools generally don't have information about the structure of the problem (where the equations for v and r can be decoupled), so this type of structure needs to be exposed to the AD tools. Therefore, the boundary condition routine needs to be modified so that it imposes the condition

$$v + \alpha \nabla v \cdot \Pi_a = \hat{u} + \alpha \frac{\partial \hat{u}}{\partial a}$$

even though α will eventually be set to zero. Using this modified boundary condition allows us to impose

$$r + \alpha \nabla r \cdot \Pi_a = \frac{\partial \hat{u}}{\partial a} - \nabla v \cdot \Pi_a$$

in the differentiated code (which is our desired boundary condition when $\alpha = 0$).

Application to the nonlinear two-point boundary value problem. We consider the implementation for the problem in section 2.1. In order to approximate the sensitivity equation using our AD tools, we consider the following two-point boundary value problem instead:

$$\frac{\partial^2}{\partial x^2} v(x;a,\alpha) + \frac{1}{8}\left[\frac{\partial}{\partial x} v(x;a,\alpha)\right]^3 = 0 \tag{14}$$

with boundary conditions

$$v(x=0;a,\alpha) = 0 \qquad \text{and} \qquad v(x=a;a,\alpha) + \alpha \frac{\partial}{\partial x} v(x=a;a,\alpha) = 4.$$

Note that changes in the algorithm to compute an approximation to v rather than to u occur only in the implementation of the boundary conditions and that

$$u(x;a) = v(x;a,\alpha=0)$$

and

$$s(x;a) \equiv \frac{\partial u}{\partial a}(x;a) = \frac{\partial v}{\partial \alpha}(x;a,\alpha=0) \equiv r(x;a,\alpha=0).$$

The finite element approximation of (1) leads to a system of nonlinear algebraic equations. These equations can be solved, e.g., using Newton's method. In this case, an update of the solution δu is constructed to improve the current guess of the solution. The boundary conditions for the update are

$$\delta u(x=0;a) = 0 - u(x=0;a)$$

and

$$\delta u(x=0;a) = 4 - u(x=a;a).$$

In the algebraic system, the last boundary condition takes the form

$$\begin{aligned} L(N,N) &= 1, \\ b(N) &= 4 - u(N), \end{aligned}$$

where L is considered to be an approximation to $\mathcal{L}_a(u)$, or the Jacobian of the nonlinear algebraic equations.

For the modified problem, the boundary conditions for the update of v are

$$\delta v(x=0;a,\alpha) = 0 - v(x=0;a,\alpha)$$

and

$$\delta v(x=a;a,\alpha) + \alpha \frac{\partial}{\partial x} \delta v(x=a;a,\alpha) = 4 - \left[v(x=a;a,\alpha) + \alpha \frac{\partial}{\partial x} v(x=a;a,\alpha) \right].$$

The second boundary condition can be implemented as

$$\begin{aligned} L(N,N-1) &= -\frac{\alpha}{\Delta x}, \\ L(N,N) &= 1 + \frac{\alpha}{\Delta x}, \\ b(N) &= 4 - \left(v(N) + \frac{\alpha}{\Delta x}(v(N) - v(N-1)) \right). \end{aligned}$$

Thus, AD can be applied to (14) with respect to α, rather than to problem (1) with respect to a. The required modification only occurs in a few lines of MATLAB code.

Application to the Laplace equation in two dimensions. To use our AD methodology on the problem in section 2.2, we need to modify the finite difference algorithm so that it approximates

$$\frac{\partial^2}{\partial x^2} v(x,y;a,\alpha) + \frac{\partial^2}{\partial y^2} v(x,y;a,\alpha) = f(x,y;a) + \alpha \frac{\partial f}{\partial a}(x,y;a) \tag{15}$$

with boundary conditions

$$\begin{aligned} v(0,y;a,\alpha) = v(1,y;a,\alpha) &= 0, \qquad y \in (0,1), \\ v(x,1;a,\alpha) &= 0, \qquad x \in (0,1), \end{aligned}$$

and

$$v(x, y = a\sin(\pi x); a, \alpha) + \alpha \tfrac{\partial}{\partial y} v(x, y = a\sin(\pi x); a, \alpha) \sin(\pi x) = \sin(\pi x), \quad x \in (0,1),$$

instead of the system (3) and evaluates this at $\alpha = 0$. The code that evaluates f can be modified in a straightforward manner by passing an additional argument for α and returning the new forcing function. In this problem, $f(x,y;a)$ is given by an explicit formula; thus $\frac{\partial f}{\partial a}$ is also known explicitly. Therefore the modification is straightforward as illustrated for the one-dimensional (1-D) problem.

Once again, the modification of the boundary conditions is more challenging. Dirichlet boundary conditions in a finite difference approximation of (3) may be implemented by creating rows in the system matrix that impose

$$\tilde{u}(\xi, \eta = 0) = \sin(\pi x(\xi, \eta = 0)) \qquad \text{for } \xi = \Delta\xi, 2\Delta\xi, \ldots, 1 - \Delta\xi.$$

The boundary condition for (15) can be implemented by imposing

$$\tilde{v}(\xi,\eta=0) + \alpha \left(\frac{\partial \xi}{\partial y}(\xi,\eta=0) \frac{\partial \tilde{v}}{\partial \xi}(\xi,\eta=0) + \frac{\partial \eta}{\partial y}(\xi,\eta=0) \frac{\partial \tilde{v}}{\partial \eta}(\xi,\eta=0) \right) \sin(\pi x(\xi,\eta=0))$$
$$\text{for } \xi = \Delta\xi, 2\Delta\xi, \ldots, 1 - \Delta\xi,$$

where the terms $\frac{\partial \tilde{v}}{\partial \xi}$ and $\frac{\partial \tilde{v}}{\partial \eta}$ are implemented via finite differences, a central difference for $\frac{\partial \tilde{v}}{\partial \xi}$ (as described in section 2.2), and a one-sided difference for $\frac{\partial \tilde{v}}{\partial \eta}$.

This modification requires the addition of nonzero elements in the system matrix $\tilde{A}$ as in the previous example (in general, we would also need to make additions to $\tilde{b}$, but this could be handled in the same way as the forcing function). By modifying $\tilde{A}$ in this way, the proper boundary conditions are realized when AD is performed with respect to α.

4. Numerical results. In this section, we present numerical results for the methodology presented above. AD is performed on both the 1-D nonlinear two-point boundary value problem (1) and the two-dimensional (2-D) Laplace equation (3) as well as the corresponding modified problems, (14) and (15). Since the modified problems avoid the computation of mesh sensitivities, comparing these results (in both the forward and reverse modes of AD) illustrate the gains that can be made with our modification. We use the following three metrics to compare the performance of these differentiated codes:

- FLOPs for the forward mode. In the modified problem, the required floating point operations (FLOPs) are expected to be less than in the original problem since the computation of mesh sensitivities is avoided. Hence, this measure indicates the amount of computation that can be saved using the modified problem. FLOP1 in Tables 1 and 2 represents the FLOPs for the modified problem and FLOP2 represents the FLOPs for the original problem. These calculations are made using the forward mode of AD. The problem size, N, (the solution is of size N in the 1-D problem and of size $N \times N$ in the 2-D problem) is varied to demonstrate how these savings scale.

- Space complexity for the reverse mode. This quantity measures the memory requirements for the reverse mode of AD. Recall that the reverse mode requires storing the function trace for the derivative computation. Hence, this measure indicates how much space is saved by avoiding the mesh sensitivity terms. In Tables 1 and 2, SPACE1 represents the space complexity (the number of floating point numbers that are stored) for the modified problem and SPACE2 is the space complexity for the original problem.

- FLOPs for the reverse mode. This measure indicates how much computation can be saved by avoiding the mesh sensitivities in the reverse mode. In Tables 1 and 2, RFLOP1 represents the FLOPs for the modified problem in the reverse mode, while RFLOP2 represents the corresponding FLOPs for the original problem.

4.1. Nonlinear two-point boundary value problem. We begin by discussing results for the nonlinear two-point boundary value problem introduced in sections 2.1 and 3.2. AD is applied to the original finite element code for (1) as well as to the modified code to approximate (14). The values of our three metrics are listed in Table 1 for various problem sizes. We observe significant gains in terms of both FLOPs (for both modes) and space requirements. All three performance metrics are more or less independent of problem size. This can be explained by the fact that the complexity of the linear system in the finite element code is linear in the problem size, N (every nonlinear equation solved converged in five iterations). The amount of time required to define the mesh and the finite element matrix is also linear in the problem size, hence, the FLOPs gain remains essentially constant. In addition, the amount of space allocated to define the mesh is proportional to N, causing the memory savings to be independent of N.

The FLOPs in the forward mode and the FLOPs and space complexity in the reverse mode are displayed in Figures 3, 5, and 4, respectively. Note that in each case, these metrics behave linearly in N.

The sensitivity solution. AD of the modified problem computes the sensitivity solution correctly. Figure 6 displays the ℓ_2-norm of the error between the sensitivity solution computed using the methodology of section 3 and the correct analytic sensitivity. The analytic sensitivity is obtained by differentiating u_{ex} and evaluating it at the nodes of the finite element mesh.

In the next section we produce the results for the 2-D Laplace equation problem, where the gains are more dramatic, especially in terms of memory requirements.

TABLE 1
1-*D problem comparisons.*

N	FLOP1	FLOP2	% gain	SPACE1	SPACE2	% gain	RFLOP1	RFLOP2	% gain
8	18924	25245	25.04	9698	10920	11.19	45800	52744	13.17
16	37352	49865	25.09	18890	20808	9.22	84984	97184	12.55
32	74624	99585	25.07	37274	40584	8.16	164008	186720	12.16
64	149168	199025	25.05	74042	80136	7.60	322056	365792	11.96
128	298256	397905	25.04	147578	159240	7.32	638152	723936	11.85

TABLE 2
2-*D problem comparisons.*

N	FLOP1	FLOP2	% gain	SPACE1	SPACE2	% gain	RFLOP1	RFLOP2	% gain
10×10	4.06×10^4	5.44×10^4	36.9	2.49×10^3	3.09×10^4	91.9	7.19×10^4	1.18×10^5	39.0
20×20	4.15×10^5	5.30×10^5	21.7	1.00×10^4	1.45×10^5	93.1	6.65×10^5	8.85×10^5	24.9
40×40	6.01×10^6	6.52×10^6	7.7	4.01×10^5	6.29×10^6	93.6	8.93×10^6	9.89×10^6	9.7
80×80	8.97×10^7	9.18×10^7	2.3	1.60×10^6	2.61×10^7	93.9	1.32×10^8	1.37×10^8	3.6
100×100	2.18×10^8	2.21×10^8	1.4	2.50×10^6	3.63×10^7	93.8	3.22×10^8	3.28×10^8	1.8

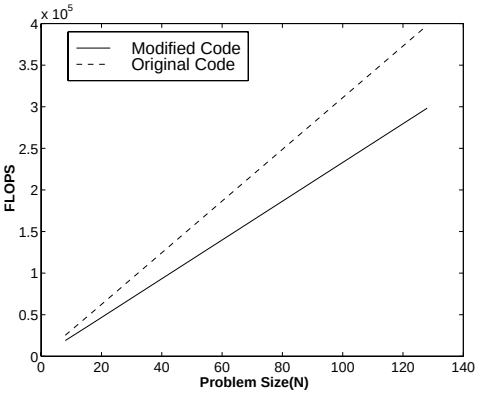


FIG. 3. *Forward mode FLOPs for the* 1*-D problem.*

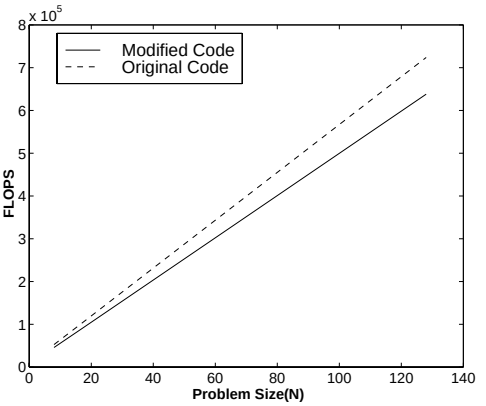


FIG. 4. *Reverse mode FLOPs for the* 1*-D problem.*

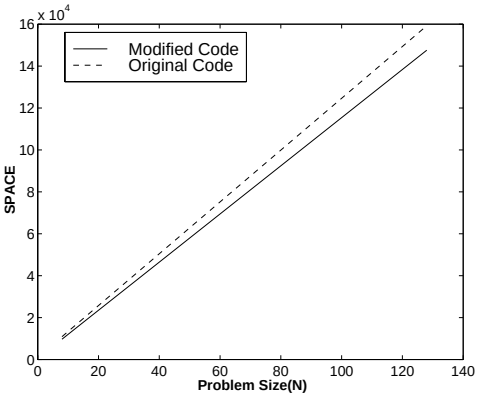


FIG. 5. *Space complexity for the* 1*-D problem.*

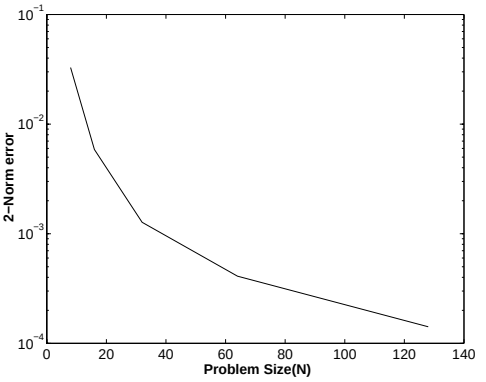


FIG. 6. *The ℓ_2-norm of the error in the sensitivity solution.*

TABLE 3
FLOPs overhead in solving the CSE.

N	Approximating the CSE	Proposed AD methodology	% overhead
10×10	3.44×10^4	4.06×10^4	18.0
20×20	4.01×10^5	4.15×10^5	3.5
40×40	5.86×10^6	6.01×10^6	2.6
80×80	8.79×10^7	8.97×10^7	2.1
100×100	2.14×10^8	2.18×10^8	1.8

4.2. Laplace equation in two dimensions. Table 2 provides computational metrics for AD of the 2-D problem introduced in sections 2.2 and 3.2. Results are provided for various problem sizes. In this case, the trend is that the improvement in the FLOPs requirement diminishes with increasing problem size. This is explained by the fact that the most expensive step in the computation is the solution of the linear system. This step dominates defining the mesh, the system matrix, etc. Therefore, the difference in the FLOPs required to compute the different right-hand sides becomes negligible.

The FLOPs requirement in the reverse mode depends on both the space complexity metric (since all stored derivatives need to be accessed again in the reverse pass) as well as the FLOPs to perform the derivative calculations themselves. Hence, in the reverse mode, the FLOPs gain is greater than that in the forward mode. However, the dependence of the space complexity metric on the FLOPs count is insignificant compared to the FLOPs required to perform the linear system solve. Figures 7 and 9 display how the FLOPs in the forward mode and reverse mode compare for the two AD methodologies. The substantial order-of-magnitude gain in space complexity can be seen in Figure 8.

The efficiency of the code generated by our AD methodology compares favorably with software written specifically to solve the CSE. We display the FLOPs requirements for solving the CSE using AD of the modified problem as well as traditional application of AD in Table 3. Note that there is very little overhead involved with this AD approach over a direct approximation. These results correspond to the same approximation of the CSE that is used for the PDE.

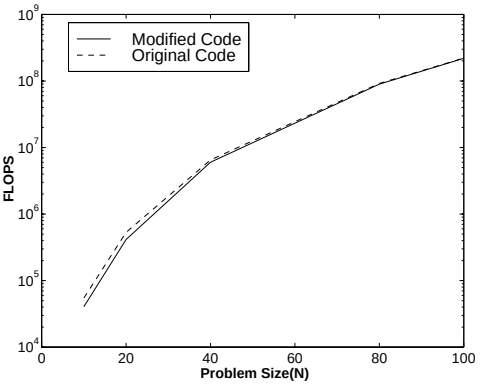


FIG. 7. *Forward mode FLOPs for the* 2*-D problem.*

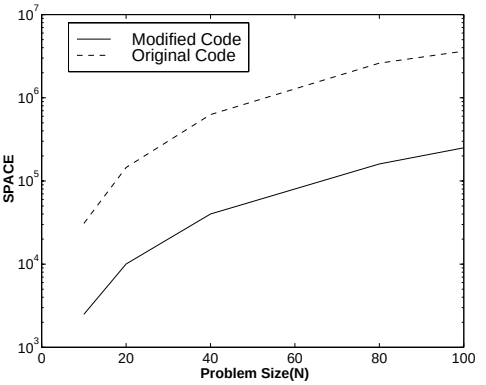


FIG. 8. *Space complexity for the* 2*-D problem.*

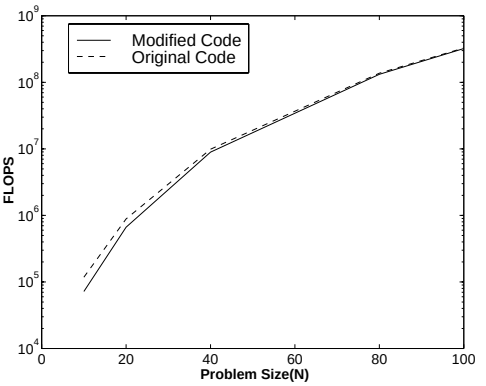


FIG. 9. *Reverse mode FLOPs for* 2*-D problem.*

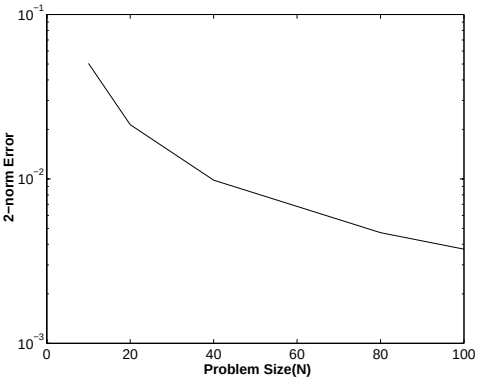


FIG. 10. *The ℓ_2-norm of the error in the sensitivity solution.*

The sensitivity solution. The ℓ_2-norm of the error in sensitivity calculations versus problem size is plotted in Figure 10. As in section 4.1, the exact solution is defined by evaluating the analytic sensitivity at the $N \times N$ finite difference nodes. Again, we see that accurate sensitivity solutions are calculated using AD of the modified problem.

4.3. Analysis of the numerical results. These results show that a sometimes dramatic, and usually substantial, gain in terms of FLOPs and memory requirements can be achieved by using the modified problem. To develop an a priori estimate of the potential gains of this AD methodology, we present the following recipe. Consider the following PDE solution code which can be broadly divided into two main parts: the code to define the mesh and the code for computing the solution. Consider a general template (here a denotes the independent variables, e.g., the shape parameters):

function $u =$ **pdesolution**(a)

Compute the Mesh, $M(a)$. (Complexity: memory $= S_m$, FLOPs $= F_m$)

Compute the Solution, $u = F(a, M(a))$. (Complexity: memory $= S_s$, FLOPs $= F_s$)

end

For generality, we assume a multiparameter case. Assume the number of parameters is given by p. In the above framework, the various complexities can now be estimated by

- FLOPs for the forward mode. For AD of the modified problem, the FLOPs will be $\mathsf{FLOP1} = F_m + (p+1)F_s$, while for the original problem it will be $\mathsf{FLOP2} = (p+1)(F_m + F_s)$. The gain is
$$\frac{\mathsf{FLOP2} - \mathsf{FLOP1}}{\mathsf{FLOP2}} \times 100\% = \frac{pF_m}{\mathsf{FLOP2}} \times 100\%.$$
If F_m is substantial, i.e., the mesh definition requires a lot of computation as in elliptic grid generation [20], then a significant gain can be expected. *In the 1-D problem, F_m is a significant portion of the computational effort as N is increased; thus the gain remains significant as the problem size increases.*
- Space complexity for the reverse mode. For the modified problem the space complexity will be $\mathsf{SPACE1} = S_m + (p+1)S_s$ while for the original problem

it will be $\mathsf{SPACE2} = (p+1)(S_m + S_s)$. The gain is

$$\frac{\mathsf{SPACE2} - \mathsf{SPACE1}}{\mathsf{SPACE2}} \times 100\% = \frac{pS_m}{\mathsf{SPACE2}} \times 100\%.$$

If the space requirement for the mesh setup, S_m, is substantial compared to the solution step, S_s, then a significant gain is expected in the space complexity. *In the 2-D problem, the memory required to store the mesh and its derivatives, S_m, remains a significant portion of the overall total memory required (since the system matrix is sparse); thus the gain remains substantial as the problem size is increased.*

- FLOPs for the reverse mode. For the modified problem, the FLOPs required for the reverse mode of AD will be $\mathsf{RFLOP1} = \mathsf{FLOP1} + \mu\mathsf{SPACE1}$ while for the original problem it will be $\mathsf{RFLOP2} = \mathsf{FLOP2} + \mu\mathsf{SPACE2}$, where μ is constant which determines the FLOPs required to save and access one unit of memory (stored for accessing in the reverse pass). The gain is

$$\frac{\mathsf{RFLOP2} - \mathsf{RFLOP1}}{\mathsf{RFLOP2}} \times 100\% = \frac{(1+\mu)\, pF_m}{\mathsf{rFLOP2}} \times 100\%.$$

5. Comments and conclusions. To summarize, recall that when a problem has a parameter-dependent domain, differentiating the discrete form of the equations (the traditional use of AD) can lead to mesh sensitivities. These terms lead to additional computations and storage in the differentiated code. This extra computation can be avoided by using AD to generate code to approximate the CSE. Minor modifications to the original software were outlined for two example problems so that AD of the modified code produced the desired result. Numerical experiments indicated that the number of FLOPs and the required memory were both reduced when this methodology was applied. In some cases, the savings were substantial (up to 25% improvement in the FLOPs requirements for a 1-D finite element problem and better than 10-fold improvement in the memory requirements for a 2-D finite difference problem). These are savings over and above the calculation of the mesh sensitivity terms themselves. In the previous section, we provided formulae that can be used to estimate the amount of savings that this AD methodology will produce.

The methodology is useful in problems where the domain is fixed, yet $\mathcal{M}_a$ is still parameter dependent (e.g., this occurs when adaptive mesh refinement strategies are used). To handle this case with AD, the mesh is simply specified as a fixed constant (it is possible to do this in most AD software). The CSE are solved on this mesh using the technique described in section 3. Since the approximation takes place after the differentiation, sensitivities are found without mesh derivatives [5]. Note that fixing the mesh is also possible in the approximate-then-differentiate approach if the domain is not parameter dependent.

This methodology has no benefit in problems where $\mathcal{M}_a = \mathcal{M}$, however, as the operations of differentiation and approximation commute in this case. For this situation, it is better to apply AD to the original code (with respect to a) than to apply the proposed AD methodology (with respect to α).

In some instances, the modification of the boundary conditions can be very difficult. Programs which are set up to handle zero Dirichlet boundary conditions may not have the capability of treating nontrivial data without some significant program modification. These cases require extending the solution vector to include the boundary points and increasing the size of the system matrix. These changes may not be

straightforward when this occurs and the benefits reported in section 4 need to be weighed against the resources required to perform the necessary changes. It's possible that the traditional application of AD is more attractive in this case.

Our methodology was presented for the situation where either the boundary conditions, forcing functions, or the shape of the domain was parameter dependent. However, it can be extended to a more general case when, for example, $\mathcal{A}_a$ has other dependencies on a, such as parameter-dependent coefficients. When this situation occurs, the CSE contains additional functions of u. These extra terms can be combined with the function $g(u)$ in problem (4). The implementation of this methodology can be more difficult in this case since we need to extract the necessary code out of the computation of $A_a(u^N)$ in order to build the modified right-hand side.

This approach is applicable to a wide variety of PDE approximation schemes. In this work, we have demonstrated the applicability of this approach for PDEs that are approximated with either finite differences or finite element methods. However, one should take the point of view that AD is being used to approximate the sensitivity equation in these problems. Therefore the success of this approach comes from the fact that the approximation scheme can be changed so that it approximates a modified PDE (and that these modifications are small, occurring in a small portion of the code). The application of AD to this modified code will yield the desired sensitivity solution.

REFERENCES

[1] C. Bischof, A. Carle, A. Griewank, and P. Hovland, *ADIFOR: Generating Derivative Codes From Fortran Programs*, Technical Report MCS-P263-0991, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL, and Center for Research on Parallel Computation, Rice University, Houston, TX, 1991.

[2] J. Borggaard and J. Burns, *Asymptotically consistent gradients in optimal design*, in Multidisciplinary Design Optimization: State of the Art, N. Alexandrov and M. Hussaini, eds., SIAM, Philadelphia, PA, 1997, pp. 303–314.

[3] J. Borggaard and J. Burns, *A PDE sensitivity equation method for optimal aerodynamic design*, J. Comput. Phys., 136 (1997), pp. 366–384.

[4] J. Borggaard, J. Burns, E. Cliff, and M. Gunzburger, *Sensitivity calculations for a 2D, inviscid, supersonic forebody problem*, in Identification and Control in Systems Governed by Partial Differential Equations, H.T. Banks, R. Fabiano, and K. Ito, eds., SIAM, Philadelphia, PA, 1993, pp. 14–24.

[5] J. Borggaard and D. Pelletier, *Computing design sensitivities using an adaptive finite element method*, in Proceedings of the 27th AIAA Fluid Dynamics Conference, New Orleans, LA, 1996.

[6] G. Bugeda and J. Oliver, *A general methodology for structural shape optimization problems using automatic adaptive remeshing*, Internat. J. Numer. Methods Engrg., 36 (1993), pp. 3161–3185.

[7] J. Burns, L. Stanley, and D. Stewart, *Computational methods for design sensitivities*, in Optimal Control: Theory, Algorithms Appl., Appl. Optim. 15, W.H. Hager and P.M. Pardalos, eds., Kluwer, Dordrecht, 1998, pp. 40–66.

[8] G.C. Buscaglia, R.A. Feijóo, and C. Padra, *A posteriori error estimation in sensitivity analysis*, Structural Optimization, 9 (1995), pp. 194–199.

[9] T. Coleman and A. Verma, *Structure and efficient Hessian calculation*, in Advances in Nonlinear Programming, Ya-Xiang Yuan, ed., Kluwer, Dordrecht, The Netherlands, 1996, pp. 57–72.

[10] T. Coleman and A. Verma, *Structure and efficient Jacobian calculation*, in Computational Differentiation: Techniques, Applications, and Tools, M. Berz, C. Bishof, G. Corliss, and A. Griewank, eds., SIAM, Philadelphia, PA, 1996, pp. 149–159.

[11] T.F. Coleman, F. Santosa, and A. Verma, *Semi-automatic differentiation*, in Computational Methods for Optimal Design and Control, J. Borggaard, J. Burns, E. Cliff, and S. Schreck, eds., Birkhäuser, Boston, 1998, pp. 113–126.

[12] A. Verma, *ADMAT: Automatic differentiation in MATLAB using object oriented methods*, in Proceedings of the 1998 SIAM Workshop on Object Oriented Methods for Interoperable Scientific and Engineering Computing, M.E. Henderson, C.R. Anderson, and S.L. Lyons, eds., SIAM, Philadelphia, 1999, pp. 174–183.

[13] T.F. Coleman and A. Verma, *ADMIT*-1: *Automatic differentiation and MATLAB interface toolbox*, ACM Trans. Math. Software, to appear.

[14] P. Eberhard and C. Bischof, *Automatic Differentiation of Numerical Integration Algorithms*, Technical Report MCS-P621-1196, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL, 1996.

[15] A. Griewank, *Achieving logarithmic growth of temporal and spatial complexity in reverse automatic differentiation*, Optim. Methods Softw., 1 (1992), pp. 35–54.

[16] A. Griewank, *Some bounds on the complexity of gradients, Jacobians, and Hessians*, in Complexity in Nonlinear Optimization, P. Pardalos, ed., World Scientific Publishers, River Edge, NJ, 1993, pp. 131–167.

[17] A. Griewank, D. Juedes, and J. Utke, *ADOL-C, a package for the automatic differentiation of algorithms written in C/C++*, ACM Trans. Math. Software, 2 (1996), pp. 131–167.

[18] J.M. Ortega and W.C. Rheinboldt, *On discretization and differentiation of operators with application to Newton's method*, SIAM J. Numer. Anal., 3 (1966), pp. 143–156.

[19] D. Stewart, *Numerical Methods for Accurate Computation of Design Sensitivities*, Ph.D. thesis, Virginia Tech, Blacksburg, VA, 1998.

[20] J.F Thompson, Z.U.A. Warsi, and C.W. Mastin, *Numerical Grid Generation: Foundations and Applications*, North-Holland Publishing Company, New York, 1985.

[21] A. Verma, *Structured Automatic Differentiation*, Ph.D. thesis, Cornell University, Ithaca, NY, 1998.

SIAM J. SCI. COMPUT.
Vol. 22, No. 1, pp. 63–80
© 2000 Society for Industrial and Applied Mathematics

MESH PARTITIONING: A MULTILEVEL BALANCING AND REFINEMENT ALGORITHM*

C. WALSHAW† AND M. CROSS†

Abstract. Multilevel algorithms are a successful class of optimization techniques which addresses the mesh partitioning problem. They usually combine a graph contraction algorithm together with a local optimization method which refines the partition at each graph level. In this paper we present an enhancement of the technique which uses imbalance to achieve higher quality partitions. We also present a formulation of the Kernighan–Lin partition optimization algorithm which incorporates load-balancing. The resulting algorithm is tested against a different but related state-of-the-art partitioner and shown to provide improved results.

Key words. graph-partitioning, mesh partitioning, load-balancing, multilevel algorithms

AMS subject classifications. 05C85, 65Y05

PII. S1064827598337373

1. Introduction.

The need for mesh partitioning arises naturally in many finite element (FE) and finite volume (FV) applications. Meshes composed of elements such as triangles or tetrahedra are often better suited than regularly structured grids for representing completely general geometries and resolving wide variations in behavior via variable mesh densities. Meanwhile, the modeling of complex behavior patterns means that the problems are often too large to fit onto serial computers, either because of memory limitations or computational demands, or both. Distributing the mesh across a parallel computer so that the computational load is evenly balanced and the data locality maximized is known as mesh partitioning. It is well known that this problem is NP-complete [7], so in recent years much attention has been focused on developing suitable heuristics, and some powerful methods, many based on a graph corresponding to the communication requirements of the mesh, have been devised, e.g., [5].

A particularly popular and successful class of algorithms which addresses this mesh partitioning problem is known as multilevel algorithms. They usually combine a graph contraction algorithm which creates a series of progressively smaller and coarser graphs together with a local optimization method which, starting with the coarsest graph, refines the partition at each graph level. In this paper we present an enhancement of the technique which uses imbalance to achieve higher quality partitions. We also present a formulation of the Kernighan–Lin (KL) partition optimization algorithm which incorporates load-balancing.

We focus here on serial partitioning. Although emphasis in the field is switching to parallel partitioning methods, we aim here to address one of the fundamental mechanisms of the multilevel paradigm and thus a serial implementation provides a clear and relatively parameter-free environment for establishing how imbalance can affect the overall performance of the strategy. However, elsewhere we have provided a different but related parallel formulation [21], and indeed the algorithms described here are used directly as part of that parallel strategy.

*Received by the editors April 15, 1998; accepted for publication (in revised form) December 10, 1999; published electronically June 13, 2000.

http://www.siam.org/journals/sisc/22-1/33737.html

†Computing and Mathematical Sciences, University of Greenwich, Park Row, Greenwich, London, SE10 9LS, UK (C.Walshaw@gre.ac.uk), (m.cross@gre.ac.uk).

1.1. Overview. Below, in section 1.2, we introduce the mesh partitioning problem and establish some terminology. In section 2 we then describe the multilevel paradigm and present a new enhancement in the idea of a multilevel balancing schedule. Next, in section 3, we describe a KL-type optimization algorithm which both balances a partition of the graph to within some given tolerance and also refines it. In section 4 we present results from the multilevel balancing and refinement algorithm, comparing it with a similar formulation which only incorporates multilevel refinement. We also compare different multilevel balancing schedules. Finally in section 5 we draw some conclusions and present some ideas for further investigation.

The principal innovations described in this paper are twofold:

- In section 2.2 we formalize the idea of combining multilevel refinement with a multilevel balancing schedule.
- In section 3.4 we describe a new formulation of a KL-type partitioning algorithm (incorporating hill-climbing), which both balances and refines.

Two implementation ideas are also described:

- In section 3.3 we describe a ranking for prioritizing vertices for migration which incorporates their weight as well as their gain.
- In section 3.5 we describe how we deal with vertices which are neighbors to more than one subdomain.

1.2. Notation and definitions. To define the mesh partitioning problem, let $G = G(V, E)$ be an undirected graph of vertices V, with edges E which represent the data dependencies in the mesh. We assume that the graph is connected. (Although if this is not the case we have, elsewhere, discussed an algorithm for connecting the components together [25].) We also assume that both vertices and edges are weighted (with positive integer values) and that $|v|$ denotes the weight of a vertex v and similarly for edges and sets of vertices and edges. The vertex weight is used to approximate processor loading whilst the edge weights model communication costs (although see section 3.1). Typically both vertex and edge weights are given a unit cost although there has been some recent work on accurate cost modeling [15], and for some real applications the processor load can depend on many other factors such as data access patterns. However, since this is a function of the final partition it is not possible to estimate such costs a priori, and we do not address this issue here.

Given that the mesh needs to be distributed to P processors, define a partition π to be a mapping of V into P disjoint subdomains S_p such that $\bigcup_P S_p = V$. The partition π induces a *subdomain graph* on G which we shall refer to as $G_\pi = G_\pi(S, C)$; there is an edge $(S_p, S_q) \in C$ if there are vertices $v_1, v_2 \in V$ with $(v_1, v_2) \in E$, and $v_1 \in S_p$, $v_2 \in S_q$, and the weight of a subdomain is just the sum of the weights of the vertices in the subdomain, $|S_p| = \sum_{v \in S_p} |v|$. We denote the set of intersubdomain or cut edges (i.e., edges cut by the partition) by E_c (note that $|E_c| = |C|$). Vertices which have an edge in E_c (i.e., $\{v \in V :$ there exists $v' \in V$, with $(v, v') \in E_c\}$) are referred to as *border* vertices.

The definition of the graph-partitioning problem is to find a partition which evenly balances the load (i.e., vertex weight) in each subdomain whilst minimizing the communications cost. To balance the load, the optimal subdomain weight is given by $\overline{S} := \lceil |V|/P \rceil$ (where the ceiling function $\lceil x \rceil$ returns the smallest integer $\geq x$) and the *imbalance* is then defined as the maximum subdomain weight divided by the optimal (since the computational speed of the underlying application is determined by the most heavily weighted processor). As is usual, throughout this paper the communications cost will be estimated by $|E_c|$, the weight of cut edges or cut-weight,

although see section 3.1 for further discussion on this point. A more precise definition of the graph-partitioning problem is therefore to find π such that $|S_p| \leq \overline{S}$ and such that $|E_c|$ is minimized. Note that perfect balance is not always possible for graphs with nonunitary vertex weights.

Throughout this paper we use some fairly specific terminology and in particular we shall refer to *refinement* as the improvement of partition quality (i.e., the cut-weight) without regard to load-balance; *balancing* will then refer to the improvement of imbalance and *optimization* refers to refinement and balancing. We also make the distinction between those algorithms, such as that of Kernighan and Lin [14], which refine a *bisection* and algorithms which refine a partition of P subdomains. Such algorithms have been known as k-way (e.g., [13]) or multiway (e.g., [24]) algorithms, but here we shall simply refer to them as *partition* (as opposed to bisection) refinement algorithms. Finally we shall use the words processor and subdomain interchangeably; the mesh is partitioned into P subdomains each of which will be mapped onto one processor.

2. The multilevel paradigm. In recent years it has been recognized that an effective way of both accelerating partition refinement and, perhaps more importantly, giving it a global perspective is to use multilevel techniques. The idea is to group vertices together to form *clusters*, use the clusters to define a new graph, and recursively iterate this procedure to create a series of increasingly coarse graphs until the size of the coarsest graph falls below some threshold. A fast and possibly crude initial partition of the coarsest graph is calculated and then successively interpolated onto and optimized on each of the graphs in reverse order. This sequence of contraction followed by repeated expansion/refinement loops is known as the multilevel paradigm and has been successfully developed as a strategy for overcoming the localized nature of the KL [14] (and other) algorithms. The multilevel idea was first proposed by Barnard and Simon [1] as a method of accelerating spectral bisection and improved by both Hendrickson and Leland [9] and Bui and Jones [2], who generalized it to encompass local refinement algorithms.

2.1. Implementation. To create a coarser graph $G_{l+1}(V_{l+1}, E_{l+1})$ from $G_l(V_l, E_l)$ we use a variant of the graph contraction algorithm proposed by Hendrickson and Leland [9]. The idea is to find a maximal independent subset of graph edges or *matching* of graph vertices and then collapse them. The set is independent if no two edges in the set are incident on the same vertex (so no two edges in the set are adjacent), and maximal if no more edges can be added to the set without breaking the independence criterion. Having found such a set, each selected edge is collapsed and the vertices, $u_1, u_2 \in V_l$ say, at either end of it are merged to form a new vertex $v \in V_{l+1}$ with weight $|v| = |u_1| + |u_2|$. Edges which have not been collapsed are inherited by the child graph, G_{l+1}, and, where they become duplicated, are merged with their weight summed. This occurs if, for example, the edges (u_1, u_3) and (u_2, u_3) exist when edge (u_1, u_2) is collapsed. Because of the inheritance properties of this algorithm, it is easy to see that the total graph weight remains the same, $|V_{l+1}| = |V_l|$, and the total edge weight is reduced by an amount equal to the weight of the collapsed edges.

A simple way to construct a maximal independent subset of edges is to visit the vertices of the graph in a random order and pair up or match unmatched vertices with an unmatched neighbor. It has been shown [12] that it can be beneficial to the optimization to collapse the most heavily weighted edges and our matching algorithm uses this heuristic.

The initial partition. Having constructed the series of graphs until the number of vertices in the coarsest graph is smaller than some threshold, the normal practice of the multilevel strategy is to carry out an initial partition. Here, following the idea of Gupta [8] we contract until the number of vertices in the coarsest graph is the same as the number of subdomains, P, and then simply assign vertex i to subdomain S_i. Unlike Gupta, however, we do not carry out repeated expansion/contraction cycles of the coarsest graphs to find a well balanced initial partition but instead, since our optimization algorithm incorporates balancing, we commence on the expansion/optimization sequence immediately.

Note that contraction down to P vertices should always be possible provided the graph is connected (assumed, section 1.2). To see this consider that every connected graph of V vertices must have at least $V-1$ edges and that the collapsing of an edge results in a connected graph. Thus, if $V > P$ there must be at least one edge which can be collapsed to create a graph with $V-1$ vertices and so on by induction.

Note also that for certain graphs the resulting initial partition can be extremely imbalanced as a result of the weights becoming extremely inhomogeneous in the coarsest graphs. This suggests that perhaps, for such examples, contraction down to P vertices does not enhance the final partition. However, because the final contractions are relatively very cheap and this imbalance does not seem to affect the final partition quality we retain this feature in order not to introduce another parameter (i.e., a contraction threshold) to the method.

Partition expansion. Having optimized the partition on a graph G_l, the partition must be interpolated onto its parent G_{l-1}. The interpolation itself is a trivial matter; if a vertex $v \in V_l$ is in subdomain S_p then the matched pair of vertices that it represents, $v_1, v_2 \in V_{l-1}$, will be in S_p.

2.2. Multilevel balancing schedule. It has been noted previously (originally in [18] and subsequently in [13, 23]) that allowing a small amount of imbalance often leads to a higher partition quality. We also observe that one of the most attractive features of the multilevel paradigm is the way in which the partition quality (usually the number of cut edges) is refined *gradually* as the expansion proceeds; i.e., after each refinement the partition quality of a given graph G_l is usually better than that of G_{l+1} (because there are more degrees of freedom). In this paper we combine both observations (imbalance can lead to higher partition quality and gradual refinement of quality being an attractive feature) by allowing a variable amount of *imbalance* which is reduced gradually as the expansion proceeds. The idea is that by allowing a large imbalance in the coarsest graphs a better partition may be found than if balance was rigidly enforced, but that this imbalance will not cause degradation in the final partition of the finest graph if removed gradually throughout the expansion procedure. Note particularly the second statement—if the finest graph starts the refinement with a high quality but poorly balanced partition, then much of the quality may be destroyed by balancing. (See the end of this section for an example of this behavior.)

In fact it is often not possible to achieve perfect balance in the coarsest graphs because the vertices may be heavily weighted and very inhomogeneous (e.g., if balance requires moving a weight of 10 from one subdomain to another but all vertices are of weight 20 or over, perfect balance cannot be attained). Hence it could be argued that all multilevel algorithms employ this idea of multilevel balancing. Indeed, our previous work in this area, e.g., [24], employs a diffusive load-balancer *at every* refinement level and so the idea has been implicit in our work for sometime. In this paper, however,

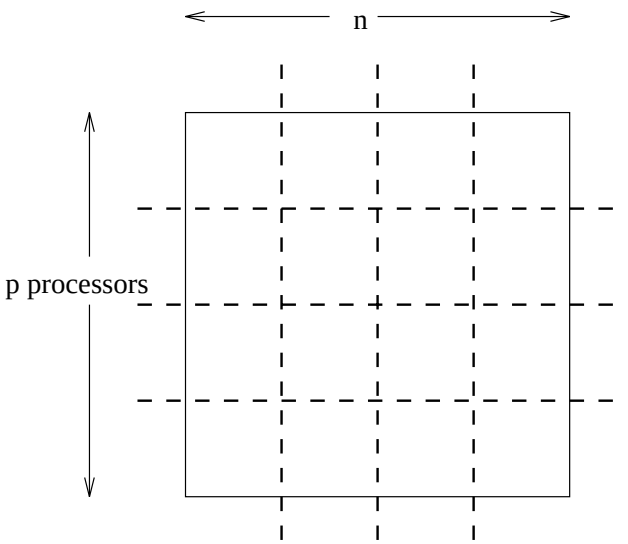


FIG. 1. *A regular domain of $N = n^2$ vertices perfectly partitioned into $P = p^2$ subdomains.*

we formalize the idea of balancing and refinement at every level and also describe an optimization algorithm which both achieves a given level of imbalance (if possible) and refines for quality. Note that we make the distinction between this work and that in the multilevel diffusion algorithm of Schloegel, Karypis, and Kumar [16], where a diffusive load-balancer is employed at each coarse level until balance is attained and thereafter partition quality refinement without active balancing is employed.

In order to talk about improving the balance gradually from one graph level to another, for each graph, G_l, let T_l be the target subdomain weight. If every subdomain, S_p, is not heavier than this target (i.e., $\max |S_p| \leq T_l$), then we say that the graph is balanced and the optimization can concentrate on refinement alone (so long as the balance is not destroyed). However, if $\max |S_p| > T_l$, then the optimization must concentrate on balancing (with some regard to refinement). Clearly this series $\{T_l\}$ is an arbitrary heuristic, but it must be determined with two caveats:

- If it ascends too rapidly, the balance inherited by G_l from G_{l+1} may cause the partition quality to be lost in trying to attain T_l. (See the end of this section for an example of this behavior.)
- If it ascends too slowly, the benefits for the partition quality of having a high imbalance tolerance may never be seen.

Some results with different functions for T_l are given in section 4.2, but with the above in mind we derive T_l as follows:

Let $G(V, E)$ be regular graph with N $(= n \times n)$ vertices perfectly partitioned into P $(= p \times p)$ subdomains as in Figure 1. The *maximum* border length of a subdomain is then given by

$$4\left(\frac{N}{P}\right)^{\frac{1}{2}}.$$

The average weight of a vertex is $|V|/N$, and so we can estimate the weight of border

vertices in the subdomain as

$$4\left(\frac{N}{P}\right)^{\frac{1}{2}} \times \frac{|V|}{N} = \frac{4|V|}{(PN)^{\frac{1}{2}}}.$$

Recall that for each graph G_l we wish to define a target subdomain weight T_l which will not cause too much degradation in partition quality when balancing its parent G_{l-1} down to its target T_{l-1}. After some experimentation, we have chosen to allow an excess weight in any given subdomain of approximately half one border layer of a subdomain in the parent graph. The target weight is given by the optimal subdomain weight plus the excess weight and so, using the regular two-dimensional (2D) model, we set T_l to be

$$T_l = \lceil |V|/P \rceil + \frac{1}{2} \times \frac{4|V|}{(PN_{l-1})^{\frac{1}{2}}}.$$

If we define the imbalance tolerance, θ_l, to be the maximum allowable subdomain weight expressed as a proportion of the optimal subdomain weight, then

$$\theta_l = \frac{\lceil |V|/P \rceil + 2|V|(PN_{l-1})^{-\frac{1}{2}}}{\lceil |V|/P \rceil} \approx 1 + 2\left(\frac{P}{N_{l-1}}\right)^{\frac{1}{2}}.$$

In other words a graph G_l is considered balanced if the imbalance is less than $\theta_l = 1 + 2(\frac{P}{N_{l-1}})^{\frac{1}{2}}$ for $l > 0$. For the final (and original) graph, G_0, which has no parent, we can either set $\theta_0 = 1$ to aim for perfect balancing or, as is often the case (e.g., [13]), allow a slight imbalance. For the results in this paper we have chosen to set $\theta_0 = 1.03$ and then we set $\theta_l = \max(\theta_0, 1 + 2(\frac{P}{N_{l-1}})^{\frac{1}{2}})$ for $l > 0$. Note that we have chosen a 2D model of the regular partition; a three-dimensional (3D) model using the same arguments gives $\theta_l = 1 + 3(\frac{P}{N_{l-1}})^{\frac{1}{3}}$, and results using this model can be found in section 4.2.

Figure 2 shows an example of some typical behavior for the balancing schedule derived above and the algorithm described in section 3. The dotted line plots the target weight or balancing schedule, and each step down represents the transition from one graph level G_l to its parent G_{l-1}. Notice that at the start of the iterations the tolerance is around 2.3—i.e., the graph is considered balanced if every subdomain is smaller than 2.3 times the optimal weight. The solid line represents the attained balance—this is below the target level most of the time and, by iteration 30, it tracks the target weight exactly, showing that the optimization algorithm in section 3 is very good at taking advantage of any leeway in the imbalance tolerance. (The final imbalance tolerance for the method is set at $\theta_0 = 1.03$ which is why the balance never reaches 1.0.) Finally the dashed line shows the evolution of the cut edges (scaled by a large factor to fit onto the graph). The peaks early on in the iterations correspond to balances which exceed the tolerance and as mentioned above this causes serious degradation in the partition quality as the algorithm balances the graph. However, after about iteration 30 the cut-weight decreases monotonically.

3. The balancing and refinement optimization algorithm. In this section we describe an optimization algorithm which combines load-balancing and partition quality refinement. It is a KL-type algorithm incorporating a hill-climbing mechanism to enable it to escape from local minima; in other words vertex migration from subdomain to subdomain can be *accepted* even if it degrades the partition quality and later,

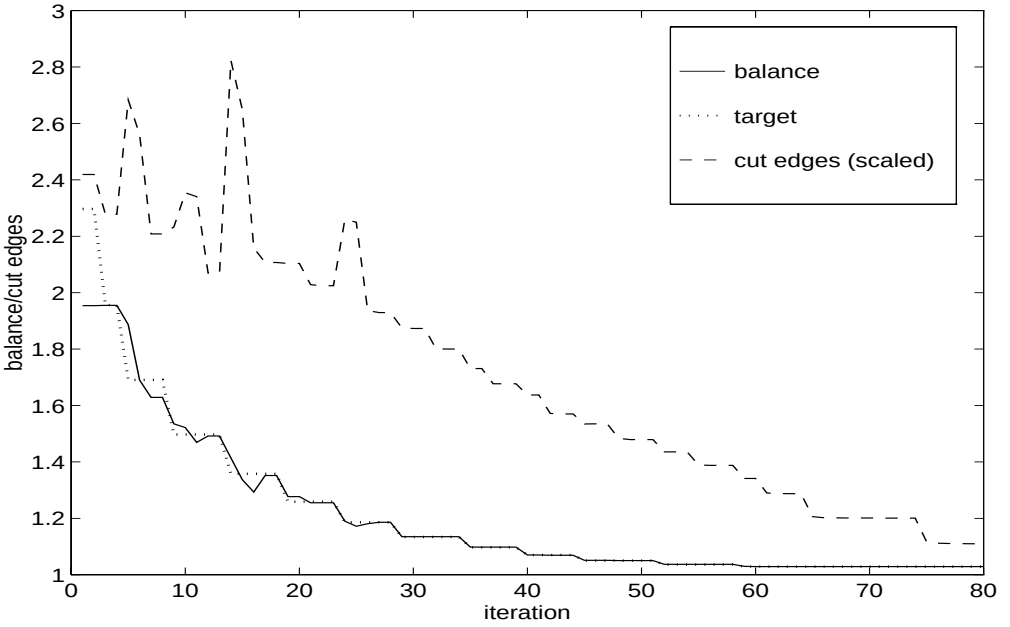


FIG. 2. *An example of the evolution of balance and cut-weight for the multilevel balancing and refinement algorithm.*

based on the subsequent evolution of the partition, either rejected or *confirmed*. The algorithm uses bucket sorting (see section 3.3), the linear time complexity improvement of Fiduccia and Mattheyses [6], and is a partition optimization formulation; in other words it optimizes a partition of P subdomains rather than a bisection. In this respect it most closely resembles the algorithm of Karypis and Kumar [13], but additionally incorporates load-balancing (using the diffusive algorithm of Hu and Blake [11]; see section 3.2. The algorithm described here is strictly serial in nature, but a parallel formulation (in which essentially each intersubdomain interface is treated as a separate problem) can be found in [21].

3.1. The gain function. A key concept in the method is the idea of *gain*. Loosely, the gain $g(v, q)$ of a vertex v in subdomain S_p can be calculated for every other subdomain, S_q, $q \neq p$, and it expresses some "estimate" of how much the partition would be "improved" were v to migrate to S_q. The gain is usually directly related to some cost function which measures the quality of the partition and which we aim to minimize. Typically the cost function used is simply the total weight of cut edges, $|E_c|$, and then the gain expresses the change in $|E_c|$. More recently, there has been some debate about the most important quantity to minimize, and in [20] Vanderstraeten and Keunings demonstrate that it can be extremely effective to vary the cost function based on a knowledge of the solver. Whichever cost function is chosen, however, the idea of gains is generic. For the purposes of this paper we shall assume that the gain $g(v, q)$ just expresses the reduction in the cut-weight, $|E_c|$.

3.2. Load-balancing: Calculating the flow. Given a graph partitioned into unequal sized subdomains, we need some mechanism for distributing the load equally. To do this we solve the load-balancing problem on the subdomain graph, G_π, in order to determine a *balancing flow*, a flow along the edges of G_π which balances the weight

of the subdomains. By keeping the flow localized in this way, vertices are not migrated between nonadjacent subdomains, and hence (hopefully) the partition quality is not degraded (since a vertex migrating to a subdomain to which it is not adjacent is almost certain to have a negative gain).

This load-balancing problem, i.e., how to distribute N tasks over a network of P processors so that none have more than $\lceil N/P \rceil$, is a very important area for research in its own right with a vast range of applications. The topic is introduced in [17] and some common strategies are described. Much work has been carried out on parallel or distributed algorithms and, in particular, on diffusive algorithms [3]; here we use an elegant diffusive variant developed by Hu and Blake [11] with fast convergence. This method was derived to minimize the Euclidean norm of the transferred weight, although it has recently been shown that all diffusion methods minimize this quantity [4, 10]. The algorithm simply involves solving the system $L\mathbf{x} = \mathbf{b}$, where L is the Laplacian of the subdomain graph:

$$L_{pq} = \begin{cases} \text{degree}(S_p) & \text{if } p = q, \\ -1 & \text{if } p \neq q \text{ and } S_p \text{ is adjacent to } S_q, \\ 0 & \text{otherwise,} \end{cases}$$

and where $b_p = |S_p| - \overline{S}$, the weight of S_p less the optimal weight. The weight to be transferred across edge (S_p, S_q) is then given by $x_p - x_q$. Note that this method is closely related to the standard diffusion algorithm [3], except that the diffusion coefficients are not fixed but are determined at each iteration by a conjugate gradient search.

This algorithm (or, in principle, any other distributed load-balancing algorithm) is used to determine *how much* weight to transfer across edges of the subdomain graph and the optimization technique below is then used to decide *which* vertices to move. The algorithm is employed as suggested in [11], solving iteratively with a conjugate gradient solver; it is solved for a real solution and the (integer) flow is determined by rounding. Note, however, that the Laplacian of any undirected graph contains a zero eigenvalue with the corresponding eigenvector $[1, 1, \ldots, 1]$ and the solution iterates are orthogonalized against this [11]. If any other singularities are detected (e.g., if the graph is disconnected) the software will switch to another method, an intuitive and entirely localized distributed load-balancing algorithm due to Song [19].

Occasionally whilst optimization is taking place vertex migration can cause the subdomain graph to change (e.g., two nonadjacent subdomains may become adjacent). If an edge disappears over which flow is scheduled to move the subdomain graph must be rebalanced, although we speed this process up by adding the extraneous flow back into its source subdomain and rebalancing the graph from that point. We also limit the number of possible rebalances on any graph since otherwise the system can exhibit cyclic behavior.

3.3. Bucket sorting. The bucket sort is an essential tool for the efficient and rapid sorting and adjustment of vertices by their gain. The concept was first suggested by Fiduccia and Mattheyses in [6], and the idea is that all vertices of a given gain g are placed together in a "bucket" which is ranked g. Finding a vertex with maximum gain then simply consists of finding the (nonempty) bucket with the highest rank and picking a vertex from it. If the vertex is subsequently migrated from one subdomain to another then its gain and the gains of all its neighbors have to be adjusted and resorted by gain. Using a bucket sort for this operation simply requires recalculating the gains of the vertex and its neighbors and transferring them to the appropriate

buckets, an essentially localized operation. If a bucket sort were not used and, say, the vertices were simply stored in a list in gain order, then the entire list would require resorting (or at least merge-sorting with the sorted list of adjusted vertices) an essentially $O(N)$ operation for every migration.

In our implementation each bucket is (as usual) represented by a double linked list of vertices (since vertices must be extracted from the list without having to search through it). However, we additionally prefer to sort the vertices by gain and then by weight. The reasoning behind this is simple: if, for example, a transfer of weight 3 between 2 subdomains is required, then it is preferable to pick 3 vertices each of gain 1 and weight 1 rather than 1 vertex of gain 1 and weight 3. Conversely, if a transfer of weight 2 is required, then it is better to move 1 vertex of weight 2 and gain -1, rather than 2 vertices of weight 1 and gain -1. Thus we order the vertices primarily by gain and then by weight, lightest first for positive gains and heaviest first for negative gains. Rather than sorting the contents of each bucket we simply provide a different bucket for each gain/weight combination, and so if W represents the weight of the largest vertex in a given graph $g(v)$ the gain of a vertex v, we rank v with the formula:

$$\text{rank}(v) = \begin{cases} g(v) \times W + W - |v| & \text{if } g(v) > 0, \\ g(v) \times W + |v| - 1 & \text{otherwise,} \end{cases}$$

which provides the desired ordering. The ranking is unique for each combination of $g(v)$ and $|v|$ because $1 \le |v| \le W$ for all vertices v (it is assumed that $|v| > 0$), and so

$$\text{rank}(v) \le g(v) \times W + W - 1 < g(v) \times W + W = [g(v) + 1] \times W.$$

Hence

$$g(v) \times W \le \text{rank}(v) < [g(v) + 1] \times W.$$

In other words, for a given vertex v with gain $g(v)$, the upper bound on $\text{rank}(v)$ is strictly less than the lower bound on $\text{rank}(w)$ for any vertex w with gain $g(w) = [g(v) + 1]$.

Note that in the very coarse graphs at the top of the multilevel process, it is possible or even common to produce graphs with a wide range of vertex weights and potential gains. For this reason, rather than maintaining a sparse but potentially huge array of pointers to buckets, we store the nonempty buckets in a binary tree adding and deleting buckets as required (see Figure 3). This tree structure may still be large but cannot exceed the number of border vertices in the graph in size. In the sections below the term bucket tree will be used to refer to the binary tree of buckets.

3.4. The iterative optimization algorithm. The serial optimization algorithm, as is typical for KL-type algorithms, has inner and outer iterative loops with the outer loop terminating when no migration takes place during an inner loop. The algorithm is shown in pseudocode form in Figure 4. The optimization uses two bucket trees (see section 3.3), and is initialized by calculating the gain for all border vertices and inserting them into one of the bucket trees. These vertices will subsequently be referred to as *candidate* vertices and the tree containing them as the *candidate tree*. The idea of only inserting the border vertices into the bucket tree (rather than all vertices) was first described in [23] and has subsequently been described as lazy initialization [9].

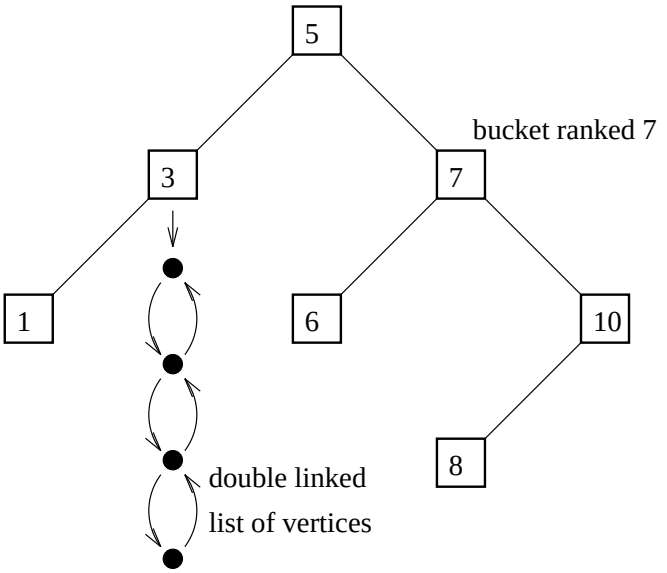


FIG. 3. *A bucket tree.*

The inner loop proceeds by examining candidate vertices highest gain first (by always picking vertices from the highest ranked bucket), testing whether the vertex is acceptable for migration, and then transferring it to the other bucket tree (the tree of *examined* vertices). This inner loop terminates when the candidate tree is empty, although it may terminate early if the partition cost (i.e., the number of cut edges) rises too far above the cost of the best partition found so far. This type of early termination is typical of KL-type algorithms; without it, the entire graph may be searched with diminishing prospect of finding a better solution along the search path. Once the inner loop has terminated any vertices remaining in the candidate tree are transferred to the examined tree and finally pointers to the two trees are swapped ready for the next pass through the inner loop.

Migration acceptance. Let T refer to the target weight for the graph (section 2.2) and W represent the weight of the largest subdomain, $W = \max_P |S_p|$. If the required flow from subdomain S_p to subdomain S_q is F_{pq}, a candidate vertex v with weight $|v|$ (≥ 0) is accepted for migration from S_p to S_q (with weights $|S_p|$ and $|S_q|$) if

$$\begin{array}{lll} & \text{(a) } W > T \quad \text{and} \quad 2F_{pq} > |v| \\ \text{or} & \text{(b) } W \leq T \quad \text{and} \quad |S_q| + |v| \leq T. \end{array} \tag{3.1}$$

These criteria reflect the aim of trying to balance the graph down to the target weight, T, and then keeping it there. If the graph is not yet within the imbalance tolerance (i.e., $W > T$), then (3.1a) only allows migration which reduces the required flow. Condition (3.1b) guarantees that once balance is achieved the graph cannot become unbalanced again.

Note that in order to satisfy the flow entirely we would only move a vertex if from S_p to S_q if the flow, F_{pq}, was greater than or equal to the vertex's weight (i.e., $F_{pq} \geq |v|$). The wider acceptance condition (3.1a), $2F_{pq} > |v|$, however, also allows moves where $|v|$ exceeds F_{pq} but which reduces the total required flow in the system.

```
while (optimizing) { /* outer loop */
    optimizing = 0
    best cost = cost
    while (vertices in candidate tree) { /* inner loop */
        vertex = best candidate
        if (migration acceptable for vertex) {
            optimizing = 1
            migrate vertex {
                adjust flow and subdomain weights
                adjust gains of neighboring vertices
                    and transfer to appropriate buckets
            }
            adjust gain of vertex and transfer to examined tree
            if (better partition) { /* confirm migration */
                best_cost = cost
                reset recent move list
            } else {
                append vertex to recent move list
                if (cost - best_cost > limit) /* early termination */
                    break
            } /* hill-climbing mechanism */
        }
        transfer vertex to examined tree
    } /* inner loop */
    for (vertices in recent move list) /* hill-climbing mechanism */
        migrate vertex back to previous partition
    for (vertices in candidate tree)
        transfer vertex to examined tree
    swap pointers to candidate and examined trees
} /* outer loop */
```

FIG. 4. *The Kernighan–Lin partition optimization algorithm.*

For example, if $|v| = 5$ and $F_{pq} = 3$, the migration would not be acceptable under the condition $F_{pq} \geq |v|$, but using (3.1a) the move is acceptable and F_{pq} changes to -2 (alternatively, $F_{qp} = 2$) after migration, which is a reduction in the total.

When a vertex is accepted for migration, its subdomain is changed and the subdomain weights and flow are adjusted. The gains are recalculated for the vertex and all of its neighbors and they are transferred to the appropriately ranked buckets. Note that examined vertices are transferred between buckets in the examined bucket tree and candidate vertices are transferred between buckets in the candidate bucket tree. Neighboring vertices which were not in the border at this point but which become border vertices as a result of the migration are put into the candidate tree. In this way it is actually possible for a vertex to be migrated more than once during the course of an inner loop (if it is moved out of and back into the border region by migration it becomes a candidate vertex at each stage), but accepted vertices which have not yet been confirmed (see below) cannot be transferred to the candidate tree as this can lead to infinitely cyclic behavior.

Migration confirmation and hill-climbing. The algorithm uses a KL-type

hill-climbing strategy. As can be seen from (3.1) migrations can be *accepted* even if they increase the partition cost (i.e., have negative gain). During each pass through the inner loop, a record of the optimal partition achieved by migration within that loop is maintained together with a list of vertices which have migrated since that value was attained. If subsequent migration finds a "better" partition, then the migration is *confirmed* and the list is reset. Once the inner loop is terminated, any vertices remaining in the list (vertices whose migration has not been confirmed) are migrated back to the subdomains they came from when the optimal cost was attained.

To define a "better" partition, let $\overline{\pi}$ represent the optimal partition reached so far and π^i the subsequent partition after some migration (i.e., after some iterations of the inner loop). Each partition has a cost associated with it, $C(\pi)$ (in this case just the total weight of cut edges), and an imbalance which depends on $W(\pi)$, the weight of the largest subdomain in that partition. Again let T represent the target weight for the graph (see section 2.2). Denoting $C(\pi^i)$ and $W(\pi^i)$ by C^i and W^i (and similarly for $\overline{\pi}$), then π^i is confirmed as a new optimal partition if:

$$\begin{array}{lll} & \text{(a) } C^i < \overline{C}, & \\ \text{or} & \text{(b) } C^i = \overline{C} & \text{and } W^i < \overline{W}, \\ \text{or} & \text{(c)} & T \leq W^i < \overline{W}. \end{array} \tag{3.2}$$

Condition (3.2c) simply states that, while the graph is unbalanced (i.e., $W^i > T$), any partition which improves the balance is confirmed. Conditions (3.2a) and (3.2b) are more typical of KL-type algorithms and confirm any partition which either improves on the optimal cost (3.2a) or improves on the optimal balance without raising the cost (3.2b). Note that whilst the partition is unbalanced, the conditions in (3.2) do not actually define an ordering of partitions (i.e., if π_1 and π_2 are partitions with $C_1 < C_2$ and $T < W_2 < W_1$, then either π_1 or π_2 would be confirmed in preference to the other). However, because of condition (3.1a), the behavior of the unbalanced system is monotonic in the sense that only changes that reduce $\Sigma_p \Sigma_q F_{pq}$ are accepted.

3.5. Vertices adjacent to several subdomains. In general, for graphs arising from FE/FV meshes with coarse granularity partitions (i.e., $V \gg P$), most border vertices will only be adjacent to vertices in one other subdomain. However, those vertices that are adjacent to several subdomains are treated slightly differently in that, if a tested migration is not acceptable, they are replaced in the *candidate* tree at the level of their next highest gain. They are not transferred to the examined tree until either being successfully migrated or all possible migrations have been tested. This is best illustrated with an example: suppose a vertex is adjacent to four subdomains, S_p, S_q, S_r, and S_s, and suppose that the respective gains are $g_p > g_q = g_r > g_s$. The vertex is initially placed in the candidate tree and ranked g_p. When subsequently tested, if migration is not acceptable using the criteria in 3.1, the vertex is replaced in the candidate tree and ranked g_q $(= g_r)$. When the vertex next comes up for testing, migration to S_p, S_q, and S_r is assessed (note that a move to S_p may now be acceptable due to the intervening migration), and if none are acceptable the vertex is replaced in the candidate tree with a rank g_s. When the vertex is again tested, migration to S_p, S_q, S_r, and S_s is tested, and if none are acceptable the vertex is transferred to the examined tree ranked g_p. Of course, it might be considered unnecessary to retest moves which have already been tested (i.e., those with gains greater than the migration under consideration), but since the edge weights to all adjacent subdomains must be calculated to determine the next highest gain, there is no great extra expense involved in doing so.

TABLE 4.1
A summary of the test meshes.

	Size		Degree			
Mesh	V	E	max	min	avg	Type
crack	10,240	30,380	9	3	5.93	2D nodal graph
4elt	15,606	45,878	10	3	5.88	2D nodal graph
t60k	60,005	89,440	3	2	2.98	2D dual graph
dime20	224,843	336,024	3	2	2.99	2D dual graph
144	144,649	1,074,393	26	4	14.86	3D nodal graph
m14b	214,765	1,679,018	40	4	15.64	3D nodal graph
fe-ocean	143,437	409,593	6	1	5.71	3D dual graph
mesh1m	1,119,663	2,212,012	4	2	3.95	3D dual graph
vibrobox	12,328	165,250	120	8	26.81	vibroacoustic matrix
memplus	17,758	54,196	573	1	6.10	digital memory circuit
oliker	50,000	95,800	4	1	3.83	3D weighted dual graph
bmw1c	100,917	208,165	38	0	4.13	3D weighted nodal graph

4. Results. We have implemented the algorithms described here within the framework of JOSTLE, a mesh partitioning software tool developed at the University of Greenwich and freely available for academic and research purposes under a licensing agreement[1]. The experiments were carried out on a Sun SPARC 20 with a 50 MHz CPU and 320 Mbytes of memory. The test graphs have been chosen to be a representative sample of medium to large scale real-life problems and include both 2D and 3D examples of nodal graphs (where the mesh nodes are partitioned) and dual graphs (where the mesh elements are partitioned). We have also included two non mesh-based graphs (vibrobox and memplus) and two weighted graphs (oliker and bmw1c). Such weighted graphs are not widely available since most applications do not accurately instrument costs and it is difficult to know whether such graphs are representative; however, they do seem to perform in a broadly similar fashion to those with unit weights.

Table 4.1 gives a list of the graphs, their sizes, the maximum, minimum, and average degree of the vertices, and a short description. The degree information (the degree of a vertex is the number of vertices adjacent to it) gives some idea of the character of the graphs. These range from the relatively homogeneous dual graphs, where every vertex represents a mesh element, in these cases a triangle (2D), tetrahedron (3D), or brick (3D), and so every vertex has at most 3, 4, or 6 neighbors, respectively, to the non mesh-based graphs memplus which has vertices of degree 573 and vibrobox, where the average degree is 26.8. Most of the graphs are not weighted and so the number of vertices in V is the same as the total vertex weight $|V|$, and similarly for the edges E. However we use two weighted graphs; oliker (with weighted vertices only and $|V| = 111,690$) is the root mesh for an hierarchical adaptively refined mesh and the weights represent the number of leaf elements that each root element has been refined into, whilst bmw1c (with $|V| = 1,073,726,486$ and $|E| = 3,396,572$) arises from an attempt to correctly instrument and model costs [15].

The results of using the multilevel balancing and refinement algorithm are shown in Table 4.2 for 4 values of P (the number of processors/subdomains). The table shows the total weight of cut edges, $|E_c|$, and the run time in seconds, t_s. The algorithm is allowed a final imbalance tolerance of $\theta_0 = 1.03$ (although this may be reset at runtime). In the following sections we compare the results with different balancing schedules and with a similar multilevel mesh partitioner which does not use

[1]Available from http://www.gre.ac.uk/jostle.

TABLE 4.2
The results of the multilevel balancing and refinement algorithm showing the cut-weight $|E_c|$ and CPU time in seconds t_s.

	$P = 16$		$P = 32$		$P = 64$		$P = 128$	
Mesh	$\|E_c\|$	t_s	$\|E_c\|$	t_s	$\|E_c\|$	t_s	$\|E_c\|$	t_s
crack	1,191	0.95	1,804	1.16	2,733	1.37	3,960	3.01
4elt	1,012	1.05	1,687	1.22	2,772	2.31	4,285	3.19
t60k	984	2.75	1,588	3.27	2,445	4.33	3,631	6.27
dime20	1,274	10.01	2,282	10.37	3,617	12.34	5,517	16.60
144	41,842	22.20	60,467	26.61	83,640	37.60	113,045	55.23
m14b	45,988	29.22	72,997	37.91	105,799	49.46	147,840	77.58
fe-ocean	8,879	13.01	14,302	18.58	23,098	25.73	32,248	36.18
mesh1m	24,522	87.22	35,178	100.31	51,580	117.63	72,834	154.45
vibrobox	34,521	8.43	45,374	12.11	54,439	20.60	62,436	25.97
memplus	13,958	17.33	16,125	25.34	18,616	41.20	22,466	87.22
oliker	2,129	3.12	3,864	4.61	5,449	6.76	7,591	13.14
bmw1c	39,825	5.91	59,665	10.48	91,704	11.86	132,135	17.00

a multilevel balancing schedule. In these sections the $|E_c|$ results in Table 4.2 are also referred to as $|E_c(J)|$ and $|E_c(T_2)|$.

4.1. Comparison results. To demonstrate the quality of the partitions, we have compared the results in Table 4.2 with those produced by METIS, another state-of-the-art partitioning package [13]. The version we have used, kmetis 2.0.6, provides multilevel coarsening with a heavy edge heuristic and we have used the option of a KL partition refinement algorithm. (The default is a greedy partition refinement algorithm which is slightly faster but provides slightly lower quality partitions.) The primary distinctions between the two partitioners, aside from implementation details, is that METIS coarsens to 2,000 vertices and then carries out a balanced initial partition, whilst JOSTLE coarsens to P vertices (one per subdomain) and then uses the multilevel balancing schedule described in section 2.2 and the balancing refinement algorithm described in section 3.4. A more recent version of METIS is now available but only allows greedy refinement.

Table 4.3 shows a comparison of the cut-weight $|E_c|$. For each value of P, the first column shows the value of $|E_c|$ for METIS, $|E_c(M)|$, while the second column shows the ratio of $|E_c|$ for METIS over that for JOSTLE, $|E_c(M)|/|E_c(J)|$. As can be seen, with 4 exceptions (mesh1m, $P = 16$; oliker, $P = 32$, and $P = 128$; vibrobox, $P = 64$), the results for METIS are always worse and can be 17% larger (4elt, $P = 16$). The average difference in the quality ranges between 4% and 9% over the different values of P and as an overall average METIS produces partitions which are 5.9% worse than JOSTLE. Note that for the 2 weighted graphs METIS failed to partition bmw1c for any value of P (apparently becoming stuck in an infinite loop—we have removed this result from the averaging) and ran very slowly for oliker. Although this does not demonstrate a dramatic improvement for our algorithm, it does indicate a consistent improvement on results perceived as state-of-the-art.

We also tried using METIS in a similar manner to JOSTLE, coarsening down to P vertices. Although not as recommended by the code authors, this gave very similar results to Table 4.3 with an overall average partition quality 6.2% worse than JOSTLE.

It is not the primary aim of this paper to compare run times for the algorithms, but Table 4.4 shows a similar comparison of t_s. Unfortunately the results are somewhat distorted by idiosyncrasies of the partitioners. METIS performed particularly badly

TABLE 4.3
A comparison of cut edge results for METIS, $|E_c(M)|$, and JOSTLE, $|E_c(J)|$.

Mesh	$P = 16$		$P = 32$		$P = 64$		$P = 128$	
	$\lvert E_c(M)\rvert$	$\frac{\lvert E_c(M)\rvert}{\lvert E_c(J)\rvert}$	$\lvert E_c(M)\rvert$	$\frac{\lvert E_c(M)\rvert}{\lvert E_c(J)\rvert}$	$\lvert E_c(M)\rvert$	$\frac{\lvert E_c(M)\rvert}{\lvert E_c(J)\rvert}$	$\lvert E_c(M)\rvert$	$\frac{\lvert E_c(M)\rvert}{\lvert E_c(J)\rvert}$
crack	1,319	1.11	2,006	1.11	2,888	1.06	4,253	1.07
4elt	1,188	1.17	1,831	1.09	2,942	1.06	4,637	1.08
t60k	1,031	1.05	1,648	1.04	2,614	1.07	3,702	1.02
dime20	1,339	1.05	2,328	1.02	3,742	1.03	5,711	1.04
144	42,219	1.01	62,482	1.03	87,045	1.04	118,079	1.04
m14b	49,744	1.08	75,743	1.04	108,221	1.02	153,154	1.04
fe-ocean	10,108	1.14	16,215	1.13	24,786	1.07	34,974	1.08
mesh1m	24,000	0.98	36,706	1.04	54,015	1.05	75,463	1.04
vibrobox	37,391	1.08	45,850	1.01	53,888	0.99	62,836	1.01
memplus	16,785	1.20	19,527	1.21	19,858	1.07	23,844	1.06
oliker	2,270	1.07	3,788	0.98	5,604	1.03	7,534	0.99
bmw1c	–		–		–		–	
Average		1.09		1.06		1.04		1.04

TABLE 4.4
A comparison of timings for METIS, $t_s(M)$, and JOSTLE, $t_s(J)$.

Mesh	$P = 16$		$P = 32$		$P = 64$		$P = 128$	
	$t_s(M)$	$\frac{t_s(M)}{t_s(J)}$	$t_s(M)$	$\frac{t_s(M)}{t_s(J)}$	$t_s(M)$	$\frac{t_s(M)}{t_s(J)}$	$t_s(M)$	$\frac{t_s(M)}{t_s(J)}$
crack	1.02	1.07	1.05	0.91	1.34	0.98	1.88	0.62
4elt	1.01	0.96	1.32	1.08	1.59	0.69	2.88	0.90
t60k	2.53	0.92	2.68	0.82	2.97	0.69	4.54	0.72
dime20	14.85	1.48	15.08	1.45	15.69	1.27	16.83	1.01
144	20.74	0.93	22.98	0.86	24.44	0.65	27.01	0.49
m14b	31.75	1.09	33.87	0.89	37.16	0.75	40.73	0.53
fe-ocean	10.55	0.81	12.05	0.65	13.39	0.52	15.68	0.43
mesh1m	146.45	1.68	162.34	1.62	175.00	1.49	163.24	1.06
vibrobox	3.82	0.45	4.18	0.35	5.09	0.25	7.26	0.28
memplus	2.51	0.14	3.69	0.15	4.74	0.12	9.08	0.10
oliker	71.08	22.78	71.11	15.43	71.73	10.61	72.43	5.51
bmw1c	–		–		–		–	
Average		2.94		2.20		1.64		1.06

for the weighted graphs, failing completely on bmw1c and running relatively very slowly for oliker (up to 22 times slower, $P = 16$), which heavily influences the averages. In contrast, JOSTLE runs relatively slowly for the non mesh-based graphs (up to 10 times slower for memplus, $P = 128$); this is because, as we discuss below (section 4.2), the vertex weight inhomogeneity in the coarse graphs means that the load-balancer is called with unnecessary frequency.

However, if we neglect the weighted and non mesh-based graphs, this table highlights a difference in implementations more than anything. For both codes the operation count is approximately linearly dependent on the number of border vertices (which increase with P), however, JOSTLE loops over border vertices while METIS loops over all vertices in the graph. This means that for coarse granularities (larger meshes or smaller values of P), where a relatively small number of vertices are in the subdomain borders, JOSTLE is faster since it visits a much smaller proportion of the data. However, for finer granularities (smaller meshes or larger values of P) METIS, by accessing the data contiguously, gains from a relatively good cache hit rate, while JOSTLE, which is essentially accessing the data at random, starts to lose out. These differences can be quite marked; for $P = 16$ on a large graph, JOSTLE is up to 1.68

TABLE 4.5
A comparison of cut edge results for a constant balancing schedule, $|E_c(T_c)|$, and the 2D schedule, $|E_c(T_2)|$.

Mesh	$P = 16$		$P = 32$		$P = 64$		$P = 128$	
	$\|E_c(T_c)\|$	$\frac{\|E_c(T_c)\|}{\|E_c(T_2)\|}$	$\|E_c(T_c)\|$	$\frac{\|E_c(T_c)\|}{\|E_c(T_2)\|}$	$\|E_c(T_c)\|$	$\frac{\|E_c(T_c)\|}{\|E_c(T_2)\|}$	$\|E_c(T_c)\|$	$\frac{\|E_c(T_c)\|}{\|E_c(T_2)\|}$
crack	1,255	1.05	1,828	1.01	2,766	1.01	4,068	1.03
4elt	1,136	1.12	1,771	1.05	2,889	1.04	4,401	1.03
t60k	976	0.99	1,613	1.02	2,567	1.05	3,776	1.04
dime20	1,490	1.17	2,458	1.08	4,039	1.12	5,825	1.06
144	44,543	1.06	62,602	1.04	86,171	1.03	117,082	1.04
m14b	51,318	1.12	82,156	1.13	110,179	1.04	152,379	1.03
fe-ocean	9,029	1.02	15,704	1.10	23,825	1.03	33,078	1.03
mesh1m	26,997	1.10	39,375	1.12	54,491	1.06	74,941	1.03
vibrobox	38,024	1.10	45,085	0.99	52,047	0.96	61,294	0.98
memplus	14,713	1.05	16,344	1.01	17,797	0.96	21,512	0.96
oliker	2,217	1.04	3,997	1.03	5,664	1.04	7,877	1.04
bmw1c	40,932	1.03	64,462	1.08	98,158	1.07	141,521	1.07
Average		1.07		1.05		1.03		1.03

times faster (mesh1m), while for $P = 128$, METIS can take about half the time (144).

4.2. Different balancing schedules. Constant schedule. The balancing schedule derived in section 2.2 is essentially an arbitrary heuristic and in this section we test some different schedules. First, Table 4.5 shows a comparison of the cut-weight for a constant schedule, $|E_c(T_c)|$, where θ_l is set to 1.03 for every graph G_l. For each value of P, the second column compares the results from this fixed schedule with the results in Table 4.2 using the 2D schedule and referred to as $|E_c(T_2)|$. As can be seen the constant schedule provides partition qualities which with 6 exceptions are always worse; the average difference in the quality ranges between 3% and 7% over the different values of P and can be as bad as 17%. This constant schedule strategy is similar to that used by METIS (which also has an imbalance tolerance of 1.03), where balance is established early on in the expansion/refinement process (in the case of METIS, during the initial partitioning) and maintained thereafter and indeed the average difference in the quality is about the same for the METIS results (Table 4.3).

Interestingly, five out of the six results for which the constant schedule is better than the 2D schedule are for the non mesh-based graphs. Our algorithms have all been tuned for performance with meshes (since that is where our own requirement for mesh partitioning lies), and so perhaps this is unsurprising. However a detailed study of the results suggests perhaps a more simple explanation. The non mesh-based graphs are very inhomogeneous, certainly in terms of vertex degree (see Table 4.1), and as a result the final coarsest graphs (indeed most of the coarse graphs) have very inhomogeneous vertex weights. This in turn means that most partitions of the coarser graphs are unbalanced (by the definition in section 2.2), and so the balancing schedule actually has very little effect since the optimization behaves in the same way while the graph is unbalanced, whether the imbalance is close to the threshold (as it might be for the 2D schedule) or far away (as in the constant schedule).

3D schedule. Table 4.6 shows a comparison of the cut-weight for the 3D schedule, $|E_c(T_3)|$, mentioned in section 2.2, and where the imbalance tolerance for graph G_l is set to $\theta_l = 1+3(\frac{P}{N_{l-1}})^{\frac{1}{3}}$. Again for each value of P, the second column compares the results from this fixed schedule with the results in Table 4.2 using the 2D schedule, $|E_c(T_2)|$. Overall both sets of results are very similar although on the average the 3D

TABLE 4.6
A comparison of cut edge results for a 3D schedule, $|E_c(T_3)|$, and the 2D schedule, $|E_c(T_2)|$.

Mesh	$P = 16$		$P = 32$		$P = 64$		$P = 128$	
	$\lvert E_c(T_3)\rvert$	$\frac{\lvert E_c(T_3)\rvert}{\lvert E_c(T_2)\rvert}$	$\lvert E_c(T_3)\rvert$	$\frac{\lvert E_c(T_3)\rvert}{\lvert E_c(T_2)\rvert}$	$\lvert E_c(T_3)\rvert$	$\frac{\lvert E_c(T_3)\rvert}{\lvert E_c(T_2)\rvert}$	$\lvert E_c(T_3)\rvert$	$\frac{\lvert E_c(T_3)\rvert}{\lvert E_c(T_2)\rvert}$
crack	1,206	1.01	1,826	1.01	2,714	0.99	4,019	1.01
4elt	993	0.98	1,643	0.97	2,781	1.00	4,339	1.01
t60k	967	0.98	1,588	1.00	2,418	0.99	3,581	0.99
dime20	1,291	1.01	2,323	1.02	3,569	0.99	5,418	0.98
144	44,455	1.06	62,367	1.03	82,825	0.99	114,046	1.01
m14b	48,893	1.06	71,349	0.98	105,759	1.00	147,756	1.00
fe-ocean	8,764	0.99	14,676	1.03	23,140	1.00	31,717	0.98
mesh1m	23,663	0.96	36,366	1.03	52,086	1.01	73,078	1.00
vibrobox	36,092	1.05	46,288	1.02	53,341	0.98	61,830	0.99
memplus	14,142	1.01	16,052	1.00	18,370	0.99	22,512	1.00
oliker	2,157	1.01	3,755	0.97	5,483	1.01	7,737	1.02
bmw1c	38,127	0.96	60,646	1.02	89,647	0.98	132,869	1.01
Average		1.01		1.01		0.99		1.00

results are marginally worse; the average difference in the quality ranges between 1% better and 1% worse with an overall average of just 0.22% deterioration. One might suspect that the 3D meshes would fare better with a 3D schedule than the 2D ones, but this does not seem to be borne out. In fact we have experimented with a number of different formulations and found the algorithm relatively insensitive to the schedule provided that the initial imbalance tolerance is sufficiently high.

4.3. Summary. Overall, we conclude from these results, together with the comparisons with METIS, and other experiments not presented here, that the use of a multilevel balancing schedule can improve the partition quality. The improvement is not enormous because the multilevel paradigm with a static schedule already provides excellent results (and hence the margin for improvement is small). However, it does exist and provides on average a 5–6% decrease in the cut-weight.

Note also that differences in quality tend to diminish as P increases. It is tempting to speculate that this is because the margins for difference decrease as the number of vertices per subdomain ($\approx V/P$) decreases. Indeed in the limit where $V = P$ the only balanced partition (for an unweighted graph at least) is to put one vertex in each subdomain and so the differences vanish altogether.

5. Conclusions and future directions. We have presented an enhancement to the multilevel paradigm where the freedom allowed by a balancing schedule is used to find higher quality partitions. We have also presented a formulation of a KL-type partition optimization algorithm which incorporates a diffusive balancing flow. The resultant algorithm has been shown to provide higher quality partitions than a state-of-the-art partitioner and, depending on granularity, is often faster.

The algorithms are fairly simple to describe and relatively parameter-free and as a result provide an ideal setting for testing new ideas before implementing them within the framework of a fully parallel mesh partitioner. Recently we have provided further results using the algorithms to address more complex partitioning problems such as balancing multiple computational phases [25], and to minimize alternative objective functions such as subdomain aspect ratio [22]. We also hope to use them in the near future to optimize mapping onto parallel machine topologies (rather than just cut-weight).

REFERENCES

[1] S. T. Barnard and H. D. Simon, *A fast multilevel implementation of recursive spectral bisection for partitioning unstructured problems*, Concurrency: Practice and Experience, 6 (1994), pp. 101–117.

[2] T. N. Bui and C. Jones, *A heuristic for reducing fill-in in sparse matrix factorization*, in Proceedings of the Sixth SIAM Conference on Parallel Processing for Scientific Computing, Vol. 1, R. F. Sincovec et al., eds., SIAM, Philadelphia, 1993, pp. 445–452.

[3] G. Cybenko, *Dynamic load balancing for distributed memory multiprocessors*, Journal of Parallel and Distributed Computing, 7 (1989), pp. 279–301.

[4] R. Diekmann, A. Frommer, and B. Monien, *Efficient schemes for nearest neighbor load balancing*, Parallel Comput., 25 (1999), pp. 789–812.

[5] C. Farhat, H. D. Simon, and Lanteri, *TOP/DOMDEC–A software tool for mesh partitioning and parallel processing*, Computing Systems Engrg., 6 (1995), pp. 13–26.

[6] C. M. Fiduccia and R. M. Mattheyses, *A linear time heuristic for improving network partitions*, in Proceedings of the 19th IEEE Design Automation Conference, Las Vegas, NV, IEEE, Piscataway, NJ, 1982, pp. 175–181.

[7] M. Garey, D. Johnson, and L. Stockmeyer, *Some simplified NP-complete graph problems*, Theoret. Comput. Sci., 1 (1976), pp. 237–267.

[8] A. Gupta, *Fast and effective algorithms for graph partitioning and sparse matrix reordering*, IBM J. Research and Development, 41 (1996), pp. 171–183.

[9] B. Hendrickson and R. Leland, *A multilevel algorithm for partitioning graphs*, in Proceedings Supercomputing '95, San Diego, CA, S. Karin, ed., ACM Press, New York, NY, 1995.

[10] Y. F. Hu and R. J. Blake, *The optimal property of polynomial based diffusion-like algorithms in dynamic load balancing*, in Computational Dynamics '98, K. D. Papailiou et al., ed., Wiley, New York, 1998, pp. 177–183.

[11] Y. F. Hu, R. J. Blake, and D. R. Emerson, *An optimal migration algorithm for dynamic load balancing*, Concurrency: Practice and Experience, 10 (1998), pp. 467–483.

[12] G. Karypis and V. Kumar, *A fast and high quality multilevel scheme for partitioning irregular graphs*, SIAM J. Sci. Comput., 20 (1998), pp. 359–392.

[13] G. Karypis and V. Kumar, *Multilevel k-way partitioning scheme for irregular graphs*, Journal of Parallel and Distributed Computing, 48 (1998), pp. 96–129.

[14] B. W. Kernighan and S. Lin, *An efficient heuristic for partitioning graphs*, Bell Systems Tech. J., 49 (1970), pp. 291–308.

[15] B. Maerten, D. Roose, A. Basermann, J. Fingberg, and G. Lonsdale, *DRAMA: A library for parallel dynamic load balancing of finite element applications*, in Proceedings of the Ninth SIAM Conference on Parallel Processing for Scientific Computing, San Antonio, TX, 1999, CD-ROM, SIAM, Philadelphia, 1999.

[16] K. Schloegel, G. Karypis, and V. Kumar, *Multilevel diffusion schemes for repartitioning of adaptive meshes*, Journal of Parallel and Distributed Computing, 47 (1997), pp. 109–124.

[17] N. G. Shivaratri, P. Krueger, and M. Singhal, *Load distributing for locally distributed systems*, IEEE Comput., 25 (1992), pp. 33–44.

[18] H. D. Simon and S.-H. Teng, *How good is recursive bisection?*, SIAM J. Sci. Comput., 18 (1997), pp. 1436–1445.

[19] J. Song, *A partially asynchronous and iterative algorithm for distributed load balancing*, Parallel Comput., 20 (1994), pp. 853–868.

[20] D. Vanderstraeten and R. Keunings, *Optimized partitioning of unstructured computational grids*, Internat. J. Numer. Methods Engrg., 38 (1995), pp. 433–450.

[21] C. Walshaw and M. Cross, *Parallel optimization algorithms for multilevel mesh partitioning*, Parallel Comput., to appear.

[22] C. Walshaw, M. Cross, R. Diekmann, and F. Schlimbach, *Multilevel mesh partitioning for optimising domain shape*, Int. J. High Performance Comput. Appl., 13 (1999), pp. 334–353.

[23] C. Walshaw, M. Cross, and M. Everett, *A Localised algorithm for optimising unstructured mesh partitions*, Int. J. Supercomputer Appl., 9 (1995), pp. 280–295.

[24] C. Walshaw, M. Cross, and M. Everett, *Parallel dynamic graph partitioning for adaptive unstructured meshes*, Journal of Parallel and Distributed Computing, 47 (1997), pp. 102–108.

[25] C. Walshaw, M. Cross, and K. McManus, *Multiphase mesh partitioning*, Appl. Math. Model., submitted.

SIAM J. SCI. COMPUT.
Vol. 22, No. 1, pp. 81–94
© 2000 Society for Industrial and Applied Mathematics

COMPUTING CONNECTING ORBITS VIA AN IMPROVED ALGORITHM FOR CONTINUING INVARIANT SUBSPACES*

J. W. DEMMEL†, L. DIECI‡, AND M. J. FRIEDMAN§

Abstract. A successive continuation method for locating connecting orbits in parametrized systems of autonomous ODEs was considered in [*Numer. Algorithms*, 14 (1997), pp. 103–124]. In this paper we present an improved algorithm for locating and continuing connecting orbits, which includes a new algorithm for the continuation of invariant subspaces. The latter algorithm is of independent interest and can be used in contexts different than the present one.

Key words. connecting orbits, invariant subspaces, bifurcations, numerical continuation

AMS subject classifications. 34B15, 65F, 65L10

PII. S1064827598344868

1. Introduction.

Homoclinic and *heteroclinic orbits*, also called *connecting orbits*, are trajectories connecting equilibrium points of a system of autonomous ODEs. Computation of connecting orbits is becoming increasingly important, both in dynamical systems research, such as understanding chaotic dynamics and in a variety of applied problems, including wave propagation in combustion models, chemical reactions, neuronal interactions, solitary waves in fluid, solitons in nonlinear optical fiber, and communication processes in living cells, to name a few. The corresponding numerical problem is that of finding solutions $(u(t), \lambda)$ of the system of autonomous ODEs

$$u'(t) - f(u(t), \lambda) = 0, \quad u(\cdot), f(\cdot, \cdot) \in \mathbb{R}^n, \lambda \in \mathbb{R}^{n_\lambda}, \tag{1.1}$$

$$\lim_{t \to -\infty} u(t) = u_0, \quad \lim_{t \to +\infty} u(t) = u_1. \tag{1.2}$$

Most algorithms for the numerical analysis of connecting orbits reduce (1.1), (1.2) to a *boundary value problem* on a finite interval using linear or higher-order approximations of stable and unstable manifolds near u_0 and u_1, respectively. See recent papers by Champneys, Kuznetsov, and Sandstede [4], Doedel, Friedman, and Kunin [10], and Moore [14] for the history of the question and the bibliography. Note that in the last work an alternative approach was used based on the arclength parametrization instead of using time t as a parameter.

The algorithms in [4] use a version of Beyn's continuation algorithm based on projection boundary conditions (see [1], [2]). They were implemented in a set of routines, `HomCont`, which are currently part of `AUTO97` [8]. `HomCont` has capabilities for detailed bifurcation analysis of homoclinic orbits and some bifurcations of heteroclinic orbits. It has limited capabilities for locating connecting orbits, namely, a simplified version of the algorithm in [10].

*Received by the editors September 21, 1998; accepted for publication (in revised form) April 6, 1999; published electronically June 13, 2000.

http://www.siam.org/journals/sisc/22-1/34486.html

†Computer Science Division, University of California, Berkeley, CA 94720-1776 (demmel@cs.berkeley.edu).

‡School of Mathematics, Georgia Institute of Technology, Atlanta, GA 30332-0160 (dieci@math.gatech.edu). This author was supported in part under NSF DMS-9625813.

§Department of Mathematical Sciences, University of Alabama in Huntsville, Huntsville, AL 35899 (friedman@math.uah.edu). This author was supported in part by NSF DMS-9404912 and by GM 29123.

The algorithms in [10] have their primary focus on locating connecting orbits and use a modification of a continuation algorithm based on projection boundary conditions (see Friedman and Doedel [12]). They were implemented in an experimental code based on AUTO94 [9].

In order to have a well-posed problem, it is necessary for the boundary conditions to be sufficiently smooth with respect to parameters. In both [4] and [10], the boundary conditions are defined with respect to bases of stable or unstable eigenspaces of $f_u(u_0, \lambda)$ and $f_u(u_1, \lambda)$. The approach in [4] is to compute an orthonormal basis in the appropriate eigenspace, *at each pseudoarclength continuation step*, using a "black box" routine based on the real Schur factorization and then to adapt this basis to be smooth with respect to parameters, using a technique due to Beyn [2, Appendix C] which amounts to the solution of a linear system of the dimension of the eigenspace in question. The approach in [10] is to compute initially an orthonormal basis in the appropriate eigenspace via the real Schur factorization and then to continue the real Schur factorization equations (as a part of boundary conditions). At the same time precise convergence of the algorithm in [10] is not clear, and it is somewhat cumbersome to use. In some recent work Dieci and Eirola [6] provide a general differential equation framework for continuation of the block Schur factorization as well as other matrix factorizations. Reference [6] includes a comprehensive set of references for smooth matrix factorization for parameter dependent matrices.

In this paper we present an improved algorithm for locating and continuing connecting orbits, which includes a new algorithm for the continuation of invariant subspaces (CIS). This CIS algorithm is based on iterative refinement techniques originally due to Stewart [15] and later revisited by Demmel [5]. We provide some new twists to these techniques: (i) we justify these iterative refinement techniques using the differential equations which model continuation of block Schur forms, and (ii) we make use of these differential equations to obtain an accurate approximation of the relevant invariant subspace.

In the end, the new algorithm is more efficient than the algorithms in [4] and [10] and is very robust. In particular, it provides several possible safeguards against fast variation of eigenvalues. It has been implemented in an experimental code based on AUTO97, which is essentially a modification of the HomCont part of AUTO97, to include the algorithm in [10] for locating and continuing connecting orbits and the CIS algorithm, while preserving the bifurcation analysis part of HomCont.

2. An improved algorithm for locating and continuing connecting orbits. Assume, for simplicity of notation, that the fixed points u_0 and u_1 are hyperbolic, and the eigenvalues of $f_u(u_0, \lambda)$ and $f_u(u_1, \lambda)$, respectively, satisfy

$$\begin{aligned} &\mathrm{Re}\mu_{0,n} \le \cdots \le \mathrm{Re}\mu_{0,n_0+1} < 0 < \mu_{0,1} < \mathrm{Re}\mu_{0,2} \le \cdots \le \mathrm{Re}\mu_{0,n_0}, \\ &\mathrm{Re}\mu_{1,1} \le \cdots \le \mathrm{Re}\mu_{1,n_1} < 0 < \mathrm{Re}\mu_{1,n_1+1} \le \cdots \le \mathrm{Re}\mu_{1,n}. \end{aligned}$$

In this paper, we will assume that the matrices $f_u(u_{0,1}, \lambda)$ are smooth functions of λ (for λ in an appropriate subset of $\mathbb{R}^{n_\lambda}$).

The method extends to the case $\mu_{0,1} = 0$, as in [4]. It also extends to the cases of complex and multiple $\mu_{0,1}$ by a simple modification of Step 0, (2.9) below, of the algorithm (see [10, Section 4.3] for a computational example). The algorithm requires evaluation of various projections associated with the eigenspaces of $f_u(u_0, \lambda)$ and $f_u(u_1, \lambda)$. Initially these projections are constructed using the real Schur factorizations [13]

$$f_u(u_0,\lambda) = Q_0 R_0 Q_0^T, \quad f_u(u_1,\lambda) = Q_1 R_1 Q_1^T.$$

The first factorization has been chosen so that the first n_0 columns $q_{0,1},\dots,q_{0,n_0}$ of Q_0 form an orthonormal basis of the right invariant subspace S_0 of $f_u(u_0,\lambda)$, corresponding to $\mu_{0,1},\dots,\mu_{0,n_0}$, and the remaining $n-n_0$ columns $q_{0,n_0+1},\dots,q_{0,n}$ of Q_0 form an orthonormal basis of the orthogonal complement $S_0^\perp$. Similarly, the first n_1 columns $q_{1,1},\dots,q_{1,n}$ of Q_1 form an orthonormal basis of the right invariant subspace S_1 of $f_u(u_1,\lambda)$, corresponding to $\mu_{1,1},\dots,\mu_{1,n_1}$, and the remaining $n-n_1$ columns $q_{1,n_1+1},\dots,q_{1,n}$ of Q_1 form an orthonormal basis of the orthogonal complement $S_1^\perp$. In the algorithm below the matrices $Q_0(\lambda)$ and $Q_1(\lambda)$ are assumed to be computed at each continuation step by a "black box" routine, described in section 3, which ensures their continuity.

The approximate finite interval problem is to find a branch of solutions $(u(t),\lambda,u_0,u_1,T)$, $u\in C^1([0,1],\mathbb{R}^n)$, $\lambda\in\mathbb{R}^{n_\lambda}$, provided $n_\lambda = n-(n_0+n_1)+2$; $n_\lambda\ge 0$, $u_0,u_1\in\mathbb{R}^n$, $T>0$, is the length of the time interval, for some small $\epsilon_0,\epsilon_1>0$, of the time-scaled differential equation

$$u'(t) - Tf(u(t),\lambda) = 0, \quad 0<t<1, \tag{2.1}$$

subject to left boundary conditions

$$(u(0)-u_0)\cdot q_{0,n_0+i}(u_0,\lambda) = 0, \quad i=1,\dots,n-n_0, \tag{2.2}$$

$$|u(0)-u_0| = \epsilon_0, \tag{2.3}$$

right boundary conditions

$$(u(1)-u_1)\cdot q_{1,n_1+i}(u_1,\lambda) = 0, \quad i=1,\dots,n-n_1, \tag{2.4}$$

$$|u(1)-u_1| = \epsilon_1, \tag{2.5}$$

and stationary state conditions

$$f(u_0,\lambda) = 0, \tag{2.6}$$

$$f(u_1,\lambda) = 0. \tag{2.7}$$

Remark 1. Initially, we integrate in time to obtain a typically crude orbit with initial point $u(0)\in S_0$, but the terminal point $u(1)\notin S_1$, in general. Hence, τ_i defined by

$$\tau_i = (u(1)-u_1)\cdot q_{1,n_1+i}(u_1,\lambda)/\epsilon_1, \quad i=1,\dots,n_\tau = n-n_1 \tag{2.8}$$

are, in general, nonzero, and the initial connecting orbit on the branch of connecting orbits is found via a sequence of homotopies that locate successive zero intercepts of the τ_j in (2.8). In each homotopy step we compute a branch, that is, a one-dimensional manifold, of solutions. For this we must have $n_c - n_v = n-1$, where n_c is the number of constraints, and n_v is the number of scalar variables. We keep $u(1)$ free, while $u\,(0)$ is allowed to vary on the surface of the sphere in S_0 of radius ϵ_0: (i) according to (2.10) below at steps 0 through n_0, when λ is fixed and c_i in (2.10) play the role of the control parameters; and (ii) according to (2.3) at steps n_0+1 through n_0+n_λ, when λ varies.

Let $S_{0,k}$, $k=1,\dots,n_0$, be the right invariant subspace of $f_u(u_0,\lambda_0)$ corresponding to the eigenvalues $\mu_{0,1},\dots,\mu_{0,k}$. Then the first k columns $q_{0,1},\dots,q_{0,k}$ of Q_0 form an orthonormal basis of $S_{0,k}$ and the remaining $n-k$ columns $q_{0,k+1},\dots,q_{0,n}$ of Q_0 form an orthonormal basis of the orthogonal complement $S_{0,k}^\perp$.

1. Initialization.

Step 0. Initialize the problem parameter vector λ, and set the algorithm parameters ϵ_0 and T to small, positive values, so that $u(t)$ is approximately constant on $[0, T]$. Set

$$u(t) = u_0 + \epsilon_0 c_1 q_{01}, \quad 0 \le t \le 1, \tag{2.9}$$

or

$$u(t) = u_0 + \epsilon_0 c_1 q_{01} e^{\mathrm{Re}\mu_{0,1} t}, \quad 0 \le t \le 1, \quad \mathrm{Re}\mu_{0,1} > 0,$$

$\epsilon_1 = |u(1) - u_1|$, $c_1 = 1$, or -1 and $c_2 = \cdots = c_{n_0} = 0$.

2. Locating a connecting orbit, λ is fixed.

Step 1. Time integrate to get an initial orbit. Compute a solution branch to the system (2.1), (2.2), (2.8), (2.5), and

$$(u(0) - u_0) \cdot q_{0,i}(u_0, \lambda)/\epsilon_0 - c_i = 0, \;\; i = 1, \ldots, n_c = n_0, \tag{2.10}$$

in the direction of increasing T, until $u(1)$ reaches an ϵ_1-neighborhood of u_1 for some $\epsilon_1 > 0$. Scalar variables are T, $\epsilon_1 \in \mathbb{R}$, $\tau \in \mathbb{R}^{n-n_1}$. There are n differential equations with $n_c = 2n - n_1 + 1$ constraints and $n_v = n - n_1 + 2$ scalar variables, and hence $n_c - n_v = n - 1$. In practice one typically continues until ϵ_1 stops decreasing, its value being not necessarily small.

Step k, $k = 2, \ldots, n_0$ (for $n_0 > 1$). Compute a branch of solutions to the system (2.1)–(2.3), (2.8), (2.5)–(2.10) to locate a zero of, say, τ_{k-1} (while $\tau_1, \ldots, \tau_{k-2} = 0$, fixed). Free scalar variables are $\epsilon_1, c_1, \ldots, c_k, \tau_{k-1}, \ldots, \tau_{n-n_1}$. There are n differential equations with $n_c = 2n - n_1 + 2$ constraints and $n_v = n - n_1 + 3$ scalar variables, and hence $n_c - n_v = n - 1$.

3. Locating a connecting orbit, λ varies.

Step k, $k = n_0 + 1, \ldots, n_0 + n_\lambda \equiv n - n_1 + 1$. Compute a branch of solutions to the system (2.1)–(2.3), (2.8), (2.5)–(2.7) to locate a zero of, say, τ_{k-1} (while $\tau_1, \ldots, \tau_{k-2} = 0$, fixed). Free scalar variables are $\epsilon_1, \tau_{k-1}, \ldots, \tau_{n-n_1}, \lambda_1, \ldots, \lambda_{k-n_0} \in \mathbb{R}$, $u_0, u_1 \in \mathbb{R}^n$. There are n differential equations with $n_c = 4n - n_0 - n_1 + 2$ constraints and $n_v = 3n - n_0 - n_1 + 3$ scalar variables, and hence $n_c - n_v = n - 1$.

4. Increasing the accuracy of the connecting orbit.

Compute a branch of solutions to the system (2.1)–(2.7) in the direction of decreasing ϵ_1 until it is "small." Free scalar variables are $\epsilon_1, T, \lambda_1, \ldots, \lambda_{n_\lambda - 1} \in \mathbb{R}$, $u_0, u_1 \in \mathbb{R}^n$. As before, $n_c = 4n - n_0 - n_1 + 2$, $n_v = 3n - n_0 - n_1 + 3$.

5. Continue the connecting orbit.

Compute a branch of solutions to the system (2.1)–(2.7). Free variables are the (real) scalar T, and $\lambda_1, \ldots, \lambda_{n_\lambda}$, and the vectors $u_0, u_1 \in \mathbb{R}^n$. As before, $n_c = 4n - n_0 - n_1 + 2$, $n_v = 3n - n_0 - n_1 + 3$. Alternatively, a phase condition

$$\int_0^1 (u'(t) - q'(t)) \cdot u''(t)\, dt = 0 \tag{2.11}$$

may be added if T is kept fixed and ϵ_0 and ϵ_1 are allowed to vary. Here $q(t)$ is a previously computed orbit on the branch.

Remark 2. In principle, our algorithm of continuation of invariant subspaces of $f_u(u_0, \lambda)$ and $f_u(u_1, \lambda)$ in section 3 breaks down only if two eigenvalues, one associated with the subspace being continued and one not, approach the same point on the imaginary axis (one from the left and one from the right). In this case the algorithm should stop anyway since a bifurcation is being approached.

3. Continuation of invariant subspaces. Let $A(\lambda) \in \mathbb{R}^{n\times n}$ denote one of the following: $f_u(u_0, \lambda)$, $f_u(u_1, \lambda)$. The basic continuation algorithm requires at each pseudoarclength continuation step computation of a right invariant (typically, stable or unstable) m-dimensional subspace $S(\lambda)$ of $A(\lambda)$. In general, the function A is smooth in λ (say, differentiable), and it is important that $S(\lambda)$ be smooth also, as otherwise convergence difficulties can be expected.

In this section we show how to constructively obtain smooth and orthogonal $Q(\lambda) = [Q_1(\lambda)\ Q_2(\lambda)] \in \mathbb{R}^{n\times n}$, $Q_1(\lambda) \in \mathbb{R}^{n\times m}$, $Q_2(\lambda) \in \mathbb{R}^{n\times(n-m)}$ so that $Q_1(\lambda)$ span $S(\lambda)$ and $Q_2(\lambda)$ span the orthogonal complement $S(\lambda)^\perp$. Moreover, if we let $S_k(\lambda)$, $k = 1, \ldots, m$, be the right invariant subspaces of $A(\lambda)$ corresponding to the first k eigenvalues $\mu_1, \ldots, \mu_k$ of A, then the first k columns $q_1, \ldots, q_k$ of $Q(\lambda)$ form an orthonormal basis of $S_k(\lambda)$, and the remaining $n - k$ columns form an orthonormal basis of the orthogonal complement $S_k(\lambda)^\perp$.

To justify our construction, we should recall that in our continuation procedure we parametrize a solution branch in terms of so called pseudoarclength; let s denote the pseudoarclength variable.[1] Thus, both fixed points u_0 and u_1 (see (2.6)–(2.7)) as well as the parameter(s) λ are smooth functions of s. The matrix-valued function $A : \lambda \in \mathbb{R}^{n_\lambda} \to \mathbb{R}^{n\times n}$ can thus be viewed as a smooth function from $s \in \mathbb{R} \to \mathbb{R}^{n\times n}$. As a consequence, we can and will think of the continuation of invariant subspaces with respect to the scalar pseudoarclength variable s. For this reason, we will abuse notation and freely write $A(s)$ for $A(\lambda)$. In what follows, a "$\dot{}$" will indicate differentiation with respect to s.

Remark 3. If $n_\lambda = 1$ in (1.1), then one may be able to continue in λ itself, rather than reparametrizing the problem by arclength.

The basic issue is the following: suppose that initially we have the (real) block Schur factorization

$$A(0) = Q(0)R(0)Q^T(0), \quad Q(0) = [Q_1(0)\ Q_2(0)], \tag{3.1}$$

where $R(0)$ is block upper triangular ($R_{ii}(0)$, $i = 1, 2$, are not required to be triangular)

$$R(0) = \begin{bmatrix} R_{11}(0) & R_{12}(0) \\ 0 & R_{22}(0) \end{bmatrix},$$

the columns of $Q_1(0)$ span an invariant subspace $S(0)$ of $A(0)$, and the columns of $Q_2(0)$ span the orthogonal complement $S(0)^\perp$. We want to obtain a block Schur factorization for the matrix $A(s)$, close to $A(0)$, exploiting (if possible) the work already done to obtain the block Schur form for $A(0)$.

This problem fits within the general framework developed in [6]. Suppose that the matrix $A(s)$ has two groups of eigenvalues, $\Lambda_1(s)$ and $\Lambda_2(s)$, which stay disjoint for all s around 0. Then, in an interval about $s = 0$ there is the smooth factorization

$$A(s) = Q(s)R(s)Q^T(s), \quad Q(s) = [Q_1(s)\ Q_2(s)], \tag{3.2}$$

where $R(s)$ is in block Schur form $R(s) = \left[\begin{smallmatrix} R_{11}(s) & R_{12}(s) \\ 0 & R_{22}(s) \end{smallmatrix}\right]$. Here, R_{11} has eigenvalues Λ_1 and R_{22} has eigenvalues Λ_2. A constructive procedure to obtain the factorization

[1] Typically, this is the general procedure of most practical continuation algorithms; e.g., this is the strategy implemented in `AUTO`.

QRQ^T of A can be based upon differential equations models. As in [6], we differentiate the relations $A = QRQ^T$ and $Q^TQ = I$, let $H := Q^T\dot{Q}$, and obtain

$$\dot{R} = Q^T\dot{A}Q + RH - HR, \tag{3.3}$$

$$\dot{Q} = QH. \tag{3.4}$$

Now we use triangularity of R and the fact that H must be skew symmetric, $H^T = -H$; we partition H in the same way as R, and then can determine H_{12} by solving the Sylvester equation

$$R_{22}H_{12}^T - H_{12}^T R_{11} = (Q^T\dot{A}Q)_{21}. \tag{3.5}$$

The blocks H_{11} and H_{22} are not uniquely determined, and we may set them to 0 (in any case, they must be chosen skew symmetric). Thus, in principle, we can solve (3.3)–(3.4) subject to initial conditions (ICs) obtained from the factorization of $A(0)$, in order to obtain a smooth path of block Schur factorizations.

We can accumulate the transformations in such a way that we are always looking for corrections close to the identity. To be more precise, we can rewrite (for all s)

$$Q(s) = Q(0)U(s), \quad \text{with} \quad U(0) = I, \tag{3.6}$$

and use this in (3.4), thereby obtaining a differential equation for U, $\dot{U} = UH$ (notice that H is the same as before). We now look for exact solutions to the U differential equation. For all s, partition $U(s) = [U_1(s)\ U_2(s)] = \left[\begin{smallmatrix} U_{11}(s) & U_{12}(s) \\ U_{21}(s) & U_{22}(s) \end{smallmatrix}\right]$, with the same block dimensions as R. Since $U(0) = I$, there is an open interval about 0, call it I_0, where we can require that U_1 has the structure $U_1 = \left[\begin{smallmatrix} I \\ U_{21}U_{11}^{-1} \end{smallmatrix}\right] U_{11}$. Next, for all $s \in I_0$, we define

$$Y(s) := U_{21}(s)U_{11}^{-1}(s), \tag{3.7}$$

use the orthogonality relation $U_1^TU_1 = I$, and choose U_{11} symmetric,[2] to obtain $U_1 = \left[\begin{smallmatrix} I \\ Y \end{smallmatrix}\right](I + Y^TY)^{-1/2}$. In a similar way, for U_2 we use $U_2^TU_2 = I$ and $U_1^TU_2 = 0$, to eventually obtain, for every $s \in I_0$,

$$U(s) = \left[\begin{pmatrix} I \\ Y \end{pmatrix}(I + Y^TY)^{-1/2} \quad \begin{pmatrix} -Y^T \\ I \end{pmatrix}(I + YY^T)^{-1/2}\right]. \tag{3.8}$$

Thus, we need to find the function $Y \in \mathbb{R}^{(n-m)\times m}$ in (3.8). For any given $s \in I_0$, define $E(s)$ by

$$Q^T(0)A(s)Q(0) = Q^T(0)[A(0) + (A(s) - A(0))]Q(0) =: R(0) + E(s) = \begin{bmatrix} \widehat{R}_{11} & \widehat{R}_{12} \\ E_{21} & \widehat{R}_{22} \end{bmatrix}. \tag{3.9}$$

Now we substitute $Q(s)$ given by (3.6), (3.8), and $A(s)$ obtained from (3.9) into the invariant subspace relation,

$$Q_2^T(s)A(s)Q_1(s) = 0, \tag{3.10}$$

[2]This way, U_{11}^{-1} is the unique positive definite square root of $I + Y^TY$.

to obtain the following algebraic Riccati equation for Y:

$$\widehat{R}_{22}Y - Y\widehat{R}_{11} = -E_{21} + Y\widehat{R}_{12}Y \tag{3.11}$$

or

$$F(Y) = 0, \quad F(Y) := \widehat{R}_{22}Y - Y\widehat{R}_{11} + E_{21} - Y\widehat{R}_{12}Y. \tag{3.12}$$

Remark 4. If we think of $S(0)$ and $Q(0)$ as approximations to $S(s)$ and $Q(s)$, for s given, then we may interpret the form (3.8) as well as the resulting Riccati (3.11) as an iterative refinement technique to improve the accuracy of computed invariant subspaces. This is the viewpoint in the works of Stewart [15], Dongarra, Moler, and Wilkinson [11], Chatelin [3], and Demmel [5]. However, we prefer to think of (3.8) as the exact solution at a given s of the differential equation $\dot{U} = UH$ (see (3.4) and (3.6)). We will exploit this fact later.

How to solve (3.11). The following two iterative methods have been often advocated to solve the Riccati equation (3.11) (or (3.12)).

1. The iteration [15]:

$$\widehat{R}_{22}\Delta_k - \Delta_k\widehat{R}_{11} = -F(Y_{k-1}) \ , \ \ Y_k = \Delta_k + Y_{k-1}, \tag{3.13}$$

with $Y_0 = 0$, $k = 1, 2, \ldots$.

2. The Newton iteration [5]:

$$(\widehat{R}_{22} - Y_{k-1}\widehat{R}_{12})\Delta_k - \Delta_k(\widehat{R}_{11} + \widehat{R}_{12}Y_{k-1}) = -F(Y_{k-1}) \ , \ \ Y_k = \Delta_k + Y_{k-1}, \tag{3.14}$$

with $Y_0 = 0$, $k = 1, 2, \ldots$.

Therefore, we need to solve a Sylvester equation in the inner loop of the iterative refinement. This can be effectively done by using `LAPACK` routines. As shown in the convergence analysis for the iterations (3.13) and (3.14) by Stewart [15] and Demmel [5], respectively, if we let

$$\kappa = \frac{\|\widehat{R}_{12}\|_F \|E_{21}\|_F}{\mathrm{sep}^2(\widehat{R}_{11}, \widehat{R}_{22})}, \tag{3.15}$$

then under the assumptions $\kappa < 1/4$ and $\kappa < 1/12$, the iterations (3.13) and (3.14) converge linearly and quadratically, respectively. In (3.15), $\| \cdot \|_F$ is the Frobenius norm of a matrix.

The parameter κ can be interpreted as follows. Its numerator, $\|\widehat{R}_{12}\|_F \|E_{21}\|_F$, measures the quality of the initial approximate invariant subspace: it will be small when the approximation is good, and the factor $\|E_{21}\|_F$ will be zero if and only if the initial approximation is in fact correct. The function $\mathrm{sep}(\widehat{R}_{11}, \widehat{R}_{22})$ in the denominator is the smallest singular value of the operator which maps Y to $\widehat{R}_{22}Y - Y\widehat{R}_{11}$, and it measures the separation of the spectra of $\widehat{R}_{11}$ and $\widehat{R}_{22}$. If $\mathrm{sep}(\widehat{R}_{11}, \widehat{R}_{22})$ is small, it means that some eigenvalues of $\widehat{R}_{11}$ and $\widehat{R}_{22}$ can be made to merge with small changes in $\widehat{R}_{ii}$; this means that the invariant subspaces belonging to the two parts of the spectrum are unstable and hard to compute. Thus κ will be small if we start with a good initial approximate invariant subspace and if the eigenvalues associated with that subspace are well separated from the remainder of the spectrum. Oversimplifying, both algorithms converge if (i) the spectra of $R_{11}(0)$ and $R_{22}(0)$ are far enough apart and (ii) the perturbation $E(s)$ of $R(0)$ is small enough.

Remark 5. We point to some additional safeguards, which ensure proper performance of the algorithms (3.13) and (3.14).

1. Note that from (3.9) we have $\|E_{21}\|_F \le \|E\|_F \le \|A(s) - A(0)\|_F$. Hence, from (3.15), if

$$\alpha\|A(s) - A(0)\|_F \le \frac{1}{4}, \quad \alpha = \frac{\|\widehat{R}_{12}\|_F}{\mathrm{sep}^2(\widehat{R}_{11}, \widehat{R}_{22})}, \tag{3.16}$$

then $\kappa < 1/4$. Therefore we are guaranteed that for any matrix $X \in \mathbb{R}^{n\times n}$ with $\alpha\|X - A(0)\|_F \le \frac{1}{4}$ we can find the invariant subspaces S and $S^\perp$ by Algorithm (3.13), say. And no eigenvalue of $A\,|_S$ can "merge" with an eigenvalue from $A\,|_{S^\perp}$, where $A\,|_S$ denotes the restriction of A to S, etc. In other words, the eigenvalues of $A\,|_S$ and $A\,|_{S^\perp}$ remain separated for all matrices in a ball around $A(0)$ of radius $1/(4\alpha)$. Thus, employing the safeguard (3.16) ensures that the iterations (3.13) and (3.14) will diverge only when a small perturbation $A(s)$ of $A(0)$ will make an eigenvalue from $S(s)$ and an eigenvalue from $S(s)^\perp$ coalesce.

2. The quantity $\|E_{21}\|_F/\mathrm{sep}(\widehat{T}_{11}, \widehat{T}_{22})$ can be interpreted [13, pp. 347–348] as an estimate from above of the tangent of the angle between the subspaces $S(0)$ and $S(s)$ spanned by $Q_1(0)$ and $Q_1(s)$, respectively. Hence the convergence of our algorithms implies that this angle always stays between 0 and $\pi/2$. It is useful to monitor this angle and, in some situations, control it. A convenient measure in this case is the sine of this angle [13, p. 77] given by

$$\mathrm{dist}(S(0), S(s)) = \sqrt{1 - \sigma^2_{\min}\left(Q_1^T(0)Q_1(s)\right)}.$$

Taking into account that by (3.2), (3.6), and (3.8),

$$Q_1(s) = (Q_1(0) + Q_2(0)Y)(I + Y^TY)^{-1/2},$$

this reduces to

$$\mathrm{dist}(S(0), S(s)) = \sqrt{1 - \sigma^2_{\min}\left(I + Y^TY\right)^{-1/2}} = \frac{\|Y\|_2}{\sqrt{1 + \|Y\|_2^2}}. \tag{3.17}$$

The previous discussion has been motivated by the standard linear algebra viewpoint, that is, of "how to refine the initial trivial estimate $U = I$." In particular, this gave us the initial conditions $Y_0 = 0$ for the iterations (3.13) and (3.14). However, from the point of view of continuation, this is usually not the best strategy, since it amounts to starting with the old solution as initial guess. We can do better by using the differential equation formulation and taking an Euler approximation to the solution.

Recall that the matrix Y, solution of (3.11), is related to U by (3.7): $Y(s) = U_{21}(s)U_{11}^{-1}(s)$ for all s in a neighborhood of 0. Here, $U_1 = \left[\begin{smallmatrix} U_{11} \\ U_{21} \end{smallmatrix}\right]$, and $U = [U_1\ U_2]$ solves the differential equation $\dot{U} = UH$ (see (3.4)–(3.6)). That is,

$$\dot{U}_1 = U_1H_{11} + U_2H_{21}, \quad \dot{U}_2 = -U_1H_{21}^T + U_2H_{22}. \tag{3.18}$$

Recall that the factor H_{21} is determined by (3.5), but there is freedom insofar as the choice of H_{11} and H_{22}. We now show that the range of s-values guaranteeing invertibility of $U_{11}(s)$ is unaffected by the choice of H_{11} and H_{22} in (3.18), and hence there is little reason not to set H_{11} and H_{22} both equal to 0 when we represent U as in (3.8).

LEMMA 3.1. *Let $U = [U_1\ U_2]$ be the solution of* (3.18) *with $U(0) = I$, obtained by setting $H_{11} = 0$ and $H_{22} = 0$ in* (3.18)*, while determining H_{21} by* (3.5)*. Let $\tilde{U}(s) = [\tilde{U}_1\ \tilde{U}_2]$ be the solution of* (3.18) *with $\tilde{U}(0) = I$ and H_{11} and H_{22} nonzero skew symmetric matrices. Then, we have (for all s)*

$$\tilde{U}_1(s) = U_1(s)C(s),$$

where $C(s) \in \mathbb{R}^{m\times m}$ is orthogonal. Therefore, partitioning $U_1 = \left[\begin{smallmatrix} U_{11} \\ U_{21} \end{smallmatrix}\right]$, and similarly for $\tilde{U}_1$, we have $Y(s) = U_{21}(s)U_{11}^{-1}(s) = \tilde{U}_{21}(s)\tilde{U}_{11}^{-1}(s)$ for the same range of s values.

Proof. For all s, let $p(\lambda, s)$ be the characteristic polynomial of the matrix $A(s)$. Then, we have the factorization $p(\lambda, s) = p_1(\lambda, s)p_2(\lambda, s)$, where the factors p_1 and p_2 have no common root, and the roots of p_1 give Λ_1, while the roots of p_2 give Λ_2. Thus, we have

$$p_1(Q^T(0)A(s)Q(0), s)U_1(s) = 0$$

and

$$p_1(Q^T(0)A(s)Q(0), s)\tilde{U}_1(s) = 0.$$

Therefore, since $\operatorname{rank}(p_1(A(s), s)) = n - m$, then $\tilde{U}_1(s) = U_1(s)C(s)$, where $C(s)$ is orthogonal (since $\tilde{U}_1^T(s)\tilde{U}_1(s) = I$). ∎

Now, because of Lemma 3.1, we can just integrate the differential equations (3.18) with H_{11} and H_{22} equal to 0:

$$[\dot{U}_1\ \dot{U}_2] = [U_2H_{21}\ \ -U_1H_{21}^T], \quad U(0) = I.$$

So, the idea is to approximate the solution of these differential equations from 0 to s in order to get an approximation V_1 to $U_1(s)$ and thus obtain an approximation to $Y(s)$ from $V_{21}V_{11}^{-1}$, where we have partitioned $V_1 = \left[\begin{smallmatrix} V_{11} \\ V_{21} \end{smallmatrix}\right]$. We use a forward Euler step to obtain V_1.[3]

1. Initialization.

Set $U(0) = I$ and obtain $H_{21}(0)$ from solving the Sylvester equation

$$R_{22}(0)H_{21}(0) - H_{21}(0)R_{11}(0) = -\,(0\ I)\,Q(0)^T\dot{A}(0)Q(0)\,(I\ 0)^T\,. \tag{3.19}$$

2. Euler step.

Let $V_1 = U_1(0) + sU_2(0)H_{21}(0)$. Obtain initial guess for the Riccati equation (3.11) (or (3.12)):

$$Y_0 = V_{21}V_{11}^{-1}. \tag{3.20}$$

Remark 6. We now look at how this new initial guess Y_0 in (3.20) impacts convergence of the iterations (3.13) and (3.14). We also make some comments on expense.

1. Observe that the value of Y_0 in (3.20) is nothing but $Y_0 = sH_{21}(0)$. It is easy to verify that this is precisely the same approximation we would have obtained by using a forward Euler step to approximate the solution at s of the differential equation satisfied by Y.

[3]Use of the Euler step is an accepted standard in continuation algorithms, and it is the strategy implemented in `AUTO`.

2. By our construction, we see that Y_0 in (3.20) is a second order approximation (in s) to the exact solution $Y(s)$. More precisely, from Taylor expansion at $s = 0$ of the exact solution $Y(s)$, we immediately get that $Y(s) = s\dot{Y}(0)+O(s^2) = sH_{21}(0)+O(s^2)$. On the other hand, the initial guess $Y_0 = 0$ is only an $O(s)$ approximation to $Y(s)$.

3. In practice, in (3.19), we do not have a close expression for $\dot{A}(0)$, but we can replace it by the difference quotient $(1/s)(A(s) - A(0))$. With this choice, the value of Y_0 (defined as in (3.20)) turns out to be the solution of the Sylvester equation

$$R_{22}(0)Y_0 - Y_0R_{11}(0) = -E_{21}. \tag{3.21}$$

Since $\dot{A}(0) = (1/s)(A(s) - A(0)) + O(s^2)$, the value of Y_0 in (3.21) is still a $O(s^2)$ approximation to the exact $Y(s)$.

4. The estimates (and proof) of convergence for (3.13) and (3.14) relied on the value of κ in (3.15) to be sufficiently small. Much the same arguments can be used for the refined guess in (3.20) by the following simple modification. In $U(s)$ in (3.8), we let $Y(s) = Y_0 + \Delta Y$ and use this form of U in the invariant subspace relation (3.10). So doing, we obtain the new Riccati equation

$$(\widehat{R}_{22} - Y_0\widehat{R}_{12})\Delta Y - \Delta Y(\widehat{R}_{11} + \widehat{R}_{12}Y_0) = -F(Y_0) + \Delta Y R_{12}\Delta Y. \tag{3.22}$$

Now, it is a simple verification that iterative solution of (3.22) by either of the two iterations, (3.13) or (3.14), appropriately reformulated for the unknown ΔY, produces precisely the same sequence as the original iterations on the unknown Y had they been started with initial guess Y_0 in (3.20). As a consequence, the convergence results we quoted after (3.15) hold unchanged, except that we have a new κ value, call it $\tilde{\kappa}$:

$$\tilde{\kappa} = \frac{\|\widehat{R}_{12}\|_F\|F(Y_0)\|_F}{\mathrm{sep}^2(\tilde{R}_{11}, \tilde{R}_{22})}, \quad \tilde{R}_{11} = \widehat{R}_{11} + \widehat{R}_{12}Y_0, \quad \tilde{R}_{22} = \widehat{R}_{22} - Y_0\widehat{R}_{12}. \tag{3.23}$$

We now look at the ratio

$$\frac{\tilde{\kappa}}{\kappa} = \frac{\|F(Y_0)\|_F}{\|E_{21}\|_F}\left(\frac{\mathrm{sep}(\widehat{R}_{11}, \widehat{R}_{22})}{\mathrm{sep}(\tilde{R}_{11}, \tilde{R}_{22})}\right)^2.$$

By our previous remark, $F(Y_0)$ is close to 0 at second order in s, whereas E_{21} is only first order close to 0. To compare $\mathrm{sep}(\tilde{R}_{11}, \tilde{R}_{22})$ with $\mathrm{sep}(\widehat{R}_{11}, \widehat{R}_{22})$, we realize that $\mathrm{sep}(\tilde{R}_{11}, \tilde{R}_{22})$ is a second order approximation to $\mathrm{sep}(R_{11}(s),\ R_{22}(s))$, whereas $\mathrm{sep}(\widehat{R}_{11}, \widehat{R}_{22})$ is only a first order approximation to the same quantity. Therefore, in first approximation, $\left(\frac{\mathrm{sep}(\widehat{R}_{11}, \widehat{R}_{22})}{\mathrm{sep}(\tilde{R}_{11}, \tilde{R}_{22})}\right)^2$ is $1 + O(s/\mathrm{sep}(R_{11}(s),\ R_{22}(s)))$. To summarize, we have that

$$\tilde{\kappa}/\kappa = O(s), \tag{3.24}$$

with an obvious improvement in the radius of convergence for both iterations (3.13) and (3.14).

5. By creating the initial guess Y_0 from (3.20) we need to solve the Sylvester equation (3.19) or (3.21). Typically, this involves Schur reduction of the matrices $R_{11}(0)$ and $R_{22}(0)$, which is somewhat expensive. However, suppose we use the Newton iteration (3.14); then, at each iteration step we need to Schur factor the matrices $\widehat{R}_{11}+\widehat{R}_{12}Y_{k-1}$ and $\widehat{R}_{22}-Y_{k-1}\widehat{R}_{12}$. At the convergence of Newton's method, these are

precisely the matrices we will need for the next continuation step in (3.19). Therefore, no extra factorizations are needed in this case. For a practical comparison of the iterations for the two initial guesses (3.20) and $Y_0 = 0$, we refer to the next section. A more extensive comparison and further details can be found in [7].

Remark 7. An alternative to our approach for continuing orthogonal invariant subspaces could be easily obtained as follows. Let $Q(0) : Q^T(0)A(0)Q(0)$ be a Schur factorization of $A(0)$, $Q(0) = [Q_1(0)\ Q_2(0)]$ as usual, so that (say) $Q_1(0)$ spans the invariant subspace relative to the eigenvalues with positive real parts. Next, consider $A(s)$ which has the same number of eigenvalues with a positive real part as $A(0)$. Then, one may think to take the ordered Schur factorization of the matrix $A(s) :\ P^T A(s)P$, and partition $P = [P_1\ P_2]$ so that P_1 spans the invariant subspace relative to the eigenvalues with a positive real part of $A(s)$. In general, P_1 is not smooth (that is, it is not true that $P_1 = Q_1(0) + s\dot{Q}_1(0) + \cdots$). To enforce smoothness, we may solve an orthogonal Procrustes problem and replace P_1 with P_1V, where V is the orthogonal polar factor of $P_1^T Q_1(0)$ (see [13]). Essentially, this is the approach used by Beyn [2], and then implemented in `HomCont` [4]. However, we believe that the approach we have adopted is preferable to the one we just outlined. For one thing, the latter is usually more expensive than our approach. Moreover, unlike the one we just outlined, our approach to continuation of invariant subspaces is more in tune with the original continuation problem: continuation of the subspaces influences the continuation step, and using first derivative information (as we did to obtain (3.20)) is bound to reflect genuine difficulties of the original differential equation (such as nearing a bifurcation) into the continuation algorithm.

4. Example: Heteroclinic orbits in a four-dimensional singular perturbation problem. Consider the problem of finding traveling wave front solutions to the FitzHugh–Nagumo equations with two diffusive variables,

$$v_t = v_{xx} + v(v-a)(1-v) - w, \quad w_t = \delta w_{xx} + \epsilon(v - \gamma w), \tag{4.1}$$

for small positive ϵ, where δ ranges between a small and large value. For δ small this is a singularly perturbed reaction-diffusion system. In moving coordinates, $v_1 = v(z)$, $v_2 = v'(z)$, $w_1 = w(z)$, $w_2 = w'(z)$ with $z = t + cx$, the reduced ODE is

$$\begin{aligned} v_1' &= v_2, \\ v_2' &= cv_2 - v_1(1-v_1)(v_1 - a) + w, \\ w_1' &= w_2, \\ w_2' &= [cw_2 - \epsilon(v_1 - \gamma w_1)]/\delta. \end{aligned} \tag{4.2}$$

In [10] we located a heteroclinic orbit of (4.2). Here we first reproduced this result with our new code and then located a heteroclinic orbit with $\delta = \epsilon = 0.001$, $\gamma = 13.23529$, $a = 0.3$, and $c = 0.2571271$. In this case $n_0 = n_1 = 2$, where the relevant eigenvalues are $\mu_{0,1} = 0.6958$, $\mu_{0,2} = 257.2$, $\mu_{1,1} = -0.4247$, and $\mu_{1,2} = -0.06553$. We then performed a two parameter continuation in (δ, c) in the direction of increasing δ; see Figure 4.1 for the bifurcation diagram and Figure 4.2 for some typical solutions in (v_1, t) coordinates.

We have implemented four methods to solve the Riccati equation: (i) the "simple iteration" (3.13) with zero initial guess, (ii) Newton's method (3.14) with zero initial guess, (iii) simple iteration (3.13) with Euler initial guess, and (iv) Newton's method with Euler initial guess. The numerical results agree with our theoretical results; in

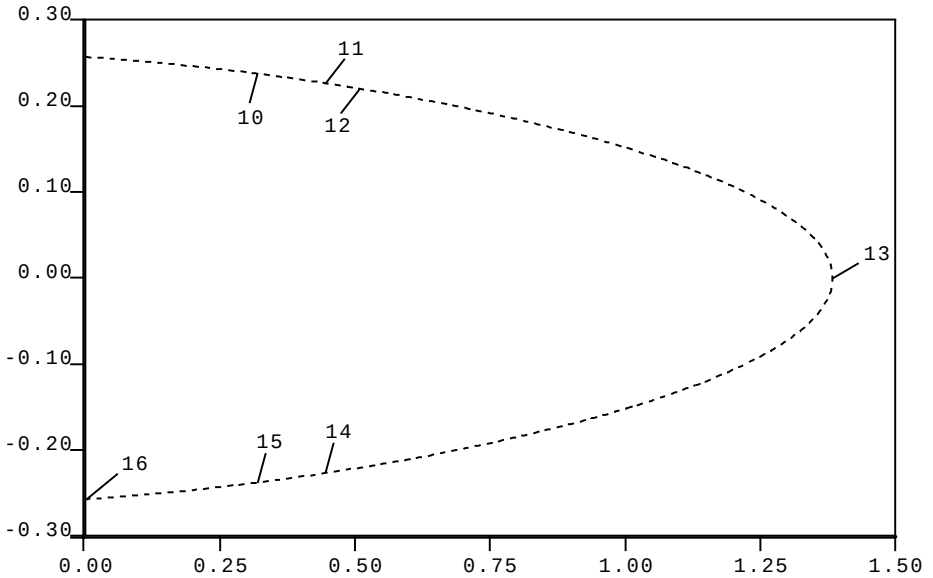


FIG. 4.1. *Bifurcation diagram in* (δ, c) *coordinates. At label* 10 $(\delta, c) = (0.3198,\ 0.2376)$ *and the eigenvalues are* $\mu_{0,1} = \mu_{0,2} = 0.7406$, $\mu_{1,1} = -0.4357$, *and* $\mu_{1,2} = -0.06502$. *At label* 11 $(\delta, c) = (0.4462, 0.2265)$ *and the eigenvalues are* $\mu_{0,1} = \mu_{0,2} = 0.6204$, $\mu_{1,1} = -0.4409$, *and* $\mu_{1,2} = -0.06565$. *The eigenvalues are complex between the labels* 10 *and* 11. *At label* 12 $(\delta, c) = (0.5085, 0.2202)$ *and the eigenvalues are* $\mu_{0,1} = 0.5098$, $\mu_{0,2} = 0.6538$, $\mu_{1,1} = -0.4436$, *and* $\mu_{1,2} = -0.06612$, *where* $|\mu_{1,1}| + |\mu_{1,2}| = \mu_{0,1}$. *At label* 13 $(\delta, c) = (1.383, 0)$ *and the eigenvalues are* $\mu_{0,1} = -\mu_{1,2} = 0.1099$, $\mu_{0,2} = -\mu_{1,1} = 0.5454$. *The part of the branch below the* δ*-axis is symmetric to the one above the* δ*-axis.*

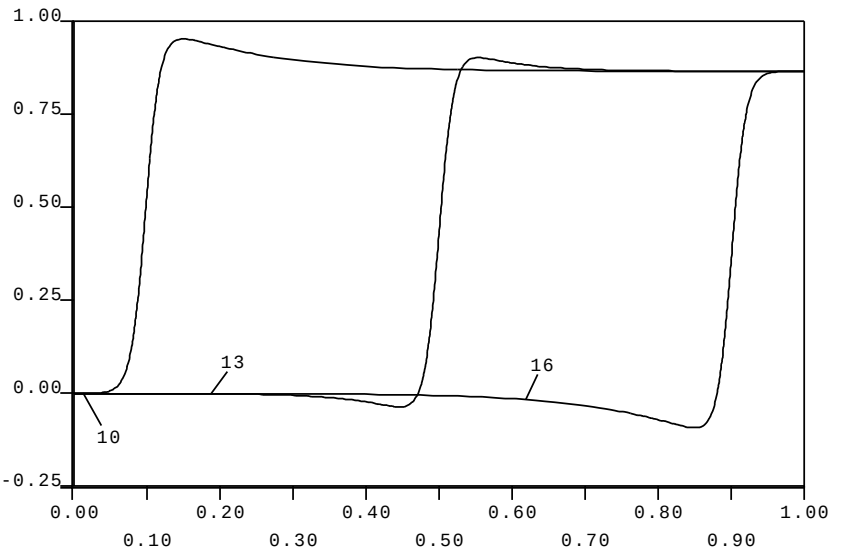


FIG. 4.2. *Some typical solutions in* (v_1, t) *coordinates.*

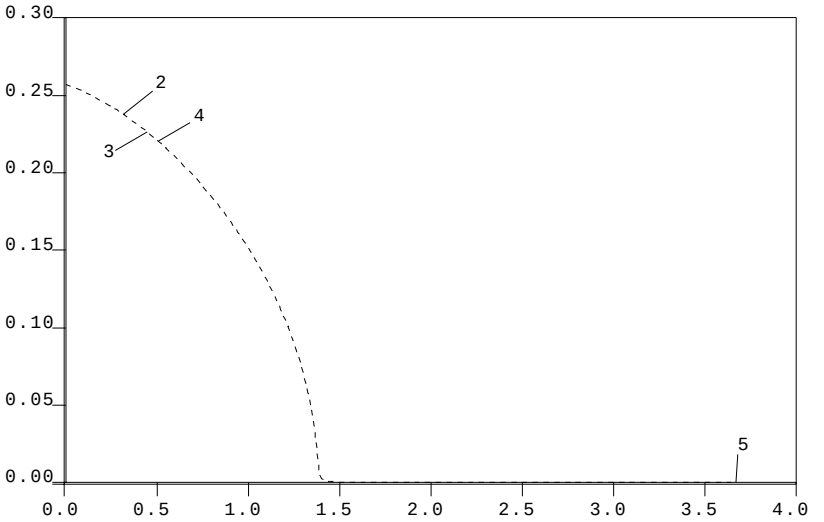


FIG. 4.3. *Bifurcation diagram in* (δ, c) *coordinates computed by* `HomCont` *(*`AUTO97`*).*

particular, the choice of the Euler initial guess is a big improvement with respect to the simpler zero guess. Specifically, for the same pseudoarclength continuation step (it took 433 such steps to compute the branch), (iii) and (iv) required much fewer iterations to converge than (i) and (ii). Some insight in the advantages gained by use of the Euler guess is obtained by considering the typical convergence behavior of the 4 methods (i)–(iv): on average, method (i) required 7 iterations for convergence, method (ii) needed 5, method (iii) needed slightly more than 4, and method (iv) required less than 3 iterations for convergence. In one exceptional case, methods (i) and (ii) required as many as 25 iterations. Finally, (i) and (ii) failed to converge in some cases (at the end of the continuation), because the continuation step was not small enough, whereas (iii) and (iv) never failed to converge.

Labels 10, 11, 12, 14, and 15 mark local bifurcations (of the eigenvalues), while label 13 marks a global bifurcation, the intersection of the current branch with the branch $c = 0$ (the detailed analysis of these bifurcations will be given elsewhere). For comparison, we repeated the computation of the above branch using `HomCont` (in `AUTO97`); see Figure 4.3. Note that in this case we have not suceeded in continuing the original branch beyond the global bifurcation point and instead switched branches. Indeed, by varying the continuation step size, with our code we could select the desired branch at label 13, but we could not achieve the same result with `HomCont`. Though we do not have a precise explanation for this difference in performance, we feel that this example confirms our theoretical insight that our numerical method is more in tune with the original continuation problem.

Acknowledgments. The authors wish to thank Z. Bai, University of Kentucky, for helpful discussions and for providing a subroutine for solving the Sylvester equation and W.-J. Beyn, University of Bielefeld, for helpful comments on an earlier version of the manuscript. The third author is also grateful to the IMA of University of

Minnesota, the Department of Mathematics of the University of Utah, the Computer Science Division of the University of California at Berkeley, and the Center for Dynamical Systems and Nonlinear Studies at the School of Mathematics of Georgia Tech for their hospitality during parts of his sabbatical stay, when this work was completed.

REFERENCES

[1] W.-J. Beyn, *The numerical computation of connecting orbits in dynamical systems*, IMA J. Numer. Anal., 9 (1990), pp. 379–405.

[2] W.-J. Beyn, *Global bifurcations and their numerical computation*, in Continuation and Bifurcations: Numerical Techniques and Applications, D. Roose et al., eds., Kluwer, Dordrecht, The Netherlands, 1990, pp. 169–181.

[3] F. Chatelin, *Simultaneous Newton's iteration for the eigenproblem*. Comput. Suppl., 5 (1984), pp. 67–74.

[4] A. R. Champneys, Yu. A. Kuznetsov, and B. Sandstede, *A numerical toolbox for homoclinic bifurcation analysis*, Internat. J. Bifur. Chaos Appl. Sci. Engrg., 6 (1996), pp. 867–887.

[5] J. Demmel, *Three methods for refining estimates of invariant subspaces*, Computing, 38 (1987), pp. 43–57.

[6] L. Dieci and T. Eirola, *On smooth decomposition of matrices*, SIAM J. Matrix Anal. Appl., 20 (1999), pp. 800–819.

[7] L. Dieci and M. Friedman, *Continuation of Invariant Subspaces*, Center for Dynamical Systems and Nonlinear Studies Tech. Report, Georgia Institute of Technology, Atlanta, CDSNS98-310, 1998.

[8] E. J. Doedel, A. R. Champneys, T. F. Fairgrieve, Yu. A. Kuznetsov, B. Sandstede, and X. J. Wang, `AUTO97:` *Continuation and bifurcation software for ordinary differential equations (with* `HomCont`*)* , available via FTP from ftp.cs.cs.concordia.ca in directory pub/doedel/auto, Department of Computer Science, Concordia University, Montreal, Canada, 1997.

[9] E. J. Doedel, X. J. Wang, and T. F. Fairgrieve, `AUTO94:` *Software for Continuation and Bifurcation Problems in Ordinary Differential Equations*, Applied Mathematics Report, California Institute of Technology, Pasadena, CA, 1994.

[10] E. J. Doedel, M. J. Friedman, and B. I. Kunin, *Successive continuation for locating connecting orbits*, Numer. Algorithms, 14 (1997), pp. 103–124.

[11] J. J. Dongarra, C. B. Moler, and J. H. Wilkinson, *Improving the accuracy of computed eigenvalues and eigenvectors*, SIAM J. Numer. Anal., 20 (1984), pp. 23–45.

[12] M. J. Friedman and E. J. Doedel, *Numerical computation and continuation of invariant manifolds connecting fixed points*, SIAM J. Numer. Anal., 28 (1991), pp. 789–808.

[13] G. H. Golub and C. F. Van Loan, *Matrix Computations*, The John Hopkins University Press, Baltimore, MD, 1989.

[14] G. Moore, *Computation and parametrization of periodic and connecting orbits*, IMA. J. Numer. Anal., 15 (1995), pp. 245–263.

[15] G. W. Stewart. *Error and perturbation bounds for subspaces associated with certain eigenvalue problems*, SIAM Rev., 15 (1973), pp. 727–764.

SIAM J. SCI. COMPUT.
Vol. 22, No. 1, pp. 95–112
© 2000 Society for Industrial and Applied Mathematics

A PARALLEL JACOBI–DAVIDSON-TYPE METHOD FOR SOLVING LARGE GENERALIZED EIGENVALUE PROBLEMS IN MAGNETOHYDRODYNAMICS*

MARGREET NOOL† AND AUKE VAN DER PLOEG‡

Abstract. We study the solution of generalized eigenproblems generated by a model which is used for stability investigation of tokamak plasmas. The eigenvalue problems are of the form $Ax = \lambda Bx$, in which the complex matrices A and B are block-tridiagonal, and B is Hermitian positive definite. The Jacobi–Davidson method appears to be an excellent method for parallel computation of a few selected eigenvalues because the basic ingredients are matrix vector products, vector updates, and inner products. The method is based on solving projected eigenproblems of order typically less than 30.

We apply a complete block LU decomposition in which reordering strategies based on a combination of block cyclic reduction and domain decomposition result in a well-parallelizable algorithm. One decomposition can be used for the calculation of several eigenvalues. Spectral transformations are presented to compute certain interior eigenvalues and their associated eigenvectors. The convergence behavior of several variants of the Jacobi–Davidson algorithm is examined. Special attention is paid to the parallel performance, memory requirements, and prediction of the speed-up. Numerical results obtained on a distributed memory Cray T3E are shown.

Key words. generalized eigenvalue problems, Jacobi–Davidson method, Arnoldi method, block-tridiagonal systems, parallelization

AMS subject classifications. Primary, 65-04, 65F10, 65F15, 65F50, 65N25; Secondary, 65Y05, 65Y20

PII. S106482759933290X

1. Introduction.

Consider the generalized eigenvalue problem

$$Ax = \lambda Bx, \qquad A, B \in \mathbb{C}^{N_t \times N_t}, \tag{1}$$

where A and B are complex block-tridiagonal $N_t \times N_t$ matrices and B is Hermitian positive definite. The number of diagonal blocks is denoted by N and the blocks are $n \times n$; therefore $N_t = N \times n$.

Eigenvalue problems arise in many applications. We are particularly interested in generalized eigenvalue problems as they occur in linear magnetohydrodynamics (MHD). Such problems are generated by a finite-element spectral code CASTOR (complex Alfvén spectrum of toroidal plasmas) [8]. This code is applied intensively at the FOM Institute voor Plasmafysica, Nieuwegein (the Netherlands) for the stability investigation of tokamak plasmas [9, 13]. The physicists are particularly interested in accurate approximations of certain interior eigenvalues, called the *Alfvén spectrum* and their associated eigenvectors. Figure 1 shows the complete and the Alfvén spectrum of (1) for a small test problem with $N = 50$ and $n = 32$. A target σ is given in the neighborhood in which we want to find several eigenvalues with corresponding eigenvectors. In general, the subblocks of A are dense and N_t can be very large,

*Received by the editors January 18, 1999; accepted for publication (in revised form) April 13, 1999; published electronically June 13, 2000. This research was sponsored partly by the Cray Research Grants Program (project CRG 96.14) of NCF (Dutch National Computing Facilities Foundation) and by the Priority Program "Massaal Parallel Rekenen" (project 95MPR04) of NWO (Dutch Organization for Scientific Research).

http://www.siam.org/journals/sisc/22-1/33290.html

†CWI, P.O. Box 94079, 1090 GB Amsterdam, The Netherlands (Margreet.Nool@cwi.nl).

‡MARIN, P.O. Box 28, 6700 AA Wageningen, The Netherlands (A.v.d.Ploeg@marin.nl).

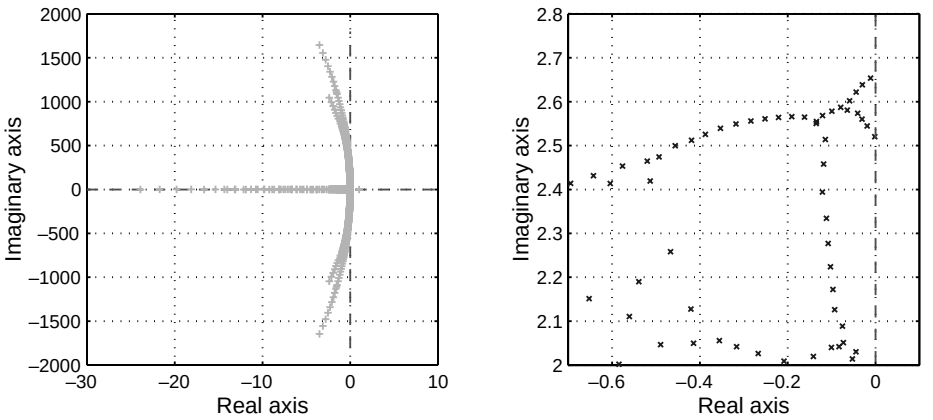


FIG. 1. *Entire (left) and Alfvén (right) spectrum of a small MHD problem:* $N = 50$, $n = 32$.

so computer storage demands are very high. Until now powerful shared memory machines with a huge amount of memory, such as the Cray C90, were used for this purpose. In this paper, we study as an alternative the feasibility of parallel computers with a large distributed memory for solving large generalized eigenvalue problems.

A promising method for computing selected eigenvalues of (1) is the Jacobi–Davidson (JD) algorithm [16, 3, 4, 5, 15]. The method has become very popular, and since we have started this project many aspects have been investigated, for instance, a variant with harmonic Ritz values has already been implemented [12]. In the future, we consider implementation of the Jacobi–Davidson QZ algorithm (JDQZ) [6] and inclusion of explicit deflation. In section 2 we briefly describe the JD algorithm for solving a block-tridiagonal eigenvalue problem. Within JD a parallel method to compute the action of the inverse of the block-tridiagonal matrix $A - \sigma B$ is used. In this approach, called DDCR, a block-reordering based on a combination of domain decomposition and cyclic reduction is combined with a complete block-tridiagonal LU decomposition of $A - \sigma B$, so $LU = A - \sigma B$. Both the construction of L and U and the triangular solves parallelize well. In this report we concentrate on the application of DDCR within JD. Originally, it was developed as a parallel preconditioner to apply within JD and appropriate for solving MHD eigenvalue problems. Some aspects of this technique are discussed in section 3.

However, the availability of a complete LU decomposition gives us the opportunity to apply the Jacobi–Davidson method to a *standard* eigenvalue problem instead of a *generalized* eigenvalue problem. Writing (1) as $(A - \sigma B)x = (\lambda - \sigma)Bx$, we have

$$x = (\lambda - \sigma)(A - \sigma B)^{-1}Bx.$$

If we define Q as $(A - \sigma B)^{-1}B$, for some value σ, we obtain the standard eigenvalue problem

$$Qx = \mu x \quad \text{with } \mu = \frac{1}{\lambda - \sigma} \Leftrightarrow \lambda = \sigma + \frac{1}{\mu}. \tag{2}$$

The eigenvalues μ of (2) form the dominant part of the spectrum, which makes them relatively easy to compute. In practice, one has to be careful because small pivot elements can be generated during the decomposition of $A - \sigma B$, especially when σ is

very close to an eigenvalue. In that case, taking $Q = (LU)^{-1}B$ in (2) may influence the computed spectrum. It is advisable to visualize the obtained eigenvalue spectrum and to compare it with results of a run with another target in the neighborhood of the spectrum. A smaller tolerance can also give information about the accuracy.

As a basis for the development of the parallel code, we have taken a sequential `FORTRAN` code for the JD algorithm, which was developed by the late Albert Booten at CWI [3, 4, 5]. The data structure for storing the matrices was modified in such a way that we could use optimized `BLAS` routines as much as possible. Furthermore, that part of the CASTOR code [8] which generates the matrices A and B has been parallelized and coupled with the JD code. Of course, other (generalized) eigenvalue problems satisfying (1) can be solved by our code. Several subroutines are available to read data efficiently from files and to distribute them to the processors. Section 4 describes the data organization of the JD code on the Cray T3E. Moreover, the memory requirements are given for both our sequential and parallel implementation.

In section 5 the convergence behavior of several variants of the Jacobi–Davidson algorithm is examined. Numerical results for three MHD eigenproblems obtained by a Cray T3E are presented and analyzed, and an analysis for the parallel speed-up is given. Conclusions are drawn in section 6.

2. The JD algorithm. Recently the JD method has been introduced as a new powerful technique for solving a variety of eigenproblems [16]. In this section we describe the method briefly and comment on our implementation. Suppose an eigenvector x is approximated by a linear combination of k search vectors v_j, $j = 1, 2, \ldots, k$, where k is very small compared with N_t. Let V_k be the $N_t \times k$ matrix whose columns are given by v_j. Then the approximation to the eigenvector can be written as $V_k s$, for some k-vector s. The search directions v_j are made orthonormal to each other, hence $V_k^* V_k = I$.

Suppose that an approximation to an eigenvalue is denoted by θ. The vector s and the scalar θ are constructed in such a way that the vector $r = QV_k s - \theta V_k s$ is orthogonal to the k search directions. From this Rayleigh–Ritz requirement it follows that

$$V_k^* Q V_k s = \theta V_k^* V_k s \Longleftrightarrow V_k^* Q V_k s = \theta s. \tag{3}$$

In this way one obtains a "projected" eigenvalue problem in which the order of the matrix $V_k^* Q V_k$ is k. By using a proper restart technique, k can remain so small that this problem can be solved by the QR method. The approximate eigenvalue θ is the eigenvalue of the projected system with the largest modulus.

At each step of the algorithm, a new search direction is constructed. Suppose that we have obtained an approximation $u = V_k s$ of the *true* eigenvector x associated with some eigenvalue μ. We assume that $\|u\|_2 = 1$, hence $\theta = u^* Q u$ is an approximation of μ. Let us define $P = uu^*$ as the orthogonal projector onto the subspace spanned by $\{u\}$. Then $I - P$ is the projector onto the orthogonal complement of span$\{u\}$, which is denoted by $u^\perp$. Any vector $x \in \mathbb{C}^n$ can be written as $x = x_1 + x_2$ with $x_1 \in$ span$\{u\}$ and $x_2 \in u^\perp$. We can scale x such that $x = u + z$ with $z \perp u$. In the JD algorithm a correction vector $z \in u^\perp$ is constructed. The restriction of Q to $u^\perp$ is given by

$$Q_{\mathrm{P}} = (I - P)Q(I - P). \tag{4}$$

If we rewrite (4) and substitute the resulting expression for Q into $Qx = \mu x$, we

Jacobi–Davidson for $Qx = \mu x, Q = (A - \sigma B)^{-1}B$.
Parameters: $iter, N_{ev}, tol_{JD}, k_{min}, m$ $(m \geq k_{min} + N_{ev})$, it_{SOL}.
step 0: initialize
Choose an initial vector v_1 with $\|v_1\|_2 = 1$; set $V_1 = [v_1]$;
$W_1 = [Qv_1]$; $k = 1$; $it = 1$; $n_{ev} = 0$
step 1: update the projected system
Compute the last column and row of $H_k := V_k^* W_k$
step 2: solve and choose approximate eigensolution of projected system
Compute the eigenvalues $\theta_1, \ldots, \theta_k$ of H_k and choose $\theta := \theta_j$ with $|\theta_j|$ maximal and $\theta_j \neq \mu_i$, for $i = 1, \ldots, n_{ev}$; compute associated eigenvector s with $\|s\|_2 = 1$
step 3: compute Ritz vector and check accuracy
Let u be the Ritz vector $V_k s$; compute the vector $r := W_k s - \theta u$;
`if` the residual $\hat{r} := (A - (\sigma + \frac{1}{\theta})B)u$ satisfies $\|\hat{r}\|_2/|\sigma + \frac{1}{\theta}| < tol_{JD}$ `then`
$n_{ev} := n_{ev} + 1$; $\mu_{n_{ev}} := \theta$; `if` $n_{ev} = N_{ev}$ `stop`; `goto` **2**
`else if` $it = iter$ `stop`
`end if`
step 4: solve correction equation approximately with it_{SOL} steps of GMRES
Determine an approximate solution $\tilde{z}$ of z in
$(I - uu^*)(Q - \theta I)(I - uu^*)z = -r \quad \wedge \quad u^* z = 0$
step 5: restart if projected system has reached its maximum order
`if` $k = m$ `then`
5a: set $k = k_{min} + n_{ev}$; compute the k eigenvalues of H_m with largest modulus; construct $C \in \mathbb{C}^{m \times k}$ with columns the associated eigenvectors; orthonormalize columns of C; compute $H_k := C^* H_m C$
5b: compute $V_k := V_m C$; $W_k := W_m C$
`end if`
step 6: add new search direction
$k := k + 1$; $it := it + 1$; call *MGS* $[V_{k-1}, \tilde{z}]$; set $V_k = [V_{k-1}, \tilde{z}]$; $W_k = [W_{k-1}, Q\tilde{z}]$;
`goto` **1**

FIG. 2. *JD algorithm after spectral transformation.*

obtain, using $Qu - \theta u = r$, $z \perp u$, $Pu = u$, and $Pz = 0$,

$$(Q_{\mathrm{P}} - \mu I)z = -r + (\mu - \theta - u^* Q z)u. \tag{5}$$

Since r is orthogonal to u, premultiplication of (5) by u^* yields $\mu = \theta + u^* Q z$. Note that μ is unknown and its best approximation will be θ. In this way, we obtain as a correction equation for z

$$(I - P)(Q - \theta I)(I - P)z = -r, \qquad u^* z = 0. \tag{6}$$

It is sufficient to solve (6) only approximately. This can be done by some steps of an iterative method, for example, GMRES [14]. When an approximate solution $\tilde{z}$ of (6) has been constructed, it is made orthogonal to the previous search directions, and the new search vector v_{k+1} is taken equal to $\tilde{z}/\|\tilde{z}\|_2$. Then k is increased by 1, and the new matrix $V_k^* Q V_k$ is constructed by expanding the "old" matrix by one new row and one new column.

Our implementation of the JD method for the computation of *several* eigenvalues is shown in Figure 2. The following remarks apply to the algorithm:

(i) The number of iteration steps that have been performed is denoted by it. The maximum allowed number of iterations is equal to $iter$.

(ii) The method is accelerated by searching for the best eigenvector approximation by selecting θ such that $|\theta|$ is maximal. To include the possibility of multiple eigenvalues, the condition $\theta_j \neq \mu_i, j = 1, \ldots, k$ must be interpreted as follows: an accepted eigenvalue μ_i can be equal to only one θ_j. So, if the projected system contains more than one θ_j close to μ_i, then it may be a multiple eigenvalue and, since its modulus is maximal, it becomes the next selected eigenvalue.

(iii) The value n_{ev} indicates the number of eigenvalues found so far that satisfy the acceptance criterion, and the parameter N_{ev} is the number of eigenvalues that we are looking for. The approximate eigenvalues θ that satisfy the acceptance criterion

$$\frac{\|\hat{r}\|_2}{|\sigma + \frac{1}{\theta}|} < tol_{JD} \quad \text{with } \hat{r} := (A - (\sigma + \tfrac{1}{\theta})B)u, \tag{7}$$

are referred to as μ_i for $i = 1, \ldots, n_{ev}$. The algorithm stops when n_{ev} is equal to N_{ev} or when *it* equals *iter*. In order to reduce computation time, we compute the actual residual $\hat{r}$ only if r satisfies

$$\frac{\|r\|_2}{|\theta|} < 100 \times tol_{JD} \quad \text{with } r := W_k s - \theta u = (Q - \theta I)u. \tag{8}$$

From experiments we observe that (7) is hardly satisfied when (8) is violated.

(iv) In the actual implementation of the algorithm, precautions have to be taken in step 2: theoretically, it is possible that all eigensolutions of the projected system satisfy the acceptance criterion. If that would happen, no new θ and corresponding approximate eigenvector u can be found in step 2. In that case, a vector $\tilde{z}$ is chosen that is not in the subspace spanned by V_k, and the algorithm proceeds at step 5.

(v) The subspace spanned by V_k contains the eigenvectors corresponding to the eigenvalues found before, together with some search directions. In this way, implicit deflation is incorporated automatically, which means that the detected eigenvectors are not filtered out. Of course, they may not influence the progress in finding the next eigenpair or lead to an eigenpair already found. The process of acceleration to find the next eigenpair, as described above, ensures that each accepted eigenpair will be unique. This technique differs from the deflation technique described in [6] which uses explicit deflation. The latter technique can be more stable but requires more operations per iteration step.

(vi) We require that the number of search directions k may not become too large. Therefore, if k has reached the value m, the upper bound for the order of the projected eigenvalue problem, k is reduced to $k_{min}+n_{ev}$. This is accomplished by extracting the most interesting information from H_k: the vectors which lead to previously accepted eigenvectors of the original problem are kept in the system together with those vectors that correspond to the k_{min} most promising eigenvalues, with $|\theta|$ maximal. In other words, $m-(k_{min}+n_{ev})$ vectors are thrown away. The columns of the remaining matrix $C \in \mathbb{C}^{m \times k}$ are again orthonormalized and a new H_k is obtained by $H_k := C^* H_m C$. Postmultiplication of V_m and W_m with C provides the new bases V_k and W_k. This restart technique is also applied in [6].

(vii) We use modified Gram–Schmidt (MGS) [7] to orthonormalize vectors;"call *MGS* $[V_{k-1}, \tilde{z}]$" in step 6 means that $\tilde{z}$ is made orthonormal to the columns of V_{k-1}. As suggested in [16] we apply MGS twice. Although classical Gram–Schmidt (CGS) has better parallel properties, we use MGS because it is more stable. Evidently, all inner products and all vector updates in our parallel implementation have been parallelized.

(viii) The original code that was taken as a basis for this project did not use harmonic Ritz values [6, 16]. Numerical experiments with a comparable algorithm using harmonic Ritz values have been described in [12]. The main advantage is that the LU decomposition of $A - \sigma B$ is used as a preconditioner and not as a shift and invert technique; then rounding-off errors in the decomposition for very large eigenvalue problems have less influence on the eigenvalue spectrum. We mention that the harmonic approach requires more memory and costs about 20% more execution time per Jacobi–Davidson iteration step.

2.1. Solving the correction equation. In step 4 of the JD algorithm (Figure 2), we must solve the correction equation

$$(I - uu^*)(Q - \theta I)(I - uu^*)z = -r \quad \text{and} \quad u^*z = 0 \tag{9}$$

approximately. From $u^*z = 0$, (9) can be rewritten as $(I-uu^*)(Q-\theta I)z = -r$, which is equivalent to $(Q - \theta I)z + \varepsilon u = -r$ with $\varepsilon = -u^*(Q - \theta I)z$ (cf. [16]). In order to construct a suitable preconditioner for (9), it may be useful to write the equation in the augmented form of order $N_t + 1$,

$$\begin{bmatrix} Q - \theta I & u \\ u^* & 0 \end{bmatrix} \begin{bmatrix} z \\ \varepsilon \end{bmatrix} = \begin{bmatrix} -r \\ 0 \end{bmatrix} \tag{10}$$

(cf. also [15, Theorem 3.5]). If $Q - \theta I$ is nearly singular, then the matrix will be ill-conditioned; the augmented form leads to a better conditioned matrix. The augmented correction equation can be solved, for instance, by applying some fixed number of steps of GMRES(m) (at most m steps with full GMRES, no restarts). A special choice for m is 0 (or $it_{SOL} = 0$), in which case the JD algorithm is closely related to the Arnoldi method [7]. Important differences between the JD algorithm of Figure 2 with $it_{SOL} = 0$ and the Arnoldi method are the structure of the matrix H_k and the way the restarts are performed. The projected matrices H_k obtained by applying the Arnoldi factorization are always upper Hessenberg with nonnegative subdiagonal elements. The matrices H_k obtained by k steps of JD do not have this structure (except for $it_{SOL} = 0$ and $k \leq m$, although rounding errors may appear for growing k). So, the computation of the eigenvalues of H_k requires an extra step to transform H_k to upper Hessenberg. Since JD does not have to repair the Hessenberg form, the restart technique in step 5 can be kept simple (see section 2). For details on the restart technique in the implicitly restarted Arnoldi method (IRAM), in which the Hessenberg structure of H_k has to be retained, we refer to [10]. A third difference with Arnoldi is the choice of θ; by selecting $|\theta|$ maximal the JD method is accelerated. We also refer to Discussion 4.1 in [15] for a clear discussion on the relation between the JD method, including the exact solution of the correction equation and shift-and-inverse Arnoldi. In that discussion no restarts are considered.

The convergence of GMRES(m) may be accelerated by incorporating a suitable (cheap) preconditioner. As in Booten et al. [5] and Sleijpen et al. [15], we rewrite the inverse of the augmented matrix, with $K = Q - \theta I$, as

$$\begin{bmatrix} K & u \\ u^* & 0 \end{bmatrix}^{-1} = \begin{bmatrix} I & -K^{-1}u \\ 0^* & 1 \end{bmatrix} \begin{bmatrix} I & 0 \\ u^*/\nu & -1/\nu \end{bmatrix} \begin{bmatrix} K^{-1} & 0 \\ 0^* & 1 \end{bmatrix} \tag{11}$$

with $\nu = u^*K^{-1}u$. It is easy to verify that

$$K = Q - \theta I = -\theta(A - \sigma B)^{-1}[(A - (\sigma + \tfrac{1}{\theta}B)] \tag{12}$$

and K^{-1} may be approximated by $-\frac{1}{\theta}I$. Hence, by replacing K^{-1} with $-\frac{1}{\theta}I$ in (11) and ν by $-\frac{1}{\theta}u^*u = -\frac{1}{\theta}$, we obtain the preconditioner $\mathcal{M}^{-1}$:

$$\begin{bmatrix} K & u \\ u^* & 0 \end{bmatrix}^{-1} \approx \begin{bmatrix} -\frac{1}{\theta}(I - uu^*) & u \\ u^* & \theta \end{bmatrix} = \mathcal{M}^{-1}. \tag{13}$$

The augmented equation (10) can be preconditioned by $\mathcal{M}^{-1}$ and then in almost all cases the number of JD iterations needed to find the first eigenvalue will be smaller than in cases where no preconditioning is used, as numerical experiments demonstrate in section 5.

3. The LU decomposition. To calculate an Alfvén spectrum we choose some target values σ in the neighborhood of this spectrum. For each σ we compute a complete LU decomposition of $A-\sigma B$. One (L, U) pair can be used for the calculation of several eigenvalues. However, if the method becomes too slow, it might be useful to choose a new target σ.

3.1. Block-tridiagonal LU approach. The matrices A and B in (1) have the same block-tridiagonal structure and therefore $A-\sigma B$ is also block-tridiagonal. Since the subblocks are more than 50% full, a complete LU decomposition of $A - \sigma B$ is relatively cheap: the total number of nonzeros in $L+U$ is at most twice as much as in $A-\sigma B$. In the rest of this paper, this technique is referred to as the block-tridiagonal LU approach. The decomposition is performed on a block level. This enables us to use partial pivoting, which makes the decomposition more robust. In order not to disturb the block-tridiagonal structure, the search for pivot elements is restricted to the blocks on the main diagonal. In our experiments, this pivot strategy seems to work well.

For a large number of diagonal blocks, the total number of complex multiplications required for the construction of L and U is approximately $\frac{7}{3}\ Nn^3$, and the number of multiplications required for performing the triangular solves with both L and U once is approximately $3Nn^2$.

3.2. The DDCR method. To improve parallelization possibilities of block-tridiagonal LU decomposition, we use a reordering based on a combination of domain decomposition (DD) and cyclic reduction (CR), discussed in [18].

Let p be the number of processors that is actually used, and let the integer $N_p = \lceil \frac{N}{p} \rceil$ [1] represent the number of diagonal blocks to be treated on each processor (except possibly the last processor, on which the number of diagonal blocks can be less). The first step of the DDCR method is to perform a block-reordering of both block rows and block columns based on a domain decomposition strategy with p nonoverlapping subdomains. The second step of the DDCR method is to construct a block lower-triangular matrix L and a block upper-triangular matrix U in such a way that $A - \sigma B = LU$ and all blocks on the main diagonal of U are identity matrices.

For N_p large, the construction of a block-tridiagonal LU decomposition of the first block-tridiagonal matrix on the first processor costs about $\frac{7}{3}\ n^3N_p$ multiplications. In [18] it is shown that the rest of the block-tridiagonal LU decomposition costs about $\frac{19}{3}\ n^3(N - N_p)$ multiplications. Hence the total number of multiplications required for the construction of L and U is approximately

$$(\tfrac{7}{3}\ N_p + \tfrac{19}{3}\ (N - N_p))n^3 = \tfrac{1}{3}\ (19N - 12N_p)n^3. \tag{14}$$

[1] By $\lceil x \rceil$ we denote the smallest integer $\geq x$ and by $\lfloor x \rfloor$ the largest integer $\leq x$.

In the same way it can be shown that for large N_p the number of multiplications required for performing the triangular solves with both L and U once is approximately

$$(3N_p + 5(N - N_p))n^2 = (5N - 2N_p)n^2. \tag{15}$$

4. The parallel implementation and data organization. We started this project with the parallelization of the original code [3, 4, 5] on a Cray T3D in Eagan, MN. The last part of the development has been done on a Cray T3E in Delft, the Netherlands. Some important characteristics of the Cray T3E in Delft are listed in Table 1.

TABLE 1
Main characteristics of the Cray T3E at HPαC centre, Delft.

# processors	80 (72 in parallel)
Clock period	3.33 ns clock
Peak performance / processor	600 Mflop/s
Total peak performance	33.6 Gflop/s
Local memory	128 Mbytes
Total memory	10 Gbytes
Data cache size	8 Kbytes
Compiling system	Cray CF90
GiGaRing channels	available

The communication steps in the recent code have been implemented with fast `SHMEM` routines (shared memory access library). These `SHMEM` routines are data passing library routines similar to message passing library routines and minimize the overhead associated with data passing requests, maximize bandwidth, and minimize data latency. Optimized `BLAS` routines are used as much as possible to achieve high performance per processor, and for the linear algebra part LAPACK routines [1] are called. For considerations on the programming technique, data distribution and communication on the MHD application, we refer to [11].

4.1. Distribution of the MHD matrices. For large problems with a lot of data, the I/O is often very expensive. Therefore, we do not read the MHD matrices from the files but generate them during execution, which takes only a few seconds. A part of the CASTOR code, which computes the matrix elements, has been coupled with our JD code. Moreover, it has been parallelized such that the available processors generate their own pieces of the matrices A and B in (1). The matrices are distributed block-row-wise along the processors; each processor gets one or more main diagonal blocks with their nearest sub- and superdiagonal blocks belonging to the same block row. All vectors are distributed accordingly. As a consequence, for a matrix-vector multiplication the memory traffic (of only two vectors of length n) is reduced to the first sub- and last superdiagonal block per processor.

4.2. Data format: CRS format versus dense block-tridiagonal format. The compressed row storage (CRS) format puts the subsequent nonzeros of the matrix rows in contiguous memory locations. To describe a sparse matrix A, we need three vectors, one for complex nonzero floating point numbers (mat_A) and two for integers ($colind\_A$, $rowptr\_A$). For more details and examples, we refer to [2].

CRS format has the advantage that it can be used to store general sparse matrices. However, the DDCR decomposition is completely based on the block-tridiagonal form of $A - \sigma B$. This implies that, for our applications, CRS format can be advantageous only if the blocks of A and B are sparse enough. From Table 2, which gives the

TABLE 2
Times for the matrix–vector multiplication on a single processor of a Cray T3E. Results for the block-tridiagonal form (dense blocks including zero values) are denoted by `DENSE`*. Results for CRS format are denoted by* `CRS`*(A) and* `CRS`*(B) for A and B, respectively.*

Execution times in μ-seconds on a Cray T3E, $N = 40$.					
MATVEC	$n = 16$	$n = 32$	$n = 48$	$n = 64$	$n = 80$
`DENSE BLOCKS`	4490	14577	32055	54666	93924
`CRS`(A)	2469	9874	22518	33733	52861
`CRS`(B)	1421	4979	10806	15814	24374
Number of nonzeros in A and B.					
$nnz(A)$	11392	47906	109542	196300	308180
$nnz(B)$	5934	22778	50532	89196	138770

execution times for matrix-vector multiplications on a single processor of a Cray T3E for matrices A and B generated by the CASTOR code, we may conclude that CRS format is to be preferred for both A and B on this platform.

4.3. Memory requirements. In this section, we will indicate which amount of memory is required for both the sequential and parallel implementation, expressed in $nnz(A)$ and $nnz(B)$, the numbers of nonzero values in the A and B, respectively, and N, n, and N_{ev}, where N denotes the number of diagonal blocks, n the blocksize of the block-tridiagonal matrices, and N_{ev} the number of eigenvalues we are looking for. The last parameter is m, the maximum allowed value of the projected system. As stated in section 2, m is much smaller than the order $N \times n$ of the matrices A and B.

A `CRS`-formatted matrix of $nnz(A)$ nonzero values requires 16 $nnz(A)$ bytes for storing *mat_A* and 8 $nnz(A)$ bytes for *colind_A*, respectively (cf. section 4.2). The integer array *rowptr_A* uses an additional 8 Nn bytes. The memory space for the `CRS`-formatted matrix B can be expressed in $nnz(B)$, analogously. The block-tridiagonal LU decomposition needs 3 Nn^2 `COMPLEX` words. The subspace matrix V_k and associated space W_k use 2 Nnm `COMPLEX` words, and the restart in step 5 of Figure 2 requires (in the worst case) another Nnm `COMPLEX` words. The total requirements for those matrices are 3 Nnm `COMPLEX` words. The accepted eigenvectors are stored in a matrix of order $Nn \times N_{ev}$. An additional working space for some vectors takes 5 Nn `COMPLEX` words.

Besides the matrix H_k, its Hessenberg form and its eigenvectors plus some working space require $4m(m+1)$ `COMPLEX` words. Summarizing, the amount of memory required for the current *sequential* implementation in complex arithmetic, where one `COMPLEX` word corresponds with 16 bytes, is approximately

$$24\,(nnz(A) + nnz(B)) + 16Nn(3(n+m) + N_{ev} + 7) + 64m(m+1) \text{ bytes.} \tag{16}$$

For the MHD matrices, the average sparsity of the sub-, super-, and diagonal blocks of A is approximately 41% and of B 20%, i.e., $nnz(A) \approx 0.41 \times 3Nn^2$ and $nnz(B) \approx 0.20 \times 3Nn^2$. Thus, the amount of memory for the sequential implementation (16) approximates

$$16Nn(5.75n + 3m + N_{ev} + 7) + 64m(m+1) \text{ bytes.} \tag{17}$$

The parallel implementation requires per processor approximately an additional $32n(N_p n + 3n)$ bytes for the DDCR solver and some extra memory (10 %) to distribute the CRS matrices, in case their sparsity is not equally partitioned over the block

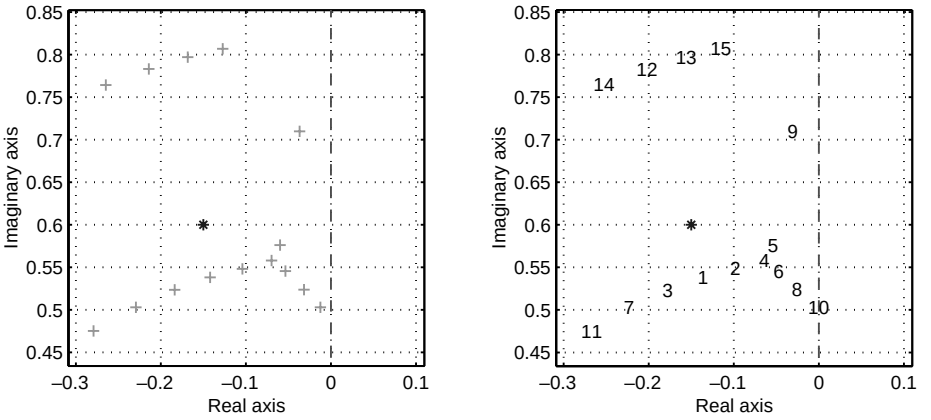


FIG. 3. *Part of the eigenvalue spectrum of an MHD problem with* $N = 40, n = 64$. *The target* σ *is indicated by an "*$*$*". In the right picture the order in which they appear are shown.*

rows. The small matrix of the projected system is present on all processors, and sequential steps, like the solution of the projected eigenvalue problem, are performed by all processors to avoid communication. Summarizing, the amount of memory (per processor) for our *parallel* implementation approximates

$$\begin{aligned} &48N_p n^2 + 16N_p n(3(n+m) + N_{ev} + 7) + 32n(N_p n + 3n) + 64m(m+1) \text{ bytes} \\ &= 128N_p n^2 + 16N_p(3m + N_{ev} + 7) + 96n^2 + 64m(m+1) \text{ bytes.} \end{aligned} \tag{18}$$

5. Numerical results.

5.1. The influence of the correction equation. In the left picture of Figure 3, the eigenvalues of an MHD problem with $N = 40$ and $n = 64$ near the target $\sigma = (-0.15, 0.60)$ are plotted. In the right picture, the eigenvalues are numbered consecutively as they are found. The solid line in Figure 4 displays the convergence behavior of this MHD problem solved by JD with $it_{SOL} = 0$. Each "$\circ$" indicates a converged eigenvalue. Obviously, to find the first eigenvalue is rather expensive, but after that only a few iteration steps are required to find the next ones. Note that after 30 and 39 JD iteration steps two eigenvalues were even found in one step. It appears that with this target even more eigenvalues (probably of the upper Alfvén branch) can be found than the 15 we asked for. However, since an implicit restarting technique is used, the order of the projected system will then grow . A better solution will be to choose a new target based on the results of the last (few) runs.

The question arises if it is possible to reduce the number of JD iteration steps when `GMRES(m)` is applied to the augmented correction equation (10). In Table 3, we distinguish four variants to solve the correction equation:

- `JD-sol`: in each JD step `GMRES(m)` is applied *without* preconditioning.
- `JD-sol`*: `GMRES(m)` is applied only if $\|\hat{r}\|_2 < \varepsilon_{tr}$ *without* preconditioning.
- `JD-solP`: in each JD step `GMRES(m)` is applied *with* preconditioning from the left by $\mathcal{M}^{-1}$.
- `JD-solP`*: `GMRES(m)` is applied only if $\|\hat{r}\|_2 < \varepsilon_{tr}$ *with* preconditioning from the left by $\mathcal{M}^{-1}$.

The preconditioning matrix $\mathcal{M}^{-1}$ has been introduced in section 2.1. The JD-sol variant does not improve the convergence process: from Figure 4 we know that if $it_{SOL} = 0$, 23 iterations are needed to get the first eigenvalue. It is known that in the

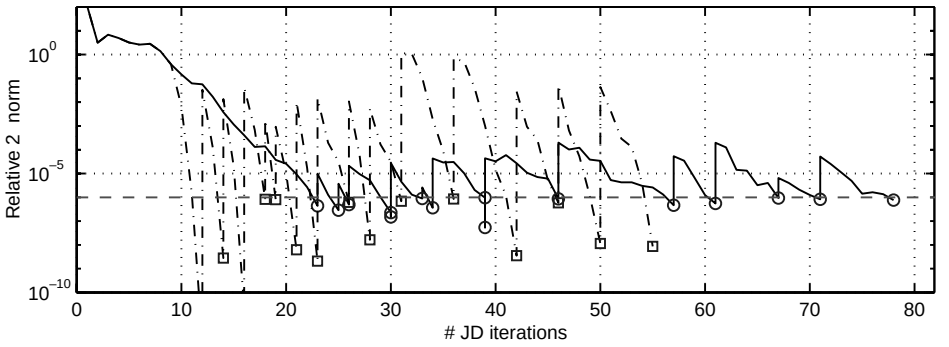


FIG. 4. *Convergence behavior of JD with* $it_{SOL} = 0$ *(solid line with "○"s) and* `JD-solP*`, `GMRES(20)` *(dashed line with "□"s);* $N = 40, n = 64$.

TABLE 3

Both the number of JD iterations as well as the execution time (in seconds on four processors) are given for several JD implementations with $it_{SOL} = 1$ *and* $\varepsilon_{tr} = 1.0$.

	Number of JD steps to find first eigenvalue					
	`GMRES(1)`	`GMRES(2)`	`GMRES(5)`	`GMRES(10)`	`GMRES(15)`	`GMRES(20)`
`JD-sol`	23	25	26	30	32	26
`JD-sol*`	23	24	21	21	13	12
`JD-solP`	23	33	13	19	22	28
`JD-solP*`	23	28	15	12	12	12
	Execution time to find first eigenvalue					
`JD-sol`	1.68	2.55	4.89	10.09	15.58	16.38
`JD-sol*`	1.44	1.97	2.69	4.51	2.33	2.28
`JD-solP`	1.69	3.46	2.31	6.20	10.49	17.82
`JD-solP*`	1.45	2.46	1.49	1.36	1.76	2.29
	Number of JD steps to find 15 eigenvalues					
`JD-sol`	72	139	140	109	107	82
`JD-sol*`	72	105	122	100	92	70
`JD-solP`	74	188	117	97	79	71
`JD-solP*`	75	147	119	90	71	55
	Execution time to find 15 eigenvalues					
`JD-sol`	7.19	17.61	29.82	39.25	54.80	54.53
`JD-sol*`	6.87	12.94	24.98	33.77	43.61	40.43
`JD-solP`	7.43	23.74	25.08	35.20	40.63	47.26
`JD-solP*`	7.26	18.15	24.33	30.25	31.83	30.45

first steps of the process the vectors v_k are usually poor approximations of the wanted eigenvectors, and solving the correction equation more accurately will not speed up the convergence. Therefore, we take in the first steps $it_{SOL} = 0$ until the 2-norm of the actual residual $\hat{r}$ in (7) drops below a threshold value ε_{tr}, i.e., for $\varepsilon_{tr} = 1.0$ after 9 "Arnoldi" steps then only an additional 3 steps are required to find the first eigenvalue (`JD-sol*`, `GMRES(20)`). We refer to [6] for more details, discussions and variants of the JD method and the influence of the way the correction equation is solved on the convergence behavior of the JD method.

If the correction equation is solved by `GMRES(m)` preconditioned from the left by $\mathcal{M}^{-1}$, then in almost all cases the number of JD iterations to the first eigenvalue is smaller than when no preconditioning is applied. This is also true for the complete process of finding 15 eigenvalues, when sufficient `GMRES` inner iterations are applied. Compared to a multiplication with $Q - \theta I$, the application of $\mathcal{M}^{-1}$ is relatively cheap

and as a consequence, the P-variants are usually much faster than those without preconditioning. The convergence history of JD-solP* with GMRES(20) is also displayed in Figure 4. Notice that the residual after an eigenpair is accepted in case $it_{SOL} = 0$ remains much smaller ($\approx 10^{-4}$) compared to the JD-solP*-variant.

However, each GMRES(m) step requires a multiplication with $Q - \theta I$, a very expensive operation even when performed in parallel. As a consequence, a reduction in the number of JD steps due to applying GMRES(m) does not guarantee a reduction in execution time. In fact, the timings listed in Table 3 indicate that GMRES(1) provides the fastest way to find 15 eigenvalues. When we replace step 4 in the algorithm of Figure 2 by $\tilde{z} = r$ (corresponding to $it_{SOL} = 0$) the number of JD steps hardly increases, resulting in the lowest execution time (see Table 4). Based on these and other results (not included in this paper), we continue with JD with $it_{SOL} = 0$.

TABLE 4
Both the number of Jacobi–Davidson iterations as well as the execution time (in seconds on four processors) is given for JD with $it_{SOL} = 0$.

	1 eigenvalue		15 eigenvalues	
	# JD steps	Time	# JD steps	Time
JD, $it_{SOL} = 0$	23	0.99	78	5.23

5.2. Parallel performance on the Cray T3E. In the rest of this paper, we set $it_{SOL} = 0$ which implies that $\tilde{z} = r$ (cf. Table 2, step 4). From numerical experiments this appears to be the best choice for minimizing the total wall clock time, as is illustrated in section 5.1. Three generalized eigenvalue problems, exploited by the linearized MHD equations (7)–(10) in [17] with resistivity $\eta = 5 \cdot 10^{-6}$ and ratio of specific heats $\gamma = \frac{5}{3}$, generated by CASTOR have been timed. Their specifications are given in Table 5. N is the number of diagonal block and the number of Fourier

TABLE 5
The MHD problems used in the numerical experiments.

Problem	N	n	σ	$nnz(A)$	$nnz(B)$
I	40	64	(-0.15, 0.70)	196300	89196
II	160	64	(-0.20, 0.70)	798632	362991
III	320	128	(-0.20, 0.70)	6491568	2886495

modes is equal to $n/16$, where n denotes the blocksize. All problems describe the same physical problem. In [17] more information on the background concerning resistive magnetohydrodynamic spectra in tokomaks can be found. The other parameters in the algorithm were chosen as follows:

$$iter = 300;\ N_{ev} = 15;\ tol_{JD} = 10^{-6};\ k_{min} = 10;\ m = 30.$$

Table 6 shows the results for the test problems I, II, and III. The third column shows the wall clock time (measured in seconds) required for the construction of the parallel block-tridiagonal LU decomposition. The numbers in parentheses show the Mflop rates, calculated by using the estimate (14) for the number of multiplications. For the block-tridiagonal LU decomposition we obtain a reasonable fraction of the theoretical peak performance, which is due to the level-3 BLAS routines that perform most of the work in this preprocessing phase. Problem I shows that when the number of processors increases from one to two, the wall clock time increases, also. This is due to the computation of the extra fill-in blocks in L and U which are generated

when the reordering technique for parallel processing is performed. Increasing the number of processors again by a factor two approximately halves the wall clock time. The reduction in wall clock time for $p = 40$ compared with $p = 20$ is very poor due to the fact that pure cyclic reduction has been applied, since $N = p$. When we calculate those speed-ups for Problems II and III, we see that a gain of 1.44 and 1.65 is reached, respectively. From the performance of more than 10 Gflop/s on 64 processors achieved for the largest problem, we may conclude that the block-tridiagonal LU decomposition is extremely suitable for this platform. However, we must realize that for growing block size the decomposition becomes very expensive.

TABLE 6

Wall clock times (in seconds) of several parts of the JD process on the Cray T3E for Problems I, II, *and* III.

Problem	p	Time for the LU decomposition		Time for JD process	Number of JD steps	Sequential time	Total time triangular solves	
I	1	1.06	(184)	16.99	81	0.77	4.74	(67)
I	2	1.15	(315)	10.07	85	0.81	3.92	(114)
I	4	0.60	(750)	5.74	87	0.82	2.11	(244)
I	5	0.52	(899)	4.78	80	0.78	1.76	(275)
I	8	0.34	(1440)	3.62	84	0.85	1.23	(424)
I	10	0.32	(1542)	3.23	79	0.74	1.17	(425)
I	20	0.26	(2003)	2.70	79	0.72	1.02	(495)
I	40	0.24	(2155)	2.44	71	0.76	0.91	(508)
II	2	4.85	(300)	153.60	300	3.77	57.98	(109)
II	4	2.44	(732)	72.59	300	3.77	29.29	(242)
II	5	2.00	(930)	59.40	300	3.76	24.17	(299)
II	8	1.27	(1547)	30.14	247	3.08	12.69	(485)
II	10	1.07	(1864)	25.23	241	3.01	10.62	(571)
II	16	0.70	(2908)	21.32	300	3.74	8.84	(868)
II	20	0.62	(3305)	21.08	300	4.61	8.18	(942)
II	32	0.45	(4654)	16.13	300	4.08	6.06	(1282)
II	40	0.43	(4857)	10.26	198	2.41	4.01	(1281)
III	10	13.52	(2355)	184.07	300	3.49	95.88	(630)
III	16	8.81	(3708)	109.82	300	3.58	61.91	(991)
III	20	7.28	(4525)	92.13	300	3.38	53.64	(1149)
III	32	4.97	(6700)	39.38	196	2.19	23.87	(1700)
III	40	4.42	(7565)	54.45	300	3.53	33.80	(1842)
III	64	3.22	(10461)	39.50	300	3.59	25.41	(2461)

The fourth column shows the wall clock time required for the JD process without the time required for preprocessing. Since a part of the algorithm is not performed in parallel (solution of the projected eigenvalue problem), we cannot expect linear speed-up. The fifth column gives the number of Jacobi–Davidson iteration steps. If this number is equal to 300, less than 15 eigenpairs have been found. In those cases it should have been better to choose a smaller value for N_{ev}. For Problem II approximately 54 iterations were needed to obtain 14 eigenpairs. We emphasize that it is hard to predict how many eigenvalues can be found in the neighborhood of some target σ. It appears that the first 12 eigenvalues of Problem III are easy to find within 100 iterations, whereas 300 iterations are not sufficient to get 15 of them.

The sixth column displays the time spent by solving the eigenvalue problem on the projected system. For an equal number of iterations, this time should be independent of the number of processors involved. However, we found deviating values for $p = 20$ and $p = 32$ in the case of Problem II. We observe that the contribution of the

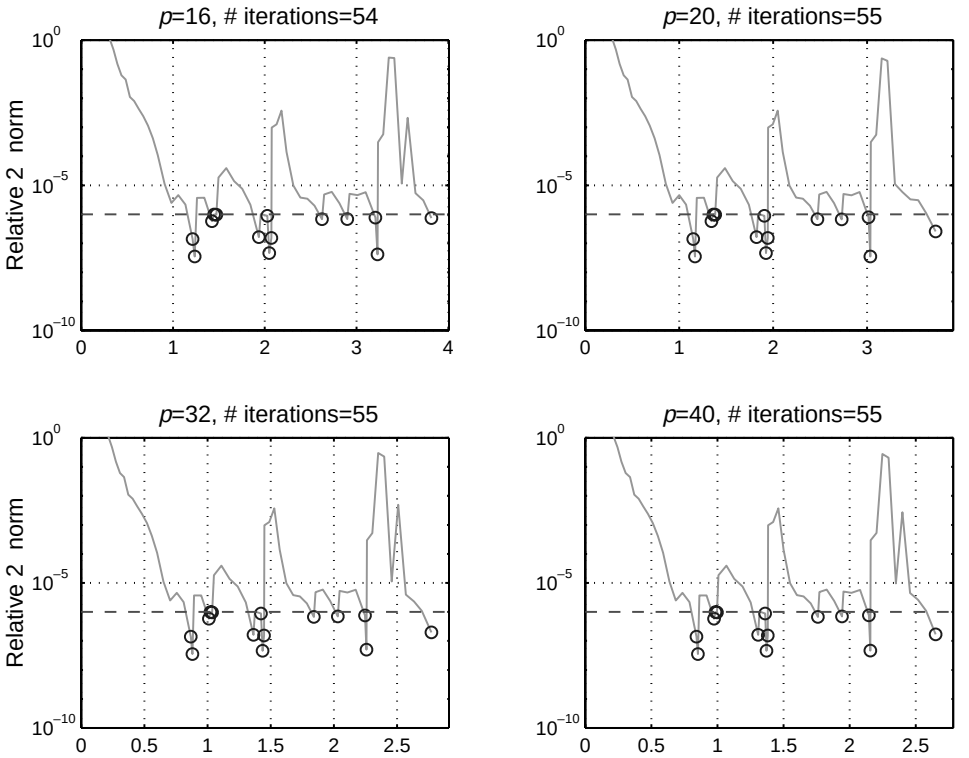


FIG. 5. *The relative* 2*-norm of the residual* (7) *or* (8) *against the wall clock times (in seconds) of Problem* II *(first* 14 *eigenpairs) obtained on* $p = 16, 20, 32, 40$ *processors of the Cray T3E.*

sequential part to the total wall clock time is small, especially for large $\lceil \frac{N}{p} \rceil$.

The seventh column shows the total time spent in performing the triangular solves. Again the numbers in parentheses show the Mflop rates, which have been calculated by using the estimate (15) of the number of multiplications. The level-2 `BLAS` routines used for the triangular solves are significantly slower than the level-3 `BLAS` routines used for the construction of L and U, because the ratio of computations and memory-to-processor data transfer is much more favorable for level-3 `BLAS` than for level-2 `BLAS`. Therefore, the Mflop rates are significantly lower than those obtained for the construction of L and U. The triangular solves in combination with one matrix-vector multiplication with the matrix B form the expensive application of the operator $Q = (LU)^{-1}B$.

In general, the convergence history hardly depends on the number of processors. This is illustrated by Figure 5, which shows the history of Problem II for the first 14 eigenpairs on 16, 20, 32, and 40 processors. In all cases, the"same" eigenvalues were found in the same order. The distance between the "same" eigenvalues is about 10^{-6} as we might expect, since $tol_{JD} = 10^{-6}$.

5.3. Analysis of the speed-ups. The computations in the JD algorithm can be divided into three parts: the solution of the (small) projected eigenvalue problems, the triangular solves with L and U, and a part consisting of matrix-vector multiplications with B and $A-\sigma B$, inner products and vector updates. The CPU times per processor for these parts are denoted by $t_{seq}^{(p)}$, $t_{LU}^{(p)}$, and $t_{lp}^{(p)}$, respectively; in case p processors are

used for execution. If we assume that the time for communication can be neglected and the inner products scale linearly, the expected speed-up $S_{JD}^{(p)}$ on p processors is

$$S_{JD}^{(p)} = \frac{t_{seq}^{(1)} + t_{LU}^{(1)} + t_{lp}^{(1)}}{t_{seq}^{(p)} + \frac{t_{LU}^{(p)}}{S_{LU}^{(p)}} + \frac{t_{lp}^{(p)}}{p}}, \tag{19}$$

in which $S_{LU}^{(p)}$ is the expected speed-up for the triangular solves. In the following, we will show that an approximation of $S_{LU}^{(p)}$, $p > 1$, is given by ($N_p = \lceil \frac{N}{p} \rceil$)

$$\begin{aligned} S_{LU}^{(p)} &\approx \frac{3pN}{(5-\frac{2}{p})(N+1-p)+5p\lceil \log_2 p - 1 \rceil} && \text{for } N_p > 1, \\ S_{LU}^{(p)} &\approx \frac{3p}{5\lfloor \log_2 p \rfloor} && \text{for } N_p = 1. \end{aligned} \tag{20}$$

If the JD code executes on one processor, the sequential block-tridiagonal LU approach is applied and in that case the number of multiplications in the triangular solves is approximately $3Nn^2$. Almost every multiplication can be combined with one addition, hence the wall clock time required on one processor is approximately $3Nn^2 t_{mul}$, in which t_{mul} is the time required for performing a complex multiplication combined with an addition. On p processors, the parallel direct solver `(SOL)DDCR` is applied. The first part of the triangular solves corresponds to the solution of $N - p + 1$ block-tridiagonal systems, which scales linearly. The number of multiplications in this part is approximately $3(N - p + 1)n^2\omega$, in which ω is the number of multiplications in the `DDCR` approach divided by the number of multiplications in the block-tridiagonal LU approach. From (15) it follows that $\omega \approx (5 - \frac{2}{p})/3$. Hence the wall clock time required for the domain decomposition part is approximately $(3\omega(N-p+1)n^2 t_{mul})/p$.

The second part deals with cyclic reduction. The wall clock time required for this part is approximately $5n^2 t_{mul}$ multiplied by the number of steps in cyclic reduction. In case $N_p > 1$, $p - 1$ processors perform the cyclic reduction in $\lceil \log_2 p - 1 \rceil$ steps. Otherwise, in the case of pure cyclic reduction ($N_p = 1$), the process is performed on p processors in $\lfloor \log_2 p \rfloor$ steps. The expressions in (20) are obtained by dividing the approximate wall clock time on one processor by the sum of the wall clock times required for the first and second part of the triangular solves on p processors.

Figure 6 shows both the predicted speed-ups by (19) and (20) and the measured speed-ups obtained from Table 6. Since this table does not contain the information for $p = 1$ for Problems II and III, the execution time must be approximated by, for instance,

$$T^{(1)} = t_{seq}^{(p)} + S_{LU}^{(p)} \times t_{LU}^{(p)} + p \times t_{lp}^{(p)} \quad \text{for both } p = 2 \text{ (Problem II) and } p = 10 \text{ (Problem III).} \tag{21}$$

The large variation in the number of JD steps has been balanced out by dividing $t_{seq}^{(p)}$, $t_{LU}^{(p)}$, and $t_{lp}^{(p)}$ by those numbers. For Problem I the predicted speed-up $S_{JD}^{(p)}$ is a little too high for p large compared with the measured speed-up. This is caused by the fact that the simple prediction model assumes that the inner products scale linearly, which is not quite true for small vectors.

Notice that the speed-ups become higher when the problem size grows, which could be expected. If we increase the order of A and B, $t_{seq}^{(p)}$ hardly changes, when we keep the order of the projected systems fixed. Accordingly, the communication time due to inner products remains equal. However, the computational work expressed by $t_{LU}^{(p)}$ and $t_{lp}^{(p)}$ will increase with the order of the eigenvalue problem.

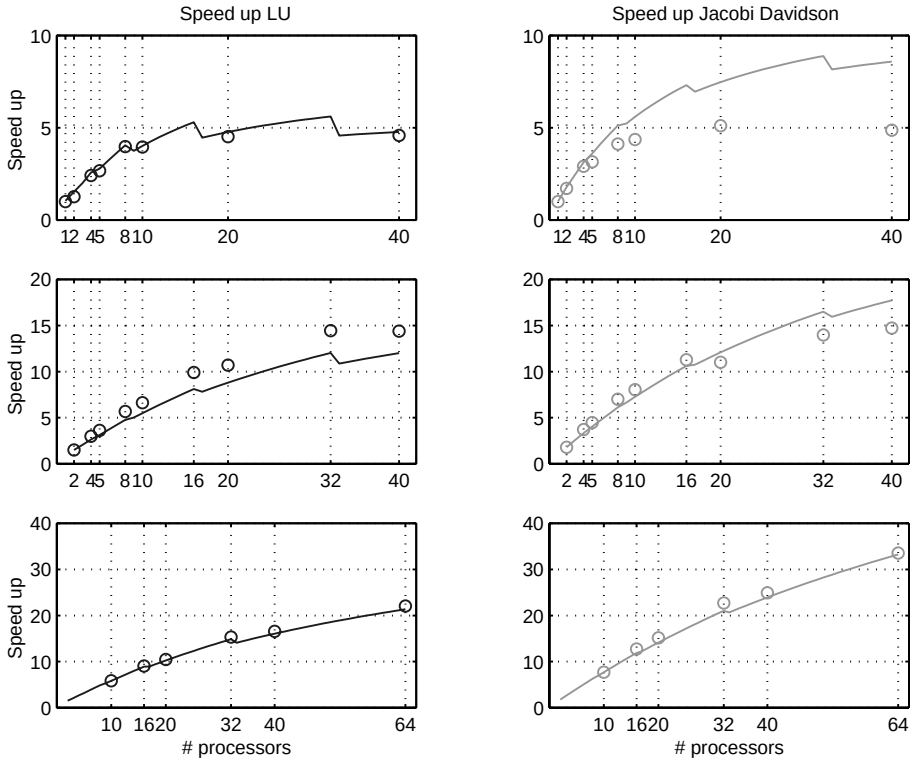


FIG. 6. *The predicted (solid line) and measured speed–up ("○") of the block-tridiagonal LU factorization and the JD algorithm for Problems* I, II, *and* III, *respectively.*

6. Conclusions. We have studied the JD method for the parallel computation of a few selected eigenvalues of large generalized eigenvalue problems arising in the stability investigation of tokamak plasmas. The method has been combined successfully with a parallel complete block-tridiagonal LU decomposition, which appears to be robust because pivoting is used. However, in order not to disturb the block structure of the matrix, the search for pivot elements is restricted to the blocks on the main diagonal. Numerical experiments performed on a Cray T3E demonstrate that this method parallelizes well.

Most other ingredients in the JD method like the matrix-vector multiplications, vector updates and inner products parallelize very well. Only the construction and solution of the small projected eigenvalue problems in the JD method do not parallelize. However, the order of these systems is kept small and is independent of the problem size. Hence for large applications, the total time spent in solving the projected systems is small compared with the time spent in the parallel parts of the method.

We observe that for large problems the measured speed-up corresponds quite well with the prediction. More important is that the speed-up for large problems is much higher than for smaller ones; therefore we may conclude that the JD method is very well suited for parallel execution.

The main reason to study implementations on distributed memory machines, like the Cray T3E for large eigenvalue problems, is the memory bound of shared memory

machines. If n and N are large ($n \gg m$ and $N \gg p$), then our parallel implementation of JD requires approximately $128Nn^2$ bytes of memory. Our sequential implementation requires approximately $92Nn^2$ bytes. The total amount of memory of the Cray T3E at HPαC, Delft, is only slightly larger than the 8 Gbytes of main memory of the Dutch National supercomputer with shared memory, a Cray C90 with 12 CPUs at SARA, Amsterdam. Hence on the current Delft configuration, we can solve problems of approximately the same size on the Cray C90. Recently, we got the opportunity to execute our code on a 512 processor Cray T3E, which enables us to examine larger eigenvalue problems (see also [12]).

We remark that the JD method is applicable for generalized eigenvalue problems $Ax = \lambda Bx$ in which the action of the inverse of A, B, or $A - \sigma B$ for a target σ is not available. The projections included in the correction equation guarantee a proper update of an approximate eigenvector in the "right" direction. When we started our research, we hoped to be able to exploit the sparsity pattern within the block-tridiagonal structure of $A - \sigma B$, by using an *incomplete* LU decomposition as a preconditioner for `GMRES(m)` to solve the correction equation approximately. If an *incomplete* decomposition would have been used, the transformation described by (2) would not have been possible and in that case the application of JD applied to the *generalized* eigenvalue problem would probably have been more efficient than Arnoldi's method. However, it turned out to be more efficient to construct a *complete* block-tridiagonal LU decomposition of $A - \sigma B$, because the block tridiagonal structure of this matrix can be exploited. Moreover, since the decomposition is performed on a block level, `BLAS` routines can be used, which guarantees an efficient implementation per processor. We therefore end up with the *standard* eigenvalue problem (2) in which the spectrum of interest is also the dominant part of the spectrum.

For this special eigenvalue problem it is not efficient to use several `GMRES` steps for the correction equation that appears in the JD method. Four variants have been examined and the best results were obtained when the iteration process starts with some Arnoldi-like steps, until the residual is sufficiently small. The process can be accelerated by applying a (cheap) preconditioner within `GMRES(m)`. However, a reduction in the number of Jacobi–Davidson steps does not guarantee a reduction in execution time. From numerical experiments, it appears that the best choice for minimizing the wall clock time is to take $\tilde{z} = r$ in step 4 of the algorithm in Figure 2. If we should use a (very) good preconditioner and solve the correction equation exactly, then the convergence to the next eigenvector will be quadratic (see [15, Theorem 3.2]). For this special case, it is probably more efficient to use Arnoldi's method than JD. However, the difference in efficiency will not be large, because the most expensive operation in both JD and Arnoldi's method is the matrix-vector multiplication with Q in (2) and both methods need approximately the same number of matrix-vector multiplications.

Acknowledgments. The authors wish to thank Herman te Riele for many stimulating discussions and numerous suggestions for improving the presentation of the paper. They also thank Gerard Sleijpen and Jos van Dorsselaer for carefully reading the paper and suggesting several improvements. We also appreciate the helpful comments by the referees very much.

Further they would like to thank Ronald van Pelt of Cray Research, the Netherlands, for his programming advice for both the Cray T3D and Cray T3E, and HPαC (Delft) and SARA (Amsterdam) for their technical support. They gratefully acknowledge Cray Research for a sponsored account on the Cray T3D, and the Dutch National Computing Facilities Foundation NCF for the provision of computer time on the Cray

C90 and the Cray T3E.

REFERENCES

[1] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J.J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammerling, A. McKenney, S. Ostrouchov, and D. Sorensen. *LAPACK Users' Guide*, 2nd ed., SIAM, Philadelphia, 1995.

[2] R. Barrett, M. Berry, T. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. van der Vorst, *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*, SIAM, Philadelphia, 1994; also available online from http://www.netlib.org/templates/.

[3] J.G.C. Booten and H.A. van der Vorst, *Cracking large-scale eigenvalue computations, part* I*: Algorithms*, Computers in Physics, 10 (1996), pp. 239–242.

[4] J.G.C. Booten and H.A. van der Vorst, *Cracking large-scale eigenvalue computations, Part* II*: Implementations*, Computers in Physics, 10 (1996), pp. 331–334.

[5] J.G.C. Booten, D.R. Fokkema, G.L.G. Sleijpen, and H.A. van der Vorst, *Jacobi-Davidson methods for generalized MHD-eigenvalue problems*, Z. Angew. Math. Mech. 76, Supplement 1 (1996), pp. 131–134.

[6] D.R. Fokkema, G.L.G. Sleijpen, and H.A. van der Vorst, *Jacobi-Davidson style QR and QZ algorithms for the reduction of matrix pencils*, SIAM J. Sci. Comput., 20 (1998), pp. 94–125.

[7] G.H. Golub and C.F. Van Loan, *Matrix Computations*, 3rd ed., The Johns Hopkins University Press, Baltimore, MD, 1996.

[8] W. Kerner, J.P. Goedbloed, G.T.A. Huysmans, S. Poedts, and E. Schwartz, *CASTOR: Normal-mode analysis of resistive MHD plasmas*, J. Comput. Physics, 142 (1998), pp. 271–303.

[9] W. Kerner, S. Poedts, J.P. Goedbloed, G.T.A. Huysmans, B. Keegan, and E. Schwartz, *Computing the damping and destabilization of global Alfvén waves in tokamaks*, in Proceedings of 18th Conference on Controlled Fusion and Plasma Physics, Vol. IV, P. Bachman and D. C. Robinson, eds., EPS, Berlin, 1991, pp. 89–92.

[10] R.B. Lehoucq, D.C. Sorensen, and C. Yang, *ARPACK Users' Guide, Solution of Large-Scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods*, SIAM, Philadelphia, 1998.

[11] M. Nool and A. van der Ploeg, *A Parallel Jacobi-Davidson Method for Solving Generalized Eigenvalue Problems in Linear Magnetohydrodynamics*, Technical Report NM–R9733, CWI, Amsterdam, 1997.

[12] M. Nool and A. van der Ploeg, *Parallel Jacobi-Davidson for solving generalized eigenvalue problems*, in Proceedings of the Third International Meeting on Vector and Parallel Processing, Lecture Notes in Comput. Sci. 1573, J.M.L.M. Palma, J. Dongarra, and V. Hernandez, eds., Springer-Verlag, Berlin, 1999, pp. 58–71.

[13] S. Poedts, W. Kerner, J.P. Goedbloed, B. Keegan, G.T.A. Huysmans, and E. Schwartz, *Damping of global Alfvén waves in tokamaks due to resonant absorption*, Plasma Physics and Controlled Fusion, 34 (1992), pp. 1397–1422.

[14] Y. Saad and M.H. Schultz, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 17 (1986), pp. 856–869.

[15] G.L.G. Sleijpen, J.G.L. Booten, D.R. Fokkema, and H.A. van der Vorst, *Jacobi-Davidson type methods for generalized eigenproblems and polynomial eigenproblems*, BIT, 36 (1996), pp. 595–633.

[16] G.L.G Sleijpen and H.A. van der Vorst, *A Jacobi-Davidson iteration method for linear eigenvalue problems*, SIAM J. Matrix Anal. Appl., 17 (1996), pp. 401–425.

[17] B. van der Holst, A.J.C. Beliën, J.P. Goedbloed, M. Nool, and A. van der Ploeg, *Calculation of resistive magnetohydrodynamics spectra in tokamaks*, Physics of Plasmas, 6 (1999), pp. 1554–1561.

[18] A. van der Ploeg, *Reordering Strategies and LU-Decomposition of Block Tridiagonal Matrices for Parallel Processing*, Technical Report NM–R9618, CWI, Amsterdam, 1996.

SIAM J. SCI. COMPUT.
Vol. 22, No. 1, pp. 113–151
© 2000 Society for Industrial and Applied Mathematics

TOWARD FRONT TRACKING BASED ON CONSERVATION IN TWO SPACE DIMENSIONS *

MAO DE-KANG†

Abstract. A two-dimensional front tracking method based on conservation for scalar conservation law and for the Euler system of gas dynamics is being developed. In the method, discontinuities are tracked by enforcing the conservation properties of the PDEs. Unlike the traditional front tracking methods, for which the conservation is only a property that may or may not be preserved, in this front tracking method the conservation is the mechanism by which the tracking is performed. The method is also extended to treat the reflection wall boundary in the Euler system. As an attempt to treat the interactions of discontinuities by enforcing the conservation properties without solving two-dimensional Riemann problems, three special cases of treatment of discontinuity interactions in scalar conservation law and in the Euler system of gas dynamics are studied. Finally, numerical experiments are implemented in both scalar and system cases to show the efficiency of the method.

Key words. front tracking, cell-average, recovery of discontinuity curve

AMS subject classifications. 65M05, 76L05, 35L65.

PII. S1064827597310609

1. Introduction.

In this paper we are concerned with the nonlinear hyperbolic system of conservation laws

$$u_t + \sum_{i=1}^{m} f_i(u)_x = 0, \tag{1.1}$$

where u and $f(u)$ can be either scalars or vectors and $m = 1, 2$. It is well known that solutions to (1.1) can develop discontinuities, no matter how smoothly the initial and boundary conditions are proposed. It is also well known that in most cases the interior structure of discontinuities is of little interest, but the effect of the discontinuity relations (the Rankine–Hugoniot conditions) on the hydrodynamic scale flow is important for applications. On these two observations the so-called front tracking methods have been developed, which treat discontinuities as interior moving boundaries coupled to finite difference calculation for the smooth part of solution.

One of the important features of front tracking methods is the usage of lower dimensional grids, called fronts, which fit the discontinuities in the numerical solutions. In regions that are surrounded by fronts, the numerical solutions to (1.1) are solved separately using methods designed for smooth solutions. The propagation of the fronts and the updating of the states on their two sides are performed by the requirement that the jump conditions be satisfied across the fronts in some way.

An approach that is widely accepted for advancing fronts is to compute the propagation speed of the fronts in every time step. This can essentially be done by solving the Riemann problems between the states on the two sides of the fronts. Once the propagation speed is obtained, the discontinuity locations at the new time can be

*Received by the editors October 14, 1996; accepted for publication (in revised form) January 15, 1999; published electronically June 13, 2000.
http://www.siam.org/journals/sisc/22-1/31060.html
†Department of Mathematics, Shanghai University, Shanghai 200436, China (dkmao@guomai.sh.cn). The research of this author was supported by China NSF grant 19671056 and Research Foundation of Shanghai University of Science and Technology grant 445.

computed, and then the front tracking can be carried on to the next time step. In doing so, the conservation properties of the system (1.1) is generally not preserved.

In the last three decades, a great number of front tracking methods of this type have been developed; see [1], [9], [10], [11], [12], [13], [14], [18], [32], [35], [39], and the references cited therein. Here we call the front tracking methods of this type the traditional front tracking methods. The traditional front tracking methods have been proved successful in application in the sense that high accuracy and resolution can be achieved for both solutions in smooth regions and discontinuity locations on significantly coarse grids. However, the price for high accuracy and resolution is the complexity of the algorithm.

We would particularly mention the contributions of Glimm and his coworkers to the traditional front tracking. In their systematic development of front tracking methods they have been building a very extensive set of tools to deal with the complexity of the algorithm so that the high accuracy and resolution on coarse grids can be obtained even for some problems with geometrically and toplogically complex fronts. However, their algorithm is really expensive.

The reason for the complexity of the algorithm is not difficult to see. The basis of the traditional front tracking methods is the assumption that the solution to be solved is piecewise smooth. The high accuracy and resolution of the methods rely heavily on the fact that the locations of fronts are accurately computed in every time step. If an error of large magnitude occurs at a time, there is no mechanism in the tracking methods to delete it in the later computation. However, the accurate computation of front locations will become difficult when solutions have geometrically and topologically complex fronts, and the assumption of piecewise smoothness becomes questionable. At this moment, much work of the algorithm must be spent on resolving the geometrical and topological configuration of the fronts for obtaining high accuracy, and often knowledge of the exact solutions is invested for this purpose.

For many years the author has been working on a different approach of front tracking in which discontinuities are advanced by enforcing the conservation properties of (1.1) rather than by computing their propagation speed. The philosophy of the work comes from, in addition to the above observation on the traditional front tracking methods, the following two observations on front capturing methods.

(1) In front capturing methods the conservation properties insure the correct propagation speed of discontinuities in numerical solutions. Capturing difference schemes are always designed in the conservative form so that when solutions are smooth the scheme is consistent with (1.1), and when discontinuities are involved and there are errors of large magnitude, the errors will be controlled by the conservation properties so that the discontinuity locations are still correct even though the consistency does not hold any more [23].

(2) Harten's subcell resolution [15] shows that the conservation properties can actually be used to locate discontinuities within grid cells.

These observations motivate us to develop a front tracking method which uses the conservation properties of (1.1) as its tracking mechanism. It is expected that the correct front locations can be guaranteed by enforcing the conservation properties. When fronts are simple, the method is consistent with (1.1) in smooth regions and its jump conditions on the fronts, and when the fronts are complex and errors of large magnitude occur, the errors can be controlled by conservation and can be deleted when the complex fronts are gone. Thus, the requirement that the locations of fronts

be accurately computed in every time step can be relaxed, the complexity of the algorithm can be greatly reduced, and the algorithm itself will be much more robust.

This approach has been proved successful in one space dimension. The one-dimensional front tracking method of this type developed in [25], [27], [28], [29], and [30] shows the following advantages over the traditional front tracking methods:

(1) The method can be applied to any front capturing schemes and it works as an adjustment of the schemes near discontinuities. Therefore, the algorithm can be easily coded in a nearly front capturing fashion.

(2) The algorithm is implemented on uniform Cartesian grids; thus, we avoid the use of adaptive grids and get rid of the related small-cell problem that troubles all the traditional front tracking methods.

(3) The method preserves the conservation of the solution; therefore, the handling of discontinuity collisions and formations is quite simple and robust. Due to a so-called "stacking technique" described in section 4 in [30], the handling is also very detail-resolving.

Some studies have been done on the accuracy, stability, and entropy satisfaction of the method in the one-dimensional scalar case. In [25] a study was made on the accuracy of the discontinuity position when the numerical solution was piecewise smooth, and it was shown that when the underlying scheme was of rth order and the reconstruction of the numerical solution in the critical cell was of $(r-1)$ order, the discontinuity position was rth order accurate. In [19], a total variation diminishing (TVD) study was made for the method and it was shown that when the underlying scheme was TVD and the time step was restricted by the Courant–Friedrichs–Lewy (CFL) condition then the whole algorithm was TVD. In [31], a study was made on the entropy satisfaction of the method and it was shown that when the underlying scheme satisfied the entropy condition and some criterions were also satisfied for the tracked fronts, the numerical solution satisfied the entropy condition.

We are going to extend this conservative front tracking method in two space dimensions, and for this purpose we plan to write a series of papers. This paper, the first one of the series, is mainly devoted to presenting the basic ideas by describing the tracking of single discontinuity curves (with some restriction in topology). Some special cases of the treatment of discontinuity interactions are studied in this paper. The systematic treatment of discontinuity interactions and some other issues of the two-dimensional front tracking method are still in development and will appear in our future papers.

The main difficulty in extending the front tracking method in two space dimensions lies in the recovery of discontinuity curves out of the cell-averages of the numerical solution in critical cells, the cells that are cut through by the discontinuity curves (see section 3). In the one-dimensional case the discontinuity positions are recovered by the requirement that the solution be conserved in the critical cells. However, in the two-dimensional case the discontinuity in a critical cell is represented by a segment of curve, the recovery of which, conceptually speaking, needs an infinite amount of information. The requirement of conservation in the critical cell provides only one equation, which is not enough for the recovery.

A recovery procedure is developed in section 3, which is close in spirit to the "reconstruction via primitive function" described in [15]. The procedure uses the cell-averages in the critical cell as well as the cell-averages in the nearby critical cells cut through by the same discontinuity to recover the segment of curve. Upon this recovery procedure a front advancing algorithm is built. In section 3 we will show

that this front advancing algorithm is more substantial, natural, and much simpler than that of the traditional front tracking methods.

There are a few other front tracking methods that also relate to the conservation properties of (1.1). Chern and Colella [2] use a "flux redistribution" algorithm to maintain the conservation in their front tracking method. The "flux differences" produced in advancing the fronts are distributed into nearby cells based on the characteristic aspects in the vicinity of the tracked discontinuity.

LeVeque and Shyue [20], [21] use a wave propagation method to track discontinuities. The cell-averages in both regular and irregular grid cells are updated by computing the waves propagating in and out of the cells during the time step so that the conservation of the solution is maintained.

However, in those front tracking methods, the conservation is only considered as an important property that should be preserved, rather than used as the mechanism of tracking as in the method presented in this paper. The front locations are still computed by computing the propagation speed from the solutions to the Riemann problems.

Therefore, the front locations do not get as much feedback from the conservation as in this method, and the methods are more complex.

In his early paper [26] the author also presented an extension of the tracking method in two space dimensions. However, the method described in that paper is still very much of the traditional type in which even the conservation is not preserved. The front locations are still computed by computing the propagation speed. The difference is that the propagation speed along the horizontal or vertical grid lines is computed, instead of the normal propagation speed as in the other traditional front tracking methods mentioned above. The method is much more complex than the present one.

The paper is organized in the following way: section 2 is a brief recall of the tracking method in one space dimension. Section 3 presents the two-dimensional tracking method for the scalar conservation law. The method is presented in the semidiscrete form because the direct full-discretization of the method will be complicated. Section 4 extends the tracking method to the Euler system of gas dynamics in two space dimensions. Section 5 uses the idea of the tracking method to develop a treatment of reflection wall boundary in the Euler system of gas dynamics. Section 6 presents the treatments of three special cases of discontinuity interactions on the observation of conservation without solving two-dimensional Riemann problems. Section 7 displays two numerical examples in the scalar conservation law and one numerical example of regular shock reflection in the Euler system of gas dynamics for the tracking method.

2. A brief recall of the tracking method in one space dimension. In this section we briefly recall the tracking method in the one-dimensional case. In our early paper we presented the method in the context of conservation error (see [25]); however, in this paper we present the method in the context of cell-average, considering that the later presentation is more natural and convenient.

We are concerned with the tracking method for the hyperbolic system of conservation laws

$$u_t + f(u)_x = 0, \tag{2.1}$$

where $u : R \times R \to R^m$ is an m-dimensional vector of conserved quantities, and $f(u)$ is a vector-valued function of u, called flux function for the system.

We begin with the scalar case in which both u and f are scalar. Assume that the

underlying difference scheme

$$u_j^{n+1} = u_j^n - \lambda(\hat{f}_{j+1/2}^n - \hat{f}_{j-1/2}^n) \tag{2.2}$$

is of Godunov type, where u_j^n is a cell-average approximation to the solution, $\hat{f}_{j+1/2}^n$ is a flux average approximation to $f(u)$ on the cell boundary, and $\lambda = \tau/h$ is the mesh ratio with τ and h being the time and space increments of the grid, respectively. As is well known, the numerical flux in the Godunov-type scheme is computed as follows: On each time level a piecewise polynomial function $\tilde{u}^n(x, t_n)$ is reconstructed out of the cell-averages, which is a polynomial of the same order in each grid cell $[x_{j-1/2}, x_{j+1/2}]$. The reconstruction maintains the conservation in the sense that in each cell

$$\frac{1}{h}\int_{x_{j-1/2}}^{x_{j+1/2}} \tilde{u}^n(x, t_n)dx = u_j^n. \tag{2.3}$$

We solve (2.1) (approximately at most time) with $\tilde{u}^n(x, t_n)$ as the initial data to obtain the solution $\tilde{u}^n(x, t)$, and then the numerical flux is computed as

$$\hat{f}_{j+1/2}^n = \frac{1}{\tau}\int_{t_n}^{t_{n+1}} f(\tilde{u}^n(x_{j+1/2}, t))dt. \tag{2.4}$$

Constructed in this way, the scheme is of $(2k + 1)$-points; i.e, the numerical flux is a function of $2k$ variables

$$\hat{f}_{j+1/2}^n = \hat{f}(u_{j-k+1}^n, \ldots, u_{j+k}^n), \tag{2.5}$$

where k is a number related to the order of the polynomial in each cell, and is consistent with the flux in (2.1) in the sense that $\hat{f}(u, \ldots, u) = f(u)$; see [23].

Assume that the cell $[x_{j_1-1/2}, x_{j_1+1/2}]$ harbors a discontinuity at t_n, which is called a critical cell. In the critical cell, in addition to the ordinary cell-average $u_{j_1}^n$, we define two more cell-averages $u_{j_1}^{n,-}$ and $u_{j_1}^{n,+}$, which are computed using information only from the left and right sides of the discontinuity, respectively. The discontinuity position can then be recovered from these cell-averages by enforcing the conservation property and the recovery will be described later.

Now we describe the step of computation from t_n to t_{n+1}. First, we extend the cell-averages on the two sides of the discontinuity by extrapolation to the other sides and obtain $u_{j_1+1}^{n,-}, \ldots, u_{j+k}^{n,-}$ and $u_{j-k}^{n,+}, \ldots, u_{j_1-1}^{n,+}$, where the data with "$-$" are from the left to the right and the data with "$+$" are from the right to the left. Then the numerical solution on the next time level are computed on the left and right sides as

$$u_j^{n+1} = u_j^n - \lambda(\hat{f}_{j+1/2}^{n,-} - \hat{f}_{j-1/2}^{n,-}) \tag{2.6}$$

for $j < j_1$, and

$$u_j^{n+1} = u_j^n - \lambda(\hat{f}_{j+1/2}^{n,+} - \hat{f}_{j-1/2}^{n,+}) \tag{2.7}$$

for $j > j_1$, respectively, where $\hat{f}_{j+1/2}^{n,-}$ and $\hat{f}_{j+1/2}^{n,+}$ are defined as

$$\hat{f}_{j+1/2}^{n,-} = \hat{f}(u_{j-k+1}^n, \ldots, u_{j_1-1}^n, u_{j_1}^{n,-}, \ldots, u_{j+k}^{n,-}) \tag{2.8}$$

and

$$\hat{f}^{n,+}_{j+1/2} = \hat{f}(u^{n,+}_{j-k+1}, \ldots, u^{n,+}_{j_1}, u^{n}_{j_1+1}, \ldots, u^{n}_{j+k}). \tag{2.9}$$

In the critical cell, the cell-averages $u^{n+1,-}_{j_1}$ and $u^{n+1,+}_{j_1}$ are computed from $u^{n,-}_{j}$ and $u^{n,+}_{j}$ as in (2.6) and (2.7). In doing so, we let the computation on each side of the discontinuity use information only from the same side. However, the cell-average $u^{n+1}_{j_1}$ is computed as

$$u^{n+1}_{j_1} = u^{n}_{j_1} - \lambda(\hat{f}^{n,+}_{j_1+1/2} - \hat{f}^{n,-}_{j_1-1/2}) \tag{2.10}$$

under the consideration that the two cell boundaries are on the two sides of the discontinuity.

We see that in the algorithm extrapolated data from one side are widely used, which is an another important feature of the method. A problem that is faced by the front tracking is that when near the fronts there will never be enough regular cells on one side to carry out the computation by the underlying scheme. Most tracking methods solve this problem by introducing adaptive grids and define cell-averages of the numerical solution on the irregular cells; however, complexities follow. In our method, the extrapolation provides all the data necessary for carrying out the computation by the underlying scheme on one side; in doing so, we avoid the complexity of adaptive grid.

We now reconstruct a piecewise polynomial function, still denote by $\tilde{u}^{n+1}(x, t_{n+1})$, out of the cell-averages at t_{n+1}. In ordinary cells, it is reconstructed by the reconstruction procedure of the underlying Godunov type scheme, only the extrapolated data from the same sides should be used when the cells are near the discontinuity. In the critical cell, we reconstruct $\tilde{u}^{n+1,-}(x, t_{n+1})$ and $\tilde{u}^{n+1,+}(x, t_{n+1})$ out of the cell-averages on the left and right sides of the discontinuity and their extrapolated data on the other sides, respectively. We then consider $\tilde{u}^{n+1}(x, t_{n+1})$ in the critical cell to be a piecewise polynomial function of the form

$$\tilde{u}^{n+1}(x, t_{n+1}) = \begin{cases} \tilde{u}^{n+1,-}(x, t_{n+1}), & x < \xi^{n+1}; \\ \tilde{u}^{n+1,+}(x, t_{n+1}), & x \geq \xi^{n+1}, \end{cases} \tag{2.11}$$

where ξ^{n+1} is the discontinuity position at t_{n+1}. Then ξ^{n+1} can be determined from

$$\begin{aligned} \frac{1}{h}\int_{x_{j_1-1/2}}^{x_{j_1+1/2}} \tilde{u}^{n+1}(x, t_{n+1})dx = \frac{1}{h}\Bigg(&\int_{x_{j_1-1/2}}^{\xi^{n+1}} \tilde{u}^{n+1,-}(x, t_{n+1})dx \\ &+ \int_{\xi^{n+1}}^{x_{j_1+1/2}} \tilde{u}^{n+1,+}(x, t_{n+1})dx\Bigg) = u^{n+1}_{j_1}, \end{aligned} \tag{2.12}$$

which comes from the requirement of conservation in the critical cell. In particular, when the underlying scheme is Godunov and the reconstruction is piecewise constant, the solution to (2.12) is simply

$$\xi^{n+1} = x_{j_1-1/2} + h\frac{u^{n+1}_{j_1} - u^{n+1,+}_{j_1}}{u^{n+1,-}_{j_1} - u^{n+1,+}_{j_1}}. \tag{2.13}$$

Once the discontinuity position is obtained, the critical cell at t_{n+1} can then be determined. If $[x_{j_1-1/2}, x_{j_1+1/2}]$ is still the critical cell, the numerical solution on the

new level will not be changed. Otherwise, one of the two adjacent cells will become the critical cell and the numerical solution in the related cells should be adjusted accordingly. For example, if the discontinuity position moves to the left adjacent grid cell at t_{n+1}, then the left adjacent cell is the critical cell at the new time. Meanwhile the numerical solution should be changed as follows:

$$\begin{aligned} u_{j_1-1}^{n+1,-} &:= u_{j_1-1}^{n+1}, \\ u_{j_1-1}^{n+1,+} &:= u_{j_1-1}^{n+1,+}, \\ u_{j_1-1}^{n+1} &:= u_{j_1-1}^{n+1} + u_{j_1}^{n+1} - u_{j_1}^{n+1,+}, \\ u_{j_1}^{n+1} &:= u_{j_1}^{n+1,+}, \end{aligned} \tag{2.14}$$

where $u_{j_1-1}^{n+1,+}$ on the right side is the extrapolated data of the cell-averages from the right in this cell. In this change, the numerical solution on the right is extended in the new critical cell in a piecewise polynomial fashion and the sum $u_{j_1-1}^{n+1} + u_{j_1}^{n+1}$ is conserved. The case that the discontinuity position moves to the right can be treated analogously.

Thus, a step of the tracking method is completed.

When (2.1) is a system, in analogy with the scalar case, we define the cells that harbor a discontinuity to be the critical cells. However, in this case, (2.1) has m different kinds of characteristics and, therefore, has m different kinds of discontinuities. Hence, the critical cells are classified in m kinds according to the types of the discontinuities they harbor. For example, if (2.1) is the Euler system of gas dynamics, (see, e.g., [3] and [23]), we say a critical cell $[x_{j_1-1/2}, x_{j_1+1/2}]$ is a left shock critical cell, denoted by LSCC, if the solution to the Riemann problem $R(u_{j_1}^{n,-}, u_{j_1}^{n,+})$ has a strong left shock, where $R(u_l, u_r)$ denotes the Riemann problem with u_l and u_r as its left and right states. The right shock critical cell, denoted by RSCC, and the contact discontinuity critical cell, denoted by CDCC, can be defined analogously.

The computation in critical cells is still implemented as in the scalar case, only we are now dealing with vectors rather than scalars. However, the recovery of discontinuity positions and the reconstruction of the solution in critical cells are more complex than those in the scalar case, in which the boundary conditions on the tracked discontinuities are to be implemented. As one can see, the conserved quantities are exchanged through any interfaces by means of propagation of waves corresponding to different characteristics. The boundary conditions on a discontinuity interface represent, physically speaking, the waves that correspond to the characteristics entering the region through the interface. Therefore, in the system case we should separate the waves associated with different characteristic fields in the critical cell so that the waves associated with the characteristic fields other than those of the tracked discontinuity can enter in the corresponding regions.

We use the Euler system as an example to describe the recovery and reconstruction. The recovery and reconstruction to be presented here are only piecewise constant; therefore, they are only first-order. A high-order piecewise polynomial version of recovery and reconstruction is still in development.

As in the scalar case, we compute the left and right cell-averages $u_{j_1}^{n+1,-}$ and $u_{j_1}^{n+1,+}$ at t_{n+1}. We solve the Riemann problem $R(u_{j_1}^{n+1,-}, u_{j_1}^{n+1,+})$. As is well known, the solution to the Riemann problem is $u_{j_1}^{n+1,-}$ far enough to the left and is $u_{j_1}^{n+1,+}$ far enough to the right, and between them there are left and right centered waves, either shocks or rarefaction waves, a contact discontinuity, and two constant states,

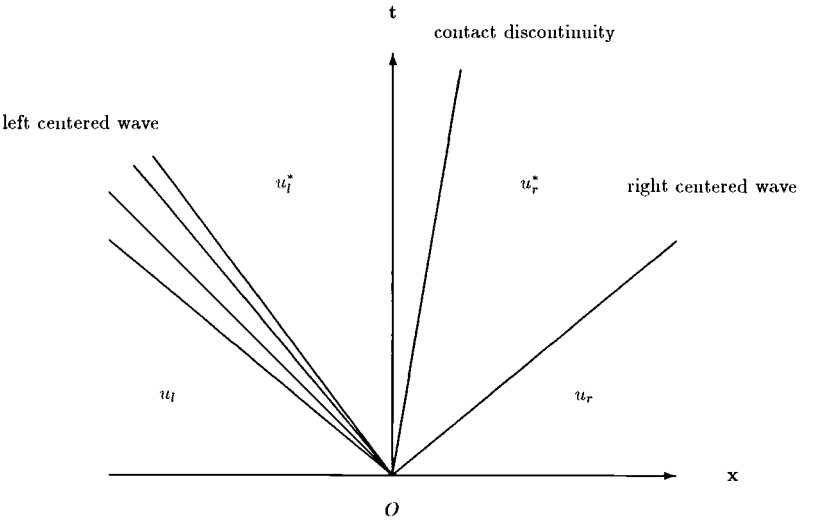


FIG. 1. *Solution to a Riemann problem. On the far left is u_l and on the far right is u_r, between them are two centered waves, either shock or rarefaction waves, a contact discontinuity, and two intermediate states u_l^* and u_r^* to connect u_l and u_r.*

denoted by u_l^* and u_r^*, respectively, to connect $u_{j_1}^{n+1,-}$ to $u_{j_1}^{n+1,+}$, as shown in Figure 1; see section 3.3 in [3]. We consider the solution in the critical cell to be a piecewise constant function of the form

$$\tilde{u}^{n+1}(x,t_{n+1}) = \begin{cases} u_{j_1}^{n+1,-}, & x < \xi_l^{n+1}; \\ u_l^*, & \xi_l^{n+1} \le x < \xi_m^{n+1}; \\ u_r^*, & \xi_m^{n+1} \le x < \xi_r^{n+1}; \\ u_{j_1}^{n+1,+}, & \xi_r^{n+1} \le x, \end{cases} \tag{2.15}$$

where ξ_l^{n+1}, ξ_m^{n+1}, and ξ_r^{n+1} are scalars that are to be solved (see Figure 2). When all the waves in the Riemann problem are discontinuities, we see that (2.15) is exactly the solution with ξ_l^{n+1}, ξ_m^{n+1}, and ξ_r^{n+1} as the three discontinuity positions. When there are rarefaction waves of first and third characteristic fields in the solution and when they are very weak, (2.15) is approximate to the exact solution up to the second order of the strength of the rarefaction waves.

ξ_l^{n+1}, ξ_m^{n+1} and ξ_r^{n+1} can be determined again by the requirement of conservation in the critical cell, which now turns to be

$$\frac{\xi_l^{n+1} - x_{j_1-1/2}}{h} u_{j_1}^{n+1,-} + \frac{\xi_m^{n+1} - \xi_l^{n+1}}{h} u_l^* + \frac{\xi_r^{n+1} - \xi_m^{n+1}}{h} u_r^* + \frac{x_{j_1+1/2} - \xi_r^{n+1}}{h} u_{j_1}^{n+1,+} = u_{j_1}^{n+1} \tag{2.16}$$

and is a linear system of equations.

If the critical cell is an LSCC, then ξ_l^{n+1} is the discontinuity position at the new time. In this case, $u_{j_1}^{n+1,+}$ should be adjusted according to the picture of the reconstructed solution in the critical cell. As one can see, the reconstructed $\tilde{u}^{n+1,+}(x,t_{n+1})$

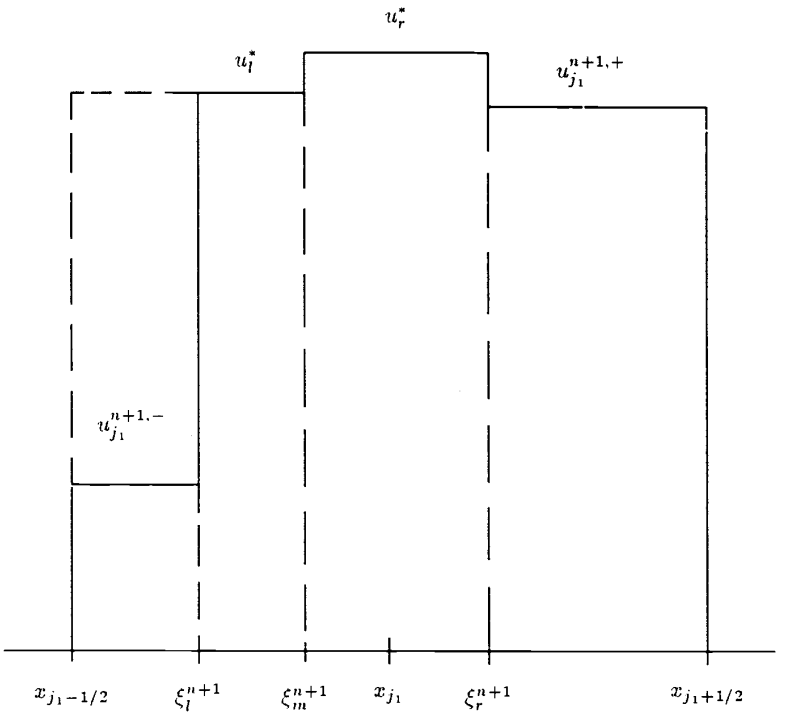


FIG. 2. *The reconstructed solution (solid line) in a critical cell is a piecewise constant vector function as shown in the figure. The two intermediate states u_l^* and u_r^* are obtained from the Riemann problem $R(u_{j_1}^{n+1,-}, u_{j_1}^{n+1,+})$. ξ_l^{n+1}, ξ_m^{n+1}, and ξ_r^{n+1} are the three discontinuity positions to be solved. The reconstructed u^+ in the critical cell is obtained by extending the right structure of u across ξ_l^{n+1} to the left as u_l^* (dash line).*

is a piecewise constant function

$$\tilde{u}^{n+1,+}(x, t_{n+1}) = \begin{cases} u_l^*, & x < \xi_m^{n+1}; \\ u_r^*, & \xi_m^{n+1} \le x < \xi_r^{n+1}; \\ u_{j_1}^{n+1,+}, & \xi_r^{n+1} \le x, \end{cases} \tag{2.17}$$

which is obtained by extending the right part of $\tilde{u}^{n+1}(x, u^{n+1})$ across ξ_l^{n+1} to the left as u_l^* (see Figure 2). According to (2.17) and the definition of cell-average, $u_{j_1}^{n+1,+}$ should be updated as

$$\begin{aligned} u_{j_1}^{n+1,+} &:= \frac{\xi_m^{n+1} - x_{j_1-1/2}}{h} u_l^* + \frac{\xi_r^{n+1} - \xi_m^{n+1}}{h} u_r^* + \frac{x_{j_1+1/2} - \xi_r^{n+1}}{h} u_{j_1}^{n+1,+} \\ &= u_{j_1}^{n+1} - \frac{\xi_l^{n+1} - x_{j_1-1/2}}{h} (u_{j_1}^{n+1,-} - u_l^*). \end{aligned} \tag{2.18}$$

In doing so, the waves associated with the second and third characteristic fields are included in the right cell-average, and thus the boundary conditions on the discontinuity are implemented. The cases of RSCC and CDCC can be treated analogously. In the case of RSCC $u_{j_1}^{n+1,-}$ is changed and $u_{j_1}^{n+1,+}$ is not, and in the case of CDCC both $u_{j_1}^{n+1,-}$ and $u_{j_1}^{n+1,+}$ are changed.

Once the discontinuity positions are obtained, the critical cells on the new time level can be determined as in the scalar case. However, in the system case in the adjustment of cell-averages, the intermediate states should be used instead of the extrapolated data. For example, if the critical cell is an LSCC and the discontinuity position moves to the left at t_{n+1}, then the $u_{j_1-1}^{n+1,+}$ on the right side of (2.14) should be replaced by u_l^*. The other cases can be treated analogously. Thus, a step of computation is completed.

As one can see, in critical cells the waves associated with the characteristic fields other than those of the tracked discontinuity are often very weak. For example, in an LSCC u_l^*, u_r^*, and $u_{j_1}^{n+1,+}$ are often quite close to each other. In this case, the coefficient matrix of (2.16) will become ill-conditioned and particular care should be taken in solving it. We set $(\xi_l^{n+1} - x_{j_1-1/2})/h$, $(\xi_m^{n+1} - x_{j_1-1/2})/h$, and $(\xi_r^{n+1} - x_{j_1-1/2})/h$ as the unknowns in the system; thus, the coefficients of $(\xi_l^{n+1} - x_{j_1-1/2})/h$ will be dominant in the coefficient matrix when the other waves are weak.

We use the Gauss elimination to solve (2.16) and choose a critical number ϵ. When the pivot in the kth step is found to be smaller than $\epsilon A_{1,1}$, where $A_{1,1}$ is the pivot in the first step, we stop the elimination and set the unknowns corresponding to the remaining part to be zero. In doing so, (2.16) is sometimes not exactly satisfied and the solved ξ_m^{n+1} and ξ_r^{n+1} are not reliable. However, the solved ξ_l^{n+1} is still reliable because the coefficients of $(\xi_l^{n+1} - x_{j_1-1/2})/h$ are still of $O(1)$. Therefore, in any case we adjust $u_{j_1}^{n+1,+}$ according to the second part of (2.18), which uses only ξ_l^{n+1}. In the numerical examples in [25], ϵ is chosen to be 10^{-4}.

Techniques that deal with interactions of discontinuities are also developed on the observation of conservation, and they resolve the numerical solution in the vicinity of the interaction points on subcell scale; see [25].

3. The tracking method for scalar conservation law in two space dimensions. From now on we develop our tracking method for conservation laws in two space dimensions

$$u_t + f(u)_x + g(u)_y = 0, \tag{3.1}$$

where $u(x, y, t)$ is either a scalar or a vector function of x, y, and t, and $f(u)$ and $g(u)$ are either scalar or vector functions of u. We first begin in this section with the scalar case, in which all u, f, and g are scalars.

We are going to develop the tracking method in semidiscretization, considering that this will be easier than the development direct in full-discretization. To this end, we partition the (x, y)-plane by setting

$$x_{i+1/2} = (i + 1/2)h, \qquad y_{j+1/2} = (j + 1/2)h \qquad (i, j \in \mathbf{Z}) \tag{3.2}$$

and denote the grid cell centered at (ih, jh) by $\Delta_{i,j}$, where h is the space increment. It will also be useful to define x_i and y_j by

$$x_i = ih, \qquad y_j = jh \qquad (i, j \in \mathbf{Z}). \tag{3.3}$$

Integrating (3.1) over $\Delta_{i,j}$ for fixed t and dividing it by h^2 gives the semidiscrete form of (3.1),

$$\frac{du_{i,j}}{dt} = -\frac{1}{h}(\hat{f}_{i+1/2,j} - \hat{f}_{i-1/2,j}) - \frac{1}{h}(\hat{g}_{i,j+1/2} - \hat{g}_{i,j-1/2}), \tag{3.4}$$

where $u_{i,j}(t)$ is a cell-average approximation to the solution at t and $\hat{f}_{i+1/2,j}(t)$ and $\hat{g}_{i,j+1/2}(t)$ are flux average approximations to $f(u)$ and $g(u)$ on the vertical and horizontal cell boundaries at the time, respectively; see section 17 in [23]. We still assume that the scheme is of Godunov-type, which means that piecewise polynomial functions are reconstructed out of cell-averages in x and y directions and the numerical fluxes are evaluated by solving Riemann problems at cell boundaries. In order to avoid using data on diagonal directions, we assume the numerical fluxes to be of the forms

$$\hat{f}_{i+1/2,j}(t) = \hat{f}(u_{i-k+1,j}(t), \ldots, u_{i+k,j}(t)) \tag{3.5}$$

and

$$\hat{g}_{i,j+1/2}(t) = \hat{g}(u_{i,j-k+1}(t), \ldots, u_{i,j+k}(t)). \tag{3.6}$$

We assume that the solution involves a discontinuity across a smooth surface in the (x, y, t)-space, on the two sides of which the solution is smooth. It is seen that at any fixed time t the discontinuity is represented by a smooth curve in the (x, y)-plane, which we denote by $C(t)$. We define the two sides of $C(t)$ to be the negative and positive sides and mark the quantities associated with the two sides by "$-$" and "$+$," respectively (the negative side may not lie on the left of the positive one).

In the two-dimensional case the critical cells are defined to be the grid cells that are cut through by $C(t)$, and the discontinuity positions are defined to be the intersection points of $C(t)$ with grid lines (see Figure 3). The discontinuity positions are classified into two kinds: the ones on horizontal grid lines, denoted by $\xi^n_{i_1,j_1-1/2}$, are called x-positions and the ones on vertical grid lines, denoted by $\eta^n_{i_1+1/2,j_1}$, are called y-positions, where Δ_{i_1,j_1} is a critical cell. Every critical cell has two discontinuity positions on its boundaries, which it shares with its adjacent critical cells (see Figure 3). The critical cells are classified into three types according to the ways $C(t)$ cuts through them. The critical cells with two x-positions or two y-positions are of xx-type or yy-type, respectively, while the ones with an x-position and a y-position are of xy-type.

The numerical solution in a critical cell Δ_{i_1,j_1}, as in the one-dimensional case, consists of three data, $u_{i_1,j_1}(t)$, the ordinary cell-average, and $u^-_{i_1,j_1}(t)$ and $u^+_{i_1,j_1}(t)$, the cell-averages computed using information only from the negative and positive sides. The discontinuity curve $C(t)$ will be recovered out of these cell-averages.

As in the one-dimensional case, the evolution of cell-averages in cells distant from the discontinuity is governed by (3.4), which is a system of ordinary differential equations (ODEs). The evolution of cell-averages in cells near the discontinuity and the evolution of positive and negative cell-averages in critical cells involve information only from the same sides. This can be done by employing extrapolated data of the cell-averages on the two sides of the discontinuity. For example, in the yy-type critical cell Δ_{i_2,j_2} with two y-positions, $\eta_{i_2-1/2,j_2}$ and $\eta_{i_2+1/2,j_2}$ (see Figure 3), the negative cell-average $u^-_{i_2,j_2}(t)$ satisfies

$$\frac{du^-_{i_2,j_2}}{dt} = -\frac{1}{h}(\hat{f}^-_{i_2+1/2,j_2} - \hat{f}^-_{i_2-1/2,j_2}) - \frac{1}{h}(\hat{g}^-_{i_2,j_2+1/2} - \hat{g}^-_{i_2,j_2-1/2}), \tag{3.7}$$

where f^- and g^- are numerical fluxes evaluated using extrapolated data from the negative side. In the evaluation, the x-fluxes use only extrapolated data in x direction while the y-fluxes use only extrapolated data in y direction. This is possible because the numerical fluxes are defined to be of the form (3.5) and (3.6). Analogously, the

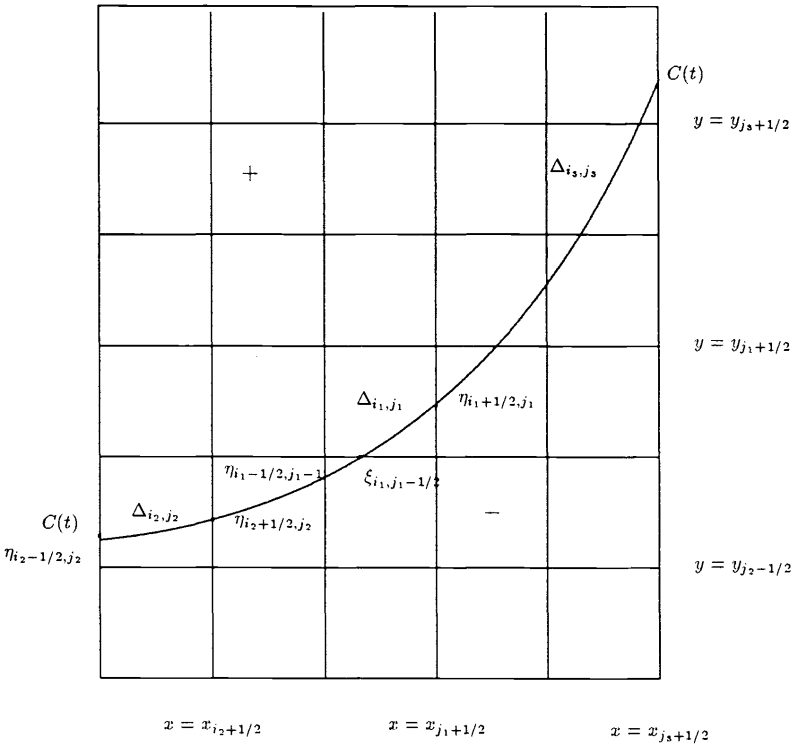


FIG. 3. *The discontinuity curve $C(t)$ cuts through the grid and divides the solution region into two parts. The negative and positive sides are indicated as in the figure. $\xi_{i_1,j_1-1/2}$ is an x-position and eta$_{i_1+1/2,j_1}$ is a y-position. δ_{i_1,j_1} is an xy-type critical cell, δ_{i_2,j_2} is a yy-type critical cell, and δ_{i_3,j_3} is an xx-type critical cell.*

positive cell-average $u^+_{i_2,j_2}(t)$ satisfies a system of equations involving information only from the positive side.

The ordinary cell-average in critical cells are computed as follows: In the yy-type critical cell Δ_{i_2,j_2}, $u_{i_2,j_2}(t)$ satisfies

$$\frac{du_{i_2,j_2}}{dt} = -\frac{1}{h}(\tilde{f}_{i_2+1/2,j_2} - \tilde{f}_{i_2-1/2,j_2}) - \frac{1}{h}(\hat{g}^+_{i_2,j_2+1/2} - \hat{g}^-_{i_2,j_2-1/2}), \tag{3.8}$$

where, according to the picture of the solution,

(3.9)

$$\tilde{f}_{i+1/2,j_2}(t) \simeq \frac{1}{h}\left(\int_{y_{j_2-1/2}}^{\eta_{i+1/2,j_2}} f(u^-(x_{i+1/2},y,t))dy + \int_{\eta_{i+1/2,j_2}}^{y_{j_2+1/2}} f(u^+(x_{i+1/2},y,t))dy\right)$$
$$i = i_2 - 1, i_2.$$

Noticing that $\hat{f}$'s are flux averages approximations, one can evaluate the two integrals in (3.9) by "reconstruction via primitive function" described in [15] on given flux averages $\hat{f}$'s, $\hat{f}^-$'s, and $\hat{f}^+$'s and the two discontinuity positions up to any specified order of accuracy. The computation in an xx-type critical cell can be carried out analogously in a symmetric way.

However, the computation is not carried out directly in an xy-type critical cell. Adjacent xy-type critical cells should first be organized to form either an xx-type or a yy-type rectangle of two cells, called *rectangle for computation* (RC). Then the computation in it is carried out as described above. For example, the critical cell Δ_{i_1,j_1} and Δ_{i_1,j_1-1} can be organized to form an RC of yy-type with two discontinuity positions $\eta_{i_1-1/2,j_1-1}$ and $\eta_{i_1+1/2,j_1}$. We define the cell-average of the RC to be

$$v_{i_1,j_1-1/2}(t) = u_{i_1,j_1}(t) + u_{i_1,j_1-1}(t). \tag{3.10}$$

Then $v_{i_1,j_1-1/2}(t)$ satisfies

$$\begin{aligned}\frac{dv_{i_1,j_1-1/2}}{dt} &= -\lambda(\tilde{f}_{i_1+1/2,j_1} + \hat{f}^-_{i_1+1/2,j_1-1} - \hat{f}^+_{i_1-1/2,j_1} - \tilde{f}_{i_1-1/2,j_1-1}) \\ &\quad -\lambda(\hat{g}^+_{i_1,j_1+1/2} - \hat{g}^-_{i_1,j_1-3/2}).\end{aligned} \tag{3.11}$$

Thus, before carrying out the computation in critical cells, we should first organize them in RCs of either xx-type or yy-type. They consist of only one grid cell when the organized critical cells are of xx-type or yy-type and consist of two grid cells when the organized critical cells are of xy-type. By organizing the critical cells in this way the small-cell problem will be avoided and the algorithm will be in a dimension-by-dimension fashion, which will be seen in the following discussion.

Now we obtain a system of ODEs that governs the evolution of the numerical solution. However, it is not closed because it involves discontinuity positions in critical cells. To close it, one needs to represent the discontinuity positions by cell-averages; in other words, one needs to recover the discontinuity positions out of the cell-averages.

To describe the recovery procedure, we begin with a special case illustrated in Figure 4(a), in which the discontinuity curve $C(t)$ crosses over a sequence of yy-type RCs of one cell centered at (x_i, y_{j_1}) for $i_1 - r \le i \le i_1 + r$; the negative and positive sides are as indicated in the figure. We are going to recover the curve segment in the cell Δ_{i_1,j_1}. It is seen that $C(t)$ can be described by a function $y = y(x)$ (we ignore the dependence of y on t since the time is fixed). We say that this discontinuity curve is of y-type, and our task is to determine the function $y(x)$.

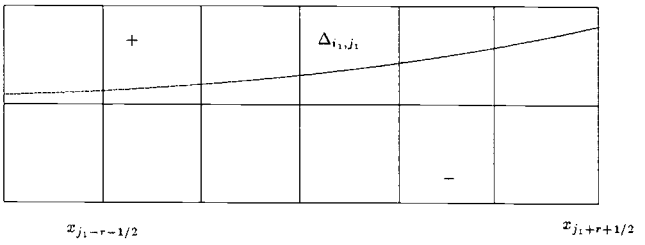
First, we reconstruct $u^-(x, y, t)$ and $u^+(x, y, t)$, the solution on the two sides of the discontinuity, by using reconstruction procedure of the underlying scheme in the y direction. The reconstruction of $u^-(x, y, t)$ uses extrapolated data in the y direction from the negative side while the reconstruction of $u^+(x, y, t)$ uses extrapolated data in the y direction from the positive side.

Now we define a function $F(x)$ as

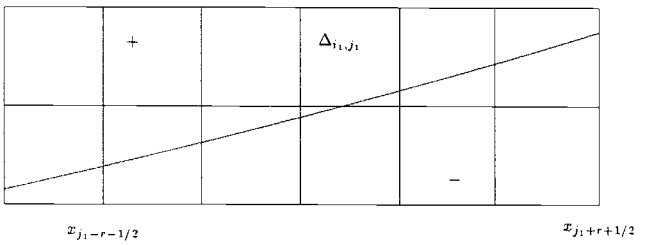
$$F(x) = \int_{x_{i_1-r-1/2}}^{x} ds \left(\int_{y_{j_1-1/2}}^{y(s)} u^-(s, y, t)dy + \int_{y(s)}^{y_{j_1+1/2}} u^+(s, y, t)dy \right). \tag{3.12}$$

On the observation of conservation and from the definition of cell-average we have

$$\begin{aligned} h^2 u_{i,j_1}(t) &= \int_{x_{i-1/2}}^{x_{i+1/2}} ds \left(\int_{y_{j_1-1/2}}^{y(s)} u^-(s, y, t)dy + \int_{y(s)}^{y_{j_1+1/2}} u^+(s, y, t)dy \right) \\ &= F(x_{i+1/2}) - F(x_{i-1/2}). \end{aligned} \tag{3.13}$$



(a)



(b)

FIG. 4. (a) *The discontinuity curve $C(t)$ crosses over a sequence of yy-type RCDs of one cell.* (b) *The discontinuity curve $C(t)$ crosses over a sequence of yy-type RCDs of either one or two cells. In the two pictures, δ_{i_1,j_1} is the yy-type critical cell in which the discontinuity is to be recovered.*

From (3.13) we can easily compute the point values $\{F(x_{i+1/2})\}$ by summation:

$$F(x_{i+1/2}) = h^2 \sum_{l=i_1-r}^{i} u_{i,j_1}(t). \tag{3.14}$$

Let $H_{r+1}(x;F)$ be an $(r+1)$th-order interpolation of F at the points $\{x_{i+1/2}\}$, where r is the order of the underlying scheme. That is,

$$H_r(x_{i+1/2};F) = F(x_{i+1/2}), \tag{3.15a}$$

$$\frac{d^m}{dx^m}H_{r+1}(x;F) = \frac{d^m}{dx^m}F(x) + O(h^{r+1-m}), \qquad 0 \le m \le r. \tag{3.15b}$$

In particular, for $m = 1$ we have

$$\begin{aligned} H'_{r+1}(x;F) &= F'(x) + O(h^r) \\ &= \int_{y_{j_1-1/2}}^{y(x)} u^-(x,y,t)dy + \int_{y(x)}^{y_{i_1+1/2}} u^+(x,y,t)dy + O(h^r). \end{aligned} \tag{3.16}$$

Then the segment of the discontinuity curve in the critical cell can be recovered by solving for $y(s)$ form (3.16). In particular, when u^- and u^+ are piecewise constant and the interpolation $H_{r+1}(x;F)$ is piecewise linear, the recovered curve in the critical cell Δ_{i_1,j_1} is a segment of line

$$y(x) = \xi = y_{j_1-1/2} + h\frac{u_{i_1,j_1}(t) - u^+_{i_1,j_1}(t)}{u^-_{i_1,j_1}(t) - u^+_{i_1,j_1}(t)} \qquad \text{for } x_{i_1-1/2} \le x \le x_{i_1+1/2}. \tag{3.17}$$

The discontinuity curve can also be recovered if it crosses over RCs of two cells, only in this case more rows of cell-averages should be involved in defining $F(x)$ (see Figure 4(b))

Recovered in such a way, the curve is actually not continuous but piecewise smooth. We should reorganize the curve segments to produce a continuous curve. For example, we can define $C(t)$ as follows: Assume that the curve segments recovered in the critical cells Δ_{i_1,j_1} and Δ_{i_1+1,j_1} are $y_{i_1}(x)$ and $y_{i_1+1}(x)$. Then the segment of the discontinuity curve in $[x_{i_1}, x_{i_1+1}]$ is defined as

$$y(x) = \frac{x - x_{i_1}}{h} y_{i_1+1}(x) + \frac{x_{i_1+1} - x}{h} y_{i_1}(x). \tag{3.18}$$

In doing so, the discontinuity position $\eta_{i_1+1/2,j_1}$ is defined just as the mean value of the two recovered positions at $x_{i_1+1/2}$.

A discontinuity curve of x-type can be defined and recovered analogously. Conceptually speaking, a general smooth discontinuity curve can always be covered by several subcurves that are either of x-type or of y-type, and each of them can be recovered as above. At the overlapped part of two adjacent subcurves the recovered curve segment can be taken as a mean weighted in some way of the two segments recovered in the two subcurves. However, in this paper we consider only discontinuity curves of either x-type or y-type and leave the issue of general curve to a future study.

It is seen that the recovery procedure is close in spirit to the "reconstruction via primitive function" for essentially nonoscillatory (ENO) schemes described in [15]. It uses several cell-averages in the vicinity to recover the segment of discontinuity. As is well known, the "reconstruction via primitive function" has degrees of freedom of selecting the stencil for the interpolation. For Godunov-type schemes, various stencil selections are designed to eliminate spurious oscillations and to retain the smoothness of the numerical solutions, among which the ENO stencil selections distinguish themselves by their moving feature that always obtains information from regions of smoothness when discontinuities are present; see [16], [17]. We see that the stencil selections of Godunov-type schemes, in particular the ENO stencil selections, can also be applied to the interpolation in this recovery procedure to stabilize the computation.

Now we close the system of ODEs that governs the evolution of the numerical solution by substituting the recovered discontinuity positions into it. Thus, the tracking method in semidiscrete form is completed and its spatial accuracy is of high order. To obtain a full-discretization of the method that is of high order in time as well as in space, we can discretize the system by the Runge–Kutta method. As one can see, the right side of the system of ODEs, which involves the recovery procedure, is not so explicit; therefore, the direct full-discretization of the tracking method will be complicated.

Finally, we need to determine the critical cells at the new time according to the picture of the discontinuity curve at the time. In doing this, the conservation should be preserved. Two typical cases illustrated in Figure 5 are discussed in the following, and the other cases can be treated analogously.

If $C(t_{n+1})$ is as shown in Figure 5(a), then Δ_{i_1,j_1} is a critical cell at t_{n+1}. If Δ_{i_1,j_1} is also the critical cell at t_n, then the numerical solution has nothing to change. Otherwise, if Δ_{i_1,j_1-1} is the critical cell at t_n, then in Δ_{i_1,j_1-1}, $u^{n+1}_{i_1,j_1-1}$ is defined to be the computed $u^{n+1,-}_{i_1,j_1-1}$, and in Δ_{i_1,j_1}, $u^{n+1,+}_{i_1,j_1}$ is defined to be the computed $u^{n+1}_{i_1,j_1}$, $u^{n+1,-}_{i_1,j_1}$ is defined to be the extrapolated data of the cell-averages from the negative side in the y direction, and the ordinary cell-average $u^{n+1}_{i_1,j_1}$ is computed by

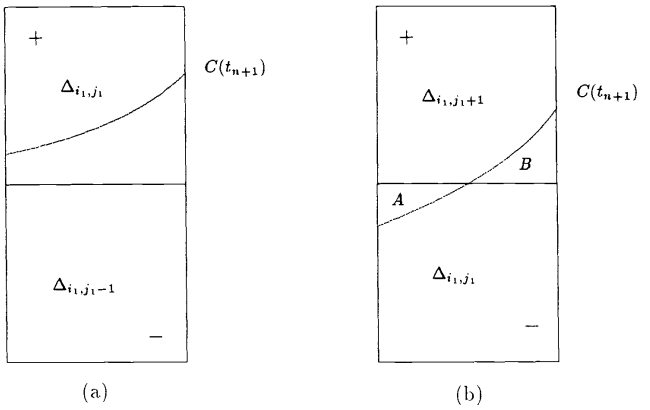


FIG. 5. (a) *Discontinuity curve $C(t_{n+1})$ crosses one grid cell in y direction.* (b) $C(t_{n+1})$ *crosses two cells in y direction.*

the requirement that the sum of the cell-averages in the two grid cells be conserved.

If $C(t_{n+1})$ is as shown in Figure 5(b), then both Δ_{i_1,j_1} and Δ_{i_1,j_1+1} are critical cells at t_{n+1}. The negative and positive cell-averages can still be computed according to the locations of the critical cells at t_n. According to the picture, it seems that the ordinary cell-averages in the two cells can be computed as

$$Q_1 = \frac{1}{h^2}\left(\iint_{\Delta_{i_1,j_1}\setminus A} u^{n,-}(x,y,t_{n+1})dxdy + \iint_A u^{n,+}(x,y,t_{n+1})dxdy\right) \tag{3.19}$$

and

$$Q_2 = \frac{1}{h^2}\left(\iint_B u^{n,-}(x,y,t_{n+1})dxdy + \iint_{\Delta_{i_1,j_1+1}\setminus B} u^{n,+}(x,y,t_{n+1})dxdy\right), \tag{3.20}$$

where A and B are the curved triangles formed by $C(t_{n+1})$ and the cell-boundaries. However, taking Q_1 and Q_2 as the new cell-averages may not maintain the conservation. To maintain the conservation we take

$$u^{n+1}_{i_1,j_1} = \frac{1}{2}(v^{n+1}_{i_1,j_1+1/2} + Q_1 - Q_2) \tag{3.21}$$

and

$$u^{n+1}_{i_1,j_1+1} = \frac{1}{2}(v^{n+1}_{i_1,j_1+1/2} + Q_2 - Q_1), \tag{3.22}$$

where $v^{n+1}_{i_1,j_1+1/2}$ is the sum of the two cell-averages. In doing so, the adjustment will not favor any one of the new critical cells. Thus, we complete a step of computation.

Now we are going to give some interesting observations on the tracking method to end this section. To this end, we assume the discontinuity curve still to be of y-type as shown in Figure 4(a). We see that the recovered discontinuity positions in the critical cell Δ_{i_1,j_1} depend on the cell-averages from $\bar{u}_{i_1-r-1,j_1}(t)$ through $\bar{u}_{i_1+r+1,j_1}(t)$; therefore, we have from (3.8) that $\bar{u}_{i_1,j_1}$ satisfies

$$\frac{d\bar{u}_{i_1,j_1}}{dt} = -\frac{1}{h}(\mathcal{F}_{i_1+1/2,j_1} - \mathcal{F}_{i_1-1/2,j_1}) + \mathcal{G}_{i_1,j_1}, \tag{3.23}$$

where

$$\mathcal{F}_{i_1+1/2,j_1} = \mathcal{F}(\bar{u}_{i_1-r,j_1}, \ldots, \bar{u}_{i_1+r+1,j_1}) \tag{3.24}$$

is a function depending on cell-averages from $\bar{u}_{i_1-r,j_1}(t)$ through $\bar{u}_{i_1+r+1,j_1}(t)$ and

$$\mathcal{G}_{i_1,j_1} = -\frac{1}{h}(g^+_{i_1,j_1+1/2} - g^-_{i_1,j_1-1/2}) \tag{3.25}$$

is a known term. Equation (3.23) is a conservative scheme that describes the evolution of the cell-average in the critical cell. The first question is, To what differential equation does (3.23) approximate? For this regard, we fix the x-space increment and shrink the y-space increment to zero (in doing so we have assumed the grid to be rectangular) and obtain

$$\frac{\partial \int_{y_{j_1-1/2}}^{y_{j_1+1/2}} u(x,y,t)dy}{\partial t} + \frac{\partial \int_{y_{j_1-1/2}}^{y_{j_1+1/2}} f(u(x,y,t))dy}{\partial x} + G(x, y_{j_1}, t) = 0, \tag{3.26}$$

where

$$\int_{yj_1-1/2}^{y_{j_1+1/2}} f(u(x,y,t))dy = \int_{y_{j_1-1/2}}^{y(x,t)} f(u^-(x,y,t))dy + \int_{y(x,t)}^{y_{j_1+1/2}} f(u^+(x,y,t))dy), \tag{3.27}$$

in which $y(x,t)$ is the discontinuity position that is determined from $\int_{y_{j_1-1/2}}^{y_{j_1+1/2}} u(x,y,t)dy$ by

$$\int_{y_{j_1-1/2}}^{y_{j_1+1/2}} u(x,y,t)dy = \int_{y_{j_1-1/2}}^{y(x,t)} u^-(x,y,t)dy + \int_{y(x,t)}^{y_{j_1+1/2}} u^+(x,y,t)dy \tag{3.28}$$

and

$$G(x, y_{j_1}, t) = g(u(x, y_{j_1+1/2}, t)) - g(u(x, y_{j_1-1/2}, t)). \tag{3.29}$$

In derivation of (3.26) we have used

$$\mathcal{F}(u, \ldots, u) = \int_{y_{j_1-1/2}}^{y_{j_1+1/2}} f(u(x,y,t))dy, \tag{3.30}$$

which is obvious in the discussion. Equation (3.26) is a partial differential equation (PDE) of conservation law with source term, which describes the evolution of the quantity $\int_{y_{j_1-1/2}}^{y_{j_1+1/2}} u(x,y,t,)dy$.

The second question is, What is the relation between (3.26) and the Hugoniot condition? For this regard, we substitute (3.27) and (3.28) into (3.26) and view $y(x,t)$ as the dependent variable in (3.26). Then by differentiation and noticing (3.1) we obtain

$$(u^+ - u^-)\frac{\partial y}{\partial t} + (f(u^+) - f(u^-))\frac{\partial y}{\partial x} - (g(u^+) - g(u^-)) = 0, \tag{3.31}$$

which is only the Hugoniot condition of the two-dimensional conservation law in the form with x and t being the independent variables; see [24].

In all the other types of front tracking methods mentioned in this paper, the tracked fronts are moved by the normal propagation speed computed from the solutions to the Riemann problems. In doing so, the Hugoniot condition is treated as a group of ODEs rather than as a PDE, from which many geometrical and topological complexities follow. In those methods much of the algorithm has to be spent on dealing with the reorganizing of the new fronts, untangling the tangled fronts, etc. However, as can be seen in the above discussion, in our method the Hugoniot condition is treated as a PDE rather than as a group of ODEs, and (3.23) shows that the complexity of the algorithm is only of that of a one-dimensional front capturing method, no reorganizing and untangling are needed.

4. The tracking method for the Euler system of gas dynamics in two space dimensions. The Euler system of gas dynamics in two space dimensions is

$$u_t + f(u)_x + g(u)_y = 0, \tag{4.1a}$$

$$u = (\rho, m^x, m^y, E)^T, \tag{4.1b}$$

$$f(u) = q^x u + (0, p, 0, q^x p)^T, \tag{4.1c}$$

$$g(u) = q^y u + (0, 0, p, q^y p)^T, \tag{4.1d}$$

$$p = (\gamma - 1)\left(E - \frac{1}{2}\rho q^2\right), \qquad q^2 = (q^x)^2 + (q^y)^2, \tag{4.1e}$$

where ρ, q^x, q^y, p, and E are the density, x-component, and y-component of velocity, pressure, and total energy, respectively, $m^x = \rho q^x$ is the x-component of momentum, $m^y = \rho q^y$ is the y-component of momentum, and γ is the ratio of specific heats.

The one-dimensional Riemann problem for the two-dimensional Euler system is an initial value problem with the initial data consisting of two constant states u^- and u^+ separated by a jump of a straight line. This problem can be solved in the normal direction of the straight line by neglecting the velocity parallel to the jump. Any difference in the parallel component is assumed to take place across the contact surface. Thus, the solution has a similar structure as that in one space dimension. It consists of four constant states, two centered waves, either shocks or rarefactions, and a contact discontinuity consisting of a discontinuity of density and a discontinuity of tangential velocity, a slip-line. The two intermediate states $u^{*,-}$ and $u^{*,+}$ have the same pressure and normal velocity but different densities and tangential velocities. Here we shall introduce in addition to the four constant states one more intermediate state $u^{*,t}$ which has the velocity and pressure of $u^{*,-}$ and the density of $u^{*,+}$. Defined in this way, $u^{*,t}$ is connected with $u^{*,-}$ by a density discontinuity and is connected with $u^{*,+}$ by a slip-line. The necessity of doing this will be seen in the following discussion.

The tracking method to be presented in this section is first-order only. This will be seen in the following discussion. The high-order tracking method is still under investigation. As one has seen in section 2, the things that need to be revised in the extension of the method in the system case because of the presence of different characteristics are (1) the classification of critical cells and of discontinuity curves and (2) the reconstruction of the numerical solution in critical cells.

We define the orientation of a discontinuity curve to be that of the normal to the curve points from the negative side to the positive side. In an LSCC the one-dimensional Riemann problem in the normal direction $\vec{n}$, $R(u^-_{i_1,j_1}(t), u^+_{i_1,j_1}(t), \vec{n})$, has

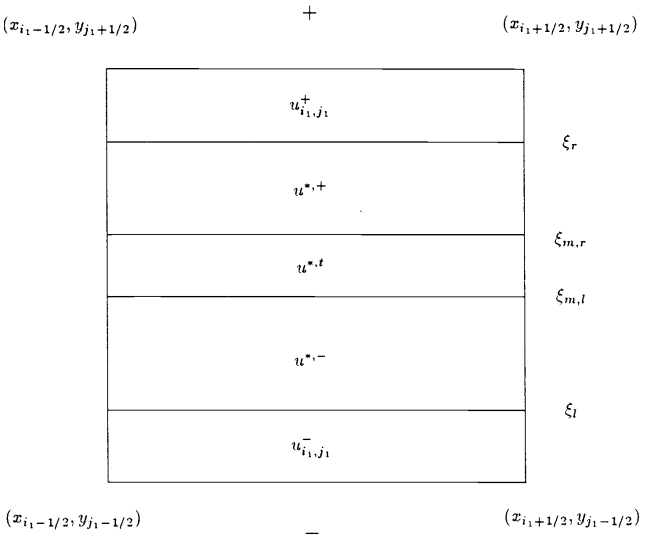


FIG. 6. *The reconstruced solution in the critical cell δ_{i_1,j_1} is constant in each rectangle, where ξ_l, $\xi_{m,l}$, and ξ_r are the locations of the jumps.*

a strong left shock, while in an RSCC and a CDCC it has a strong right shock and a strong contact discontinuity, respectively. A discontinuity curve is said to be LS, RS, or CD if all its critical cells are LSCC, RSCC, or CDCC, respectively.

We still consider the method in semidiscrete form and consider the special case illustrated in Figure 4(a) for the reconstruction of solution. The cases containing the RC of two cells can be treated also by involving more rows of cell-averages. We consider that the solution to be reconstructed in the critical cell Δ_{i_1,j_1} is piecewise constant. However, unlike the scalar case, in the system case it should involve all the five states that come from the Riemann problem $R(u^-_{i_1,j_1}(t), u^+_{i_1,j_1}(t), \vec{n})$. We also consider that the jumps of the solution in the critical cell are segments of straight lines parallel to the x axis; i.e., the solution has the form

$$u(x,y,t) = \begin{cases} u^-_{i_1,j_1}, & y_{j_1-1/2} < y \le \xi_l, \\ u^{*,-}, & \xi_l < y \le \xi_{m,l}, \\ u^{*,t}, & \xi_{m,l} < y \le \xi_{m,r}, \\ u^{*,+}, & \xi_{m,r} < y \le \xi_r, \\ u^+_{i_1,j_1}, & \xi_r < y \le y_{j_1+1/2}, \end{cases} \tag{4.2}$$

where ξ's are the positions of the jumps as shown in Figure 6. We then follow the idea described in section 3 to recover the discontinuity curve in Δ_{i_1,j_1}. Under the above considerations, the discontinuity is recovered only by the cell-averages in this cell, and we are led to the 4×4 linear system

$$\begin{aligned} &\frac{\xi_l - y_{j_1-1/2}}{h} u_{i_1,j_1} + \frac{\xi_{m,l} - \xi_l}{h} u^{*,-} + \frac{\xi_{m,r} - \xi_{m,l}}{h} u^{*,t} \\ &+ \frac{\xi_r - \xi_{m,r}}{h} u^{*,+} + \frac{y_{j_1+1/2} - \xi_r}{h} u_{i_1,j_1} = u_{i_1,j_1}, \end{aligned} \tag{4.3}$$

which comes from the requirement of conservation in the critical cell.

We see that the contact discontinuity has two degrees of freedom. One can has a contact discontinuity with only a density jump and one can have a contact discontinuity with only a tangential velocity jump. These two degrees of freedom must be separated in the reconstruction. Otherwise, if we still define only one discontinuity position for the contact discontinuity, (4.3) will become a system of four equations but with only three unknowns and, thus, will be overdetermined.

Once (4.3) is solved, the discontinuity curve in the critical cell can be recovered. If the tracked discontinuity is an LS, the recovered discontinuity curve in the critical cell is a segment of line

$$y = \xi_l, \quad x_{i_1-1/2} < x \le x_{i_1+1/2} \ . \tag{4.4}$$

The positive cell-average in the critical cell $u^+_{i_1,j_1}$ should be adjusted as

(4.5)

$$\begin{aligned} u^+_{i_1,j_1} &:= \frac{\xi_{m,l} - y_{j_1-1/2}}{h} u^{*,-} + \frac{\xi_{m,r} - \xi_{m,l}}{h} u^{*,t} + \frac{\xi_r, -\xi_{m,l}}{h} u^{*,+} + \frac{y_{j_1+1/2} - \xi_r}{h} u^+_{i_1,j_1} \\ &= u_{i_1,j_1} - \frac{\xi_l - y_{j_1-1/2}}{h} (u^-_{i_1,j_1} - u^{*,-}). \end{aligned}$$

Once the discontinuity curve is recovered in each critical cell, a continuous discontinuity curve can be produced as in (3.18). An RS can be treated analogously.

When the discontinuity is a CD, there will be two computed positions $\xi_{m,l}$ and $\xi_{m,r}$ for it. According to the structure of the solution to the Riemann problem, $\xi_{m,l}$ and $\xi_{m,r}$ are supposed to be the same since the density discontinuity and the slip-line coincide. However, in the numerical computation they are different due to the truncation errors and round-off errors. The final discontinuity position of the contact discontinuity can be taken as a weighted mean of $\xi_{m,l}$ and $\xi_{m,r}$, and the error of doing this is of the same order as the truncation error. The weight can be taken, say, according to the strengths of the density jump and the slip-line. However, in this paper we consider only shocks and leave the issue to our future study.

As in the one-dimensional case, the coefficient matrix of (4.3) will often become ill-conditioned because the waves associated with the other characteristics are often quite weak. The particular care described in section 3 should also be taken in solving (4.3) by the Gauss elimination. However, the situation in two space dimensions is a bit more severe than that in one space dimension. We design the critical number ϵ as follows: We select three numbers $\epsilon_1 < \epsilon_2 < \epsilon_3$ for the critical number. First we set ϵ to be ϵ_1 and then check the computed discontinuity positions. If any one of them is found more than two cells away from the center of the critical cell, we set ϵ to be ϵ_2 and then check the computed discontinuity positions again. If again any one of them is found more than two cells away from the center of the critical cell, we set ϵ to be ϵ_3. In the numerical example of the regular reflection problem presented in section 7, the ϵ_1, ϵ_2, and ϵ_3 are taken to be 10^{-3}, 2.5×10^{-3}, and 5×10^{-3}. We found that almost all the cases can be taken care of with ϵ_1, only very few cases should be taken care of with ϵ_2 and ϵ_3, and the bad cases occur only in the regions near to the foot of the reflected shock and very occasionally.

Up to now the normal direction $\vec{n}$ needed in defining the Riemann problem is still unknown. The normal direction can only be computed from the discontinuity positions. For example, the normal can be taken as the normal to the line connecting

the two discontinuity positions in the critical cell. Therefore, the above discussion does not actually recover the discontinuity positions because the discontinuity positions themselves are used in it. However, it produces an operator equation about the discontinuity positions,

$$(\xi, \eta) = \mathcal{P}(\xi, \eta), \tag{4.6}$$

where $\mathcal{P}$ stands for the recovery procedure described above. In the full-discretization of the tracking method we can use the discontinuity positions obtained at the previous time step in the right side of (4.6). For example, when we use the predictor-corrector method to discretize the time derivative, in the predictor step the discontinuity positions at t_n are used to compute the normal, and in the corrector step the discontinuity positions obtained in the predictor step are used to compute the normal. In doing so, the first order of accuracy is maintained.

Finally, critical cells on the new time level are determined according to the picture of the discontinuity curve on this level. Only in the system case should the left or right intermediate states be used to define the positive and negative cell-averages in the new critical cells when the discontinuity moves to new grid cells, as it was done in the one-dimensional case. Thus, we complete a step of computation.

The tracking method presented here is only first-order. To obtain the high-order tracking method, we should replace the piecewise constant reconstruction by a piecewise polynomial reconstruction to reconstruct the solution, replace the piecewise constant recovery by a piecewise polynomial recovery to recover the discontinuity curve, and find a feasible solver of the operator equation (4.6) which is high-order accurate.

5. Treatment of reflection wall boundary. The basic ideas of the tracking method can be summarized as follows:

(1) On each side of a discontinuity the computation uses information only from the same side.

(2) We reconstruct the solution and locate the discontinuity positions based on conservation.

(3) We extend the solution on each side of the discontinuity properly to the other side so that the solution is well defined on either side in critical cells (refer, for example, to (2.18) and (4.5)).

These ideas can also be applied to treat a reflection wall (RW) boundary for the Euler system of gas dynamics so that the computation in this part will be carried out on a uniform Cartesian grid. The things that are different for an RW boundary are as follows:

(1) The RW boundary is fixed in the flow; therefore, it needs not to be tracked,

(2) The Hugoniot conditions do not hold on the RW boundary; instead, the flow flux is reflected on it.

We develop a treatment of RW boundaries in this section. As we will see in the following discussion, the treatment is only first-order because it is developed under the consideration that the numerical solution is piecewise constant.

Assume that there is a reflection wall and that it is represented by a fixed curve C at all the times. The grid cells that are cut through by C are called the reflection wall critical cells (RWCC). The discontinuity positions, geometrical types of the critical cells, are defined as those for physical discontinuities. As one can see, discontinuity positions of an RW boundary are fixed in the flow; therefore, the treatment of an RW boundary can be developed in full-discrete form.

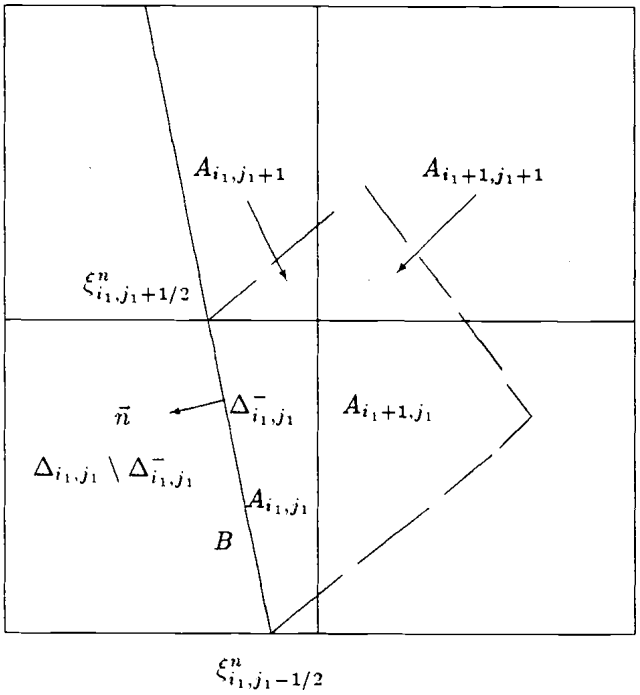


FIG. 7. *δ_{i_1,j_1} is an xx-type RWCC with two discontinuity positions $\xi^n_{i_1,j_1+1/2}$ and $\xi^n_{i_1,j_1-1/2}$. A_{i_1,j_1}, A_{i_1+1,j_1}, A_{i_1+1,j_1+1} are the intersections of the mirror image of $\delta_{i_1,j_1} \setminus \delta^-_{i_1,j_1}$ with the grid cell δ_{i_1,j_1}, δ_{i_1+1,j_1}, δ_{i_1+1,j_1+1}, and δ_{i_1,j_1+1}, respectively.*

We describe only the treatment of xx-type RWCC; the other types of RWCC can be treated analogously without any substantial difference. Assume that Δ_{i_1,j_1} is an xx-type RWCC with two x-positions $\xi_{i_1,j_1-1/2}$ and $\xi_{i_1,j_1+1/2}$ as shown in Figure 7. The wall boundary in Δ_{i_1,j_1} is denoted by B and is assumed to be a segment of a line. For the RW boundary, there is only one side of the flow; the flow in the other side does not exist. We define the side in the flow to be the negative side and define $u^{n,-}_{i_1,j_1}$ to be the cell-average at t_n computed using information only from the negative side. We also define $\Delta^-_{i_1,j_1}$ to be the part of the critical cell that lies on the negative side and define $u^{n,p}_{i_1,j_1}$ to be the partial cell-average that approximates the quantity

$$\frac{1}{h^2}\iint_{\Delta^-_{i_1,j_1}} u(x,y,t_n)dxdy. \tag{5.1}$$

In $\Delta^-_{i_1,j_1}$ the exact solution of the Euler system satisfies

$$\begin{aligned}&\frac{1}{h^2}\iint_{\Delta^-_{i_1,j_1}} u(x,y,t_{n+1})dxdy\\ &= \frac{1}{h^2}\iint_{\Delta^-_{i_1,j_1}} u(x,y,t_n)dxdy - \lambda f^n_{i_1+1/2,j_1} - \lambda(g^{n,p}_{i_1,j_1+1/2} - g^{n,p}_{i_1,j_1-1/2}) - \lambda\mathcal{F}^{n,b},\end{aligned} \tag{5.2}$$

where

$$f^n_{i_1+1/2,j_1} = \frac{1}{h\tau}\int_{t_n}^{t_{n+1}}\int_{y_{j_1-1/2}}^{y_{j_1+1/2}} f(u(x_{i_1+1/2},y,t))dydt, \tag{5.3}$$

$$g^{n,p}_{i_1,j_1\mp 1/2} = \frac{1}{h\tau}\int_{t_n}^{t_{n+1}}\int_{\xi_{i_1,j_1\mp 1/2}}^{x_{i_1+1/2}} g(u(x,y_{j_1\mp 1/2},t))dxdt, \tag{5.4}$$

and

$$\mathcal{F}^{n,b} = \frac{1}{h\tau}\int_{t_n}^{t_{n+1}}\int_B (f(u),g(u))\cdot\vec{n}dsdt, \tag{5.5}$$

where $\vec{n}$ is the outward normal vector of the wall boundary. $\mathcal{F}^{n,b}$ is, in fact, the average of the flux reflected by the wall. A main ingredient in the treatment is the evaluation of $\mathcal{F}^{n,b}$.

Assume that the full-discrete scheme for (3.1) is

$$u^{n+1}_{i,j} = u^n_{i,j} - \lambda(\hat{f}^n_{i+1/2,j} - \hat{f}^n_{i-1/2,j} + \hat{g}^n_{i,j+1/2} - \hat{g}^n_{i,j-1/2}), \tag{5.6}$$

where $u^n_{i,j}$ and $u^{n+1}_{i,j}$ are the cell-average approximations at the two times, $\hat{f}^n_{i+1/2,j}$ and $\hat{g}^n_{i,j+1/2}$ are the flux average approximations, and $\lambda = \tau/h$ is the mesh ratio. First we compute a provisional value of $u^{n+1,-}_{i_1,j_1}$ by (5.6) using information only from the negative side. Now we discretize (5.2) as follows:

$$u^{n+1,p}_{i_1,j_1} = u^{n,p}_{i_1,j} - \lambda\hat{f}^n_{i_1+1/2,j_1} - \lambda(\hat{g}^{n,p}_{i_1,j_1+1/2} - \hat{g}^{n,p}_{i_1,j_1-1/2}) - \lambda\hat{\mathcal{F}}^{n,b}, \tag{5.7}$$

where $u^{n,p}_{i_1,j_1}$ and $u^{n+1,p}_{i_1,j_1}$ are the approximations to the partial cell-averages at the two times, $\hat{g}^{n,p}_{i_1,j_1+1/2}$ and $\hat{g}^{n,p}_{i_1,j_1-1/2}$ are the approximations to $g^{n,p}_{i_1,j_1+1/2}$ and $g^{n,p}_{i_1,j_1-1/2}$, and $\hat{\mathcal{F}}^{n,b}$ is the approximation to $\mathcal{F}^{n,b}$.

The flux averages $g^{n,p}_{i_1,j_1\mp 1/2}$ can be computed from the $\hat{g}^n_{i,j+1/2}$'s and the discontinuity positions in the way as the two integrals in (3.9) are computed from $\hat{f}_{i+1/2,j}$'s and the discontinuity positions. $\hat{\mathcal{F}}^{n,b}$ is computed as follows: For a given normal direction $\vec{n}$, we define an operator $\mathcal{R}_{\vec{n}}$ such that $\mathcal{R}_{\vec{n}}(u)$ is a state which has the same density, pressure, and tangential velocity of u but opposite normal velocity of u. In the RWCC Δ_{i_1,j_1}, we solve the Riemann problem $R(u^{n,-}_{i_1,j_1}, \mathcal{R}_{\vec{n}}(u^{n,-}_{i_1,j_1}), \vec{n})$ and obtain the intermediate state $u^{n,*}$ (this Riemann problem has only one intermediate state). We do the same job for $u^{n+1,-}_{i_1,j_1}$ and obtain $u^{n+1,*}$. Then $\hat{\mathcal{F}}^{n,b}$ is computed as

$$\hat{\mathcal{F}}^{n,b} = 0.5(f(u^{n,*}) + f(u^{n+1,*}), g(u^{n,*}) + g(u^{n+1,*}))\cdot\vec{n}\frac{|B|}{h}, \tag{5.8}$$

where $|B|$ is the length of the segment B.

Once $u^{n+1,p}_{i_1,j_1}$ is computed, $u^{n+1,-}_{i_1,j_1}$ should be recomputed. According to the definition,

$$u^{n+1,-}_{i_1,j_1} \simeq \frac{1}{h^2}\iint_{\Delta_{i_1,j_1}} u^{n,-}dxdy = \frac{1}{h^2}\iint_{\Delta^-_{i_1,j_1}} u^n dxdy + \iint_{\Delta_{i_1,j_1}\setminus\Delta^-_{i_1,j_1}} u^{n,-}dxdy. \tag{5.9}$$

The first term on the left side of (5.9) is approximated by $u^{n+1,p}_{i_1,j_1}$. The difficulty is the evaluation of the second term since the solution on the positive side does not exist.

The question is how to properly extend u in $\Delta_{i_1,j_1} \setminus \Delta^-_{i_1,j_1}$ so that the waves reflected by the wall will go into the flow.

It is well known that the reflection wall boundary conditions can be accomplished by setting on the other side of the wall a state that has the same density, pressure, and tangential velocity but opposite normal velocity. This means that on the wall boundary we should solve the Riemann problem between the two states and the solution u should be extended to the positive side as the intermediate state in the Riemann problem. Under this consideration we compute $u^{n+1,-}_{i_1,j_1}$ as follows: We mirror $\Delta_{i_1,j_1} \setminus \Delta^-_{i_1,j_1}$ at the wall and obtain its mirror image lying in the flow (see Figure 7, the area indicated by dash lines). The mirror image intersects with the grid cells on the negative side and we denote by $A_{i,j}$ the intersection of the image with the cell $\Delta_{i,j}$ and by $|A_{i,j}|$ the area of the intersection. In each cell that intersects with the mirror image we solve the Riemann problem $R(\tilde{u}^{n+1}_{i,j}, \mathcal{R}_{\vec{n}}(\tilde{u}^{n+1}_{i,j}), \vec{n})$ to obtain an intermediate state $u^*_{i,j}$, where $\tilde{u}^{n+1}_{i,j}$ is $u^{n+1}_{i,j}$ if the cell is ordinary and is the provisional value of $u^{n+1,-}_{i,j}$ if the cell is an RWCC. Then $u^{n+1,-}_{i_1,j_1}$ is evaluated as

$$u^{n+1,-}_{i_1,j_1} = u^{n+1,p}_{i_1,j_1} + \frac{1}{h^2} \sum_{i,j} |A_{i,j}| u^*_{i,j}. \tag{5.10}$$

Thus, the treatment of the RWCC is completed.

Here we would like to mention the work of [22], [6], and [7], in which treatments of RW boundaries are also studied. The idea of the mirror image and its intersections with the grid cells actually comes from those papers. However, the boundary treatment presented here is different from theirs in essence.

6. Treatments of three special cases of discontinuity interactions. As was discussed in the introduction, there are difficulties for the traditional front tracking methods in dealing with complex fronts. One typical case of complex fronts is the discontinuity interactions. The treatment of discontinuity interactions of the traditional front tracking methods relies heavily on the knowledge of the exact solutions to two dimensional Riemann problems; see [11] and the references cited therein. However, the solutions to two-dimensional Riemann problems are much more complicated than those for one-dimensional problems; see [8]. This situation makes the two-dimensional traditional front tracking methods very expansive and it is impossible to build general algorithms for them. The solutions to two-dimensional Riemann problems have too many possible cases and it is not realistic to build an algorithm that covers all of them.

The problem seems to have a solution in our conservative front tracking method. As was discussed in the introduction, the method works by enforcing the conservation properties of the PDE; thus, it relaxes the requirement that the discontinuity positions be accurately computed in every time step. This relaxation allows us to not solve the two dimensional Riemann problems in treating the two dimensional discontinuity interactions.

In this section, as a beginning of the study, we present the treatments of three special cases of discontinuity interactions on the observation of conservation without solving two-dimensional Riemann problems. These cases will occur in the numerical examples in the following section. The idea of the treatments applies to more complex cases. The development of a systematic treatment of discontinuity interactions will be discussed in our future papers. The treatments are presented in full-discrete form.

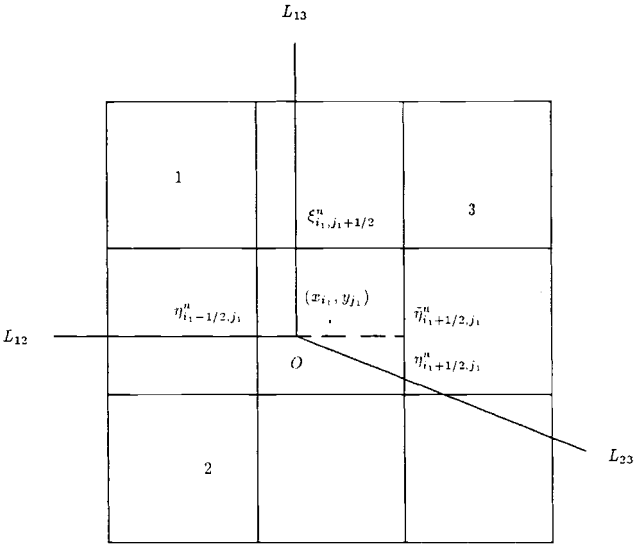


FIG. 8. *Incoming discontinuites L_12 and L_13 meet at O and produce an outgoing discontinuity L_23. The three discontinuities divide the whole region into three parts marked by 1, 2, and 3, respectively. The cell δ_{i_1,j_1}, which contains O, is the node cell. $\xi^n_{i_1,j_1+1/2}$, $\eta^n_{i_1+1/2,j_1}$, and $\eta^n_{i_1-1/2,j_1}$ are the three discontinuity positions of the node cell. The imaginary y-postion $\tilde{\eta}^n_{i_1+1/2,j_1}$ that belongs to L_12 is the intersection point of the extension of L_12 and the grid $x = x_{i_1+1/2}$.*

(1) *Triple point in scalar conservation law.*

We are concerned with the scalar conservation law and are using (5.6) as the underlying scheme. Assume that there is a horizontal discontinuity L_{12} moving to the above and a vertical discontinuity L_{13} moving to the left. They meet at point O and produce the third discontinuity L_{23}, which is of either x-type or y-type. The xy plane is then divided into three regions in which the numerical solution is denoted by $u^{n,1}$, $u^{n,2}$, and $u^{n,3}$, respectively (see Figure 8). It is seen that L_{12} and L_{13} are the incoming discontinuities, L_{23} is the outgoing discontinuity, and O is the triple point.

We call the grid cell containing O the node cell. In the node cell the numerical solution has four cell-averages, $u^n_{i_1,j_1}$, the ordinary cell-average, and $u^{n,1}_{i_1,j_1}$, $u^{n,2}_{i_1,j_1}$, and $u^{n,3}_{i_1,j_1}$, the cell-averages computed using information only from regions 1, 2, and 3, respectively, where (x_{i_1}, y_{j_1}) is the center of the node cell. The node cell has three discontinuity positions, $\eta^n_{i_1-1/2,j_1}$, belonging to L_{12}, $\xi^n_{i_1,j_1+1/2}$, belonging to L_{13}, and either $\xi^n_{i_1,j_1-1/2}$ or $\eta^n_{i_1+1/2,j_1}$, belonging to L_{23} (see also Figure 8).

We define three critical cells overlapping in the node cell, each of which belongs to one of the discontinuities. For each critical cell, one of the discontinuity positions is one of the discontinuity positions of the node cell, and the other one is obtained by the smooth extension of the corresponding discontinuity to the cell boundary. For example, for the critical cell belonging to L_{12}, one of its discontinuity positions is $\eta^n_{i_1-1/2,j_1}$, and for the other one, $\tilde{\eta}^n_{i_1+1/2,j_1}$ is the intersection point of the extension of L_{12} and the vertical grid line $x = x_{i_1+1/2}$ (see also Figure 8). Once the imaginary discontinuity position is obtained, the "ordinary" cell-averages on each critical cell can be computed according to the picture of the discontinuity. Of course, this "ordinary" cell-average has only imaginary meaning.

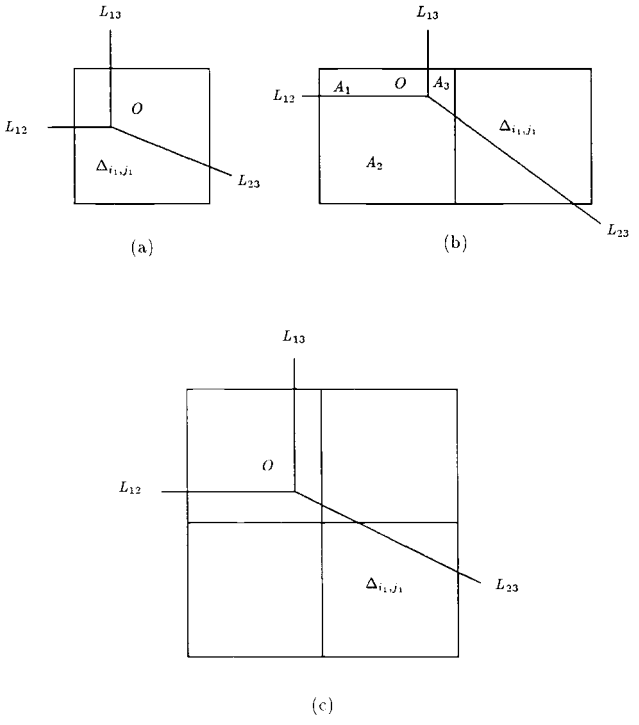


FIG. 9. (a) *O is still in the cell* δ_{i_1,j_1}. (b) *O moves into the cell* δ_{i_1-1,j_1+1}.

The computation in all ordinary and critical cells, including the imaginary critical cells in the node cell, is implemented as described in the previous sections. In doing so, we also compute the numerical fluxes $\tilde{f}^n_{i+1/2,j}$'s and $\tilde{g}^n_{i,j+1/2}$'s defined as in (3.9). On the cell boundaries of the node cell we have three sets of numerical fluxes corresponding to the three overlapping critical cells, which are denoted by $\tilde{f}^{n,12}$ and $\tilde{g}^{n,12}$, $\tilde{f}^{n,13}$ and $\tilde{g}^{n,13}$, and $\tilde{f}^{n,23}$ and $\tilde{g}^{n,23}$, respectively.

To determine the node cell, we first locate the triple point O at t_{n+1}, which is the intersection point of the two incoming discontinuities L_{12} and L_{13}. Under the conventional restriction of the CFL number there can be only three possible cases: (1) O is still in Δ_{i_1,j_1}, (2) O is in either Δ_{i_1-1,j_1} or $\Delta_{i_1.j_1+1}$, and (3) O is in Δ_{i_1-1,j_1+1} (see Figure 9). In the following we are going to discuss the three cases one-by-one.

1. O is still in Δ_{i_1,j_1}. In this case $u^{n+1}_{i_1,j_1}$ is computed from $u^n_{i_1,j_1}$ using numerical fluxes chosen according to the picture of the discontinuities in the node cell. For example, in the case illustrated in Figure 9(a), L_{23} is a y-type discontinuity curve,

$$u^{n+1}_{i_1,j_1} = u^n_{i_1,j_1} - \lambda(\tilde{f}^{n,23}_{i_1+1/2,j_1} - \tilde{f}^{n,12}_{i_1-1/2,j_1}) - \lambda(\tilde{g}^{n,13}_{i_1,j_1+1/2} - \tilde{g}^{n,2}_{i_1,j_1-1/2}). \tag{6.1}$$

In (6.1) $\tilde{g}^{n,2}_{i_1,j_1-1/2}$ is a numerical flux using information only from region 2, which is also $\tilde{g}^{n,23}_{i_1,j_1-1/2}$ or $\tilde{g}^{n,12}_{i_1,j_1-1/2}$ according to the definition of the three sets of numerical fluxes. Δ_{i_1,j_1} is still the node cell on the new time level.

2. O is in Δ_{i_1-1,j_1}. In this case we combine the cells Δ_{i_1,j_1} and Δ_{i_1-1,j_1} and compute $u^{n+1}_{i_1,j_1} + u^{n+1}_{i_1-1,j_1}$ from $u^n_{i_1,j_1} + u^n_{i_1-1,j_1}$ by choosing numerical fluxes on the boundary of the union of the two grid cells according to the picture of the discontinuities, as we compute $u^{n+1}_{i_1,j_1}$ in the previous case. Then we evaluate $u^{n+1}_{i_1-1,j_1}$ according

to the picture of the solution in the grid cell. For example, in the case illustrated in Figure 9(b),

$$u^{n+1}_{i_1-1,j_1} = \frac{1}{h^2}\left(\iint_{A_1} u^1(x,y,t_{n+1})dxdy + \iint_{A_2} u^2(x,y,t_{n+1})dxdy + \iint_{A_3} u^3(x,y,t_{n+1})dxdy\right), \tag{6.2}$$

where u^1, u^2, and u^3 are the solutions reconstructed in the three regions using information only from the same regions and A_1, A_2, and A_3 are the three areas indicated in Figure 9(b). Once $u^{n+1}_{i_1-1,j_1}$ is computed $u^{n+1}_{i_1,j_1}$ can be computed by subtracting $u^{n+1}_{i_1-1,j_1}$ from $u^{n+1}_{i_1-1,j_1} + u^{n+1}_{i_1,j_1}$. Δ_{i_1-1,j_1} is the node cell on the new time level. Discontinuity L_{12} loses a critical cell and L_{23} obtains a critical. The case that O is in Δ_{i_1,j_1+1} can be treated likewise.

3. O is in Δ_{i_1-1,j_1+1}. In this case we combine the four grid cells, Δ_{i_1,j_1}, Δ_{i_1-1,j_1}, Δ_{i_1,j_1+1}, and Δ_{i_1-1,j_1+1}, and compute the sum of the cell-averages on them using the properly chosen numerical fluxes on the boundary of the union of the four grid cells. Then we evaluate $u^{n+1}_{i_1-1,j_1+1}$, $u^{n+1}_{i_1-1,j_1}$, and $u^{n+1}_{i_1,j_1+1}$ according to the picture of the solution in the corresponding grid cells in the same way as $u^{n+1}_{i_1-1,j_1}$ is evaluated in (6.2) in case 2. Once they are computed, $u^{n+1}_{i_1,j_1}$ can be computed by subtracting them from the sum of the four cell-averages. Δ_{i_1-1,j_1+1} is the node cell on the new time level. Both L_{12} and L_{13} lose a critical cell and L_{23} obtain two critical cells.

In cases 2 and 3 we evaluate $u^{n+1}_{i_1,j_1}$ after the evaluation of cell-averages on the other cells, under the consideration that the discontinuity L_{23} is produced by the interaction of L_{12} and L_{13}.

(2) *Reflection point of regular shock reflection.*

We are concerned with the Euler system of gas dynamics in two space dimensions (4.1). A shock travels down a reflection wall to the right with a sufficiently large angle and forms a regularly reflected shock behind it as shown in Figure 10. The dashed lines represent the incident and reflection shocks at t_n and the solid lines represent them at t_{n+1}. The incident and reflection shocks divide the region of solution into three regions denoted by 1, 2, and 3 as shown in the figure.

The wall is taken to be the bottom of the region of solution and a Cartesian grid is set as shown in the figure, on which the wall is set on the line $y = -0.5h$ with h the space increment. We denote by O' and O the reflection points at t_n and t_{n+1} and again call the grid cells containing the reflection points the node cells. In the node cell at t_n the numerical solution has four cell-averages, $u^n_{i_1,0}$, the ordinary cell-average, and $u^{n,1}_{i_1,0}$, $u^{n,2}_{i_1,0}$, and $u^{n,3}_{i_1,0}$, the cell-averages computed using information only from region 1, 2 and 3, respectively, where $(x_{i_1}, 0)$ is the center of the node cell. The node cell has three discontinuity positions: $\eta^n_{i_1+1/2,0}$, belonging to the incident shock; $\eta^n_{i_1-1/2,0}$, belonging to the reflected shock; and $\xi^n_{i_1,-1/2}$, the x-coordinate of the reflection point.

As in the previous case, we define two critical cells overlapping in the node cell, each of which belongs to either the incident or reflected shock. For each critical cell, one of the discontinuity positions is one of the discontinuity positions of the node cell, and the other one is obtained by the smooth extension of the corresponding shock to the cell boundary. For example, for the critical cell belonging to the reflected shock, one of its discontinuity positions is $\eta^n_{i_1-1/2,0}$, and the other one, $\tilde{\eta}^n_{i_1+1/2,0}$, is the intersection point of the extension of the reflected shock and the vertical grid line

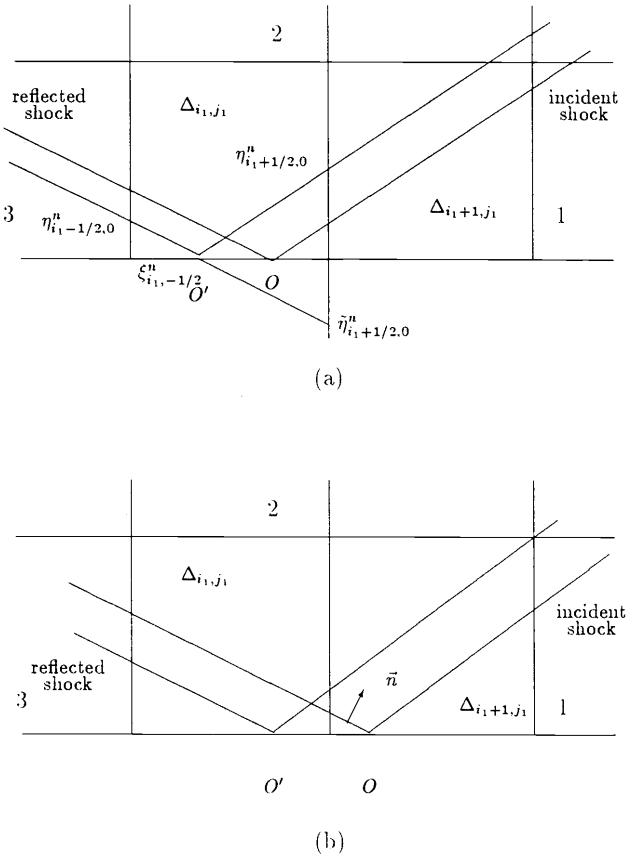


FIG. 10. (a) *The reflection point at t_n is still in δ_{i_1,j_1}.* (b) *The reflection point at t_n+1 moves to the right adjacent cell.*

$x = x_{i_1+1/2}$. As one sees, the imaginary discontinuity position is out of the region of solution. One way to solve this problem is to extend the solution across the wall to define one more critical cell and then define an RC of two cells (see section 3). However, in this paper we consider the RC is still of one cell by viewing part of its interior pulled out by the moving out of one of its discontinuity positions. Once the imaginary discontinuity position is obtained, the imaginary ordinary cell-average on each critical cell can then be computed on the above observation. For example, the imaginary ordinary cell-average in the RC of the reflected shock at t_n can be computed as

$$\frac{1}{h^2}\left\{\left(0.5h - \frac{1}{2}(\eta^n_{i_1-1/2,0} + \tilde{\eta}^n_{i_1+1/2})\right) u^{n,2}_{i_1,0} + \left(\frac{1}{2}(\eta^n_{i_1-1/2,0} + \tilde{\eta}^n_{i_1+1/2}) - 0.5h\right) u^{n,3}_{i_1,0}\right\}.$$

Again as in the previous case, the computation in all ordinary and critical cells, including the imaginary critical cells in the node cell, is implemented as described in the previous sections. The reflection point is determined as the intersection point of the incident shock and the wall. There will be two possible cases for the node cell at t_{n+1}.

1. The reflection point O at t_{n+1} is still in $\Delta_{i_1,0}$ as shown in Figure 10(a). In this case, the cell is still the node cell, and the ordinary cell-average is computed by the

properly chosen numerical fluxes on the boundary of the cell according to the picture of the discontinuities. In particular, the flux on the bottom boundary is evaluated as

$$\begin{aligned}\tilde{g}^n_{i_1,-1/2} &= \frac{\frac{1}{2}(\xi^n_{i_1,-1/2}+\xi^{n+1}_{i_1,-1/2})-x_{i_1-1/2}}{h}\hat{g}^{n,3}_{i_1,-1/2}\\ &\quad+\frac{x_{i_1+1/2}-\frac{1}{2}(\xi^n_{i_1,-1/2}+\xi^{n+1}_{i_1,-1/2})}{h}\hat{g}^{n,1}_{i_1,-1/2}.\end{aligned}\tag{6.3}$$

2. The reflection point O moves to $\Delta_{i_1+1,0}$ as shown in Figure 10(b). In this case $\Delta_{i_1+1,0}$ is the critical cell at t_{n+1}, and the sum of the ordinary cell-averages on the two cells is computed by the properly chosen numerical fluxes on the boundary of the union of the two cells. In this case the fluxes on the bottom boundary are evaluated as

$$\begin{aligned}&\tilde{g}^n_{i_1,-1/2}+\tilde{g}^n_{i_1+1,-1/2}\\ &=\frac{\frac{1}{2}(\xi^n_{i_1,-1/2}+\xi^{n+1}_{i_1,-1/2})-x_{i_1-1/2}}{h}\hat{g}^{n,3}_{i_1,-1/2}+\frac{x_{i_1+3/2}-\frac{1}{2}(\xi^n_{i_1,-1/2}+\xi^{n+1}_{i_1,-1/2})}{h}\hat{g}^{n,1}_{i_1+1,-1/2}.\end{aligned}\tag{6.4}$$

The cell-averages in each region in the two cells should be updated according to the picture of the discontinuities. In particular, $u^{n+1,3}_{i_1+1,0}$ in the new node cell is computed as follows: Extrapolate the numerical solution in region 3 on the y direction and denote the extrapolated data in this cell by v. Solve the Riemann problem $R(v, u^{n+1,2}_{i_1+1,0}, \vec{n})$ and obtain the intermediate state $u^{*,+}$, where $\vec{n}$ is the normal of the reflected shock as shown in the figure, which can be evaluated from the corresponding discontinuity positions. We then define the first, second, and fourth components of $u^{n+1,3}_{i_1+1,0}$ to be that of $u^{*,+}$ and the third component to be 0, considering that the flow on the reflection wall has zero vertical velocity. The ordinary cell-average in the new node cell is computed according to the picture of the discontinuities, and the cell-average in $\Delta_{i_1,0}$ is computed by the requirement that the sum of the cell-averages in the two cells be conserved.

(3) *Foot point of vertical shock on reflection wall.*

A vertical shock travels up on a reflection wall as shown in Figure 11. The shock divides the region of solution into two regions denoted by 1 and 2. We also denote the imaginary region on the other side of the wall by 3. We denote the foot point of the shock on the wall by O.

A Cartesian grid is set such that the reflection wall crosses the grid obliquely. Again, we call the grid cell that contains the foot point O the node cell. We assume that the node cell is at (x_{i_1}, y_{j_1}) as shown in the figure. The ideas used in the previous two cases can still be used in the present case, only we shall deal with partial cell-averages because of the presence of the wall.

First we extend the tracked shock vertically across the wall to the imaginary region and denote the intersection point of the extension and the vertical grid line $x = x_{i_1-1/2}$ by $\tilde{\eta}^n_{i_1-1/2,j_1}$. We then define a yy-type critical cell in the cell with $\eta^n_{i_1+1/2,j_1}$ and $\tilde{\eta}^n_{i_1-1/2,j_1}$ as the two discontinuity positions and with regions 1 and 2 as the two sides. The ordinary cell-average in this critical cell then can be computed according to the picture of the shock. Of course, this ordinary cell-average has only imaginary meaning.

The computation in all ordinary and critical cells, including the imaginary critical cell, is implemented as described in the previous sections. The foot point at the new

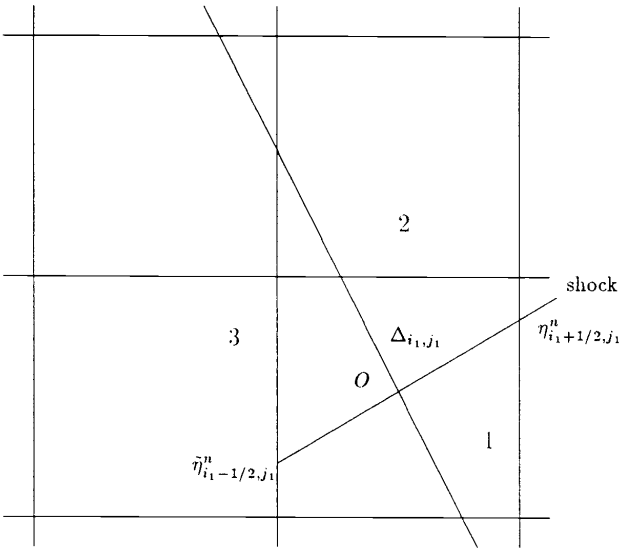


FIG. 11. *The foot point of the vertical shock on the reflection wall is O.* $\eta^n_{i_1+1/2,j_1}$ *is the discontinuity position of the tracked shock and* $\tilde{\eta}^n_{i_1-1/2,j_1}$ *is the imaginary discontinuity position of the tracked on the other side of the wall.*

time is determined as the intersection point of the vertical shock and the wall, which can be computed from the corresponding discontinuity positions. The node cell at the new time is then determined after the foot point is determined.

In the present case partial cell-averages on the node cell should be dealt with instead of the ordinary cell-averages because the node cell is an reflection wall cell. For example, if the foot point is still in Δ_{i_1,j_1} at t_{n+1}, we compute the partial cell-average $u^{n+1,p}_{i_1,j_1}$ which is the approximation to partial cell-average of the exact solution

$$\frac{1}{h^2}\iint_{\Delta^{1,2}_{i_1,j_1}} u(x,y,t_{n+1})dxdy, \tag{6.5}$$

where $\Delta^{1,2}_{i_1,j_1}$ is the part of the grid cell that lies in regions 1 and 2. This cell-average can also be computed by the properly chosen numerical fluxes on the boundary of the cell according to the picture of the shock and the wall boundary. We then reconstruct a piecewise constant function on the part of the cell lying in regions 1 and 2 as in (4.2) and (4.3) with $u^-_{i_1,j_1}$, $u^+_{i_1,j_1}$, and u_{i_1,j_1} replaced by $u^{n+1,1}_{i_1,j_1}$, $u^{n+1,2}_{i_1,j_1}$, and $u^{n+1,p}_{i_1,j_1}$, respectively. We see that the ξ's now still have the geometric meaning that, e.g., $\xi_{m,l} - \xi_l$ represents the area in which the function is valued as $u^{*,-}$. We then update the partial cell-averages in region 1 or region 2 in the way as $u^+_{i_1,j_1}$ is updated in (4.5). Thus the computation is completed.

If the foot point moves out of the original node cell, then we should combine the corresponding cells, as we have done in the previous cases, and implement the computation on the union of the combined cells.

It is seen that the treatments of the node cells in all the three cases maintain the conservation of the numerical solution and does not solve two-dimensional Riemann problems.

7. Numerical examples. In this section we display three numerical examples of the method. Two of them are in scalar conservation law and the third one is a regular shock reflection problem in the Euler system of gas dynamics.

The following scalar conservation law in two space dimensions

$$u_t + \left(\frac{1}{2}u^2\right)_x + \left(\frac{1}{2}u\right)_y = 0 \tag{7.1}$$

is considered. The underlying semidiscrete scheme (3.4) is a second-order TVD scheme of the type described in [33]. The time derivative is discretized by the predictor-corrector method. The recovery of discontinuity curve is of piecewise constant; i.e., it is first-order. Two Riemann problems studied in [37] are tested. Their initial data are defined as

$$u_0(x,y) = \begin{cases} u_1, & x > 0, y > 0, \\ u_2, & x \le 0, y > 0, \\ u_3, & x \le 0, y \le 0, \\ u_4, & x > 0, y \le 0. \end{cases} \tag{7.2}$$

The first problem is with the initial data

$$(u_1, u_2, u_3, u_4) = (-0.2, -1.0, 0.5, 0.8). \tag{7.3}$$

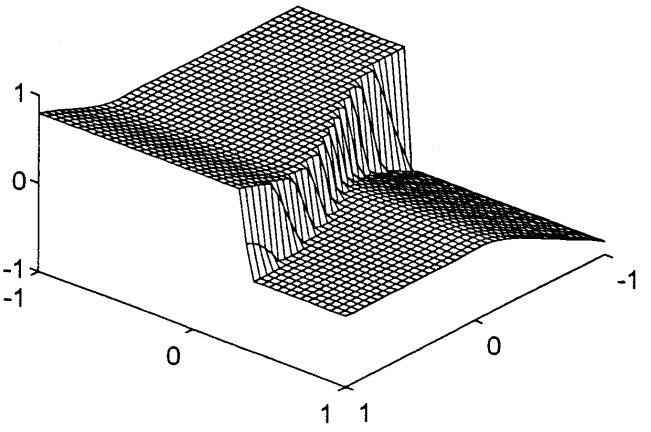
The solution has a y-type shock curve, which has several corner points, the points of discontinuity in first derivative. The test is taken on a grid of 40×40 and the numerical solution is shown in Figure 12. Figure 12(a) represents the surface of the numerical solution and Figure 12(b) represents the contour plot of the solution plus the discontinuity curve. The analysis of the data shows that both the numerical solution and the shock curve have second-order accuracy except at the corner points, where they still have first-order accuracy. We compare the numerical shock to the exact shock and find that they almost coincide. Only near the corner points can small differences be observed in an amplified picture.

The second problem is with the initial data

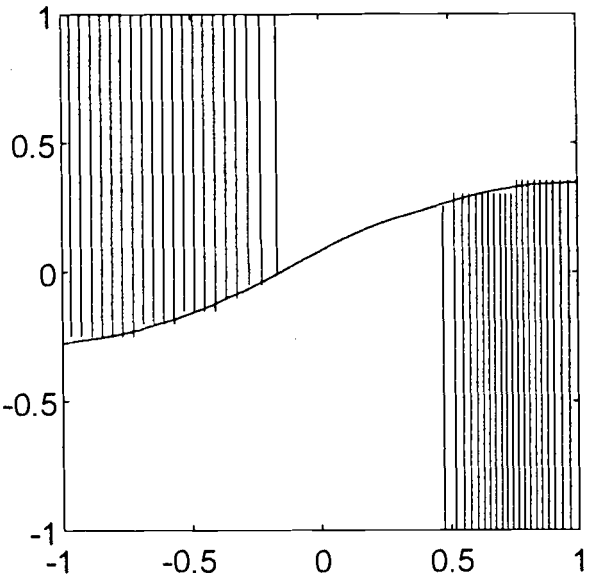
$$(u_1, u_2, u_3, u_4) = (-1.0, -0.3, 0.5, 0.8). \tag{7.3}$$

The solution has two y-type shock curves, an x-type shock curve, and a triple point of the kind discussed in the previous section. The test is again taken on a grid of 40×40 and the numerical solution is shown in Figure 13. Figure 13(a) represents the surface of the numerical solution and Figure 13(b) represents the contour plot of the numerical solution plus the discontinuity curves. We see that one of the y-type shock curves has a sharp corner; however, the numerical results around it still have first-order accuracy and the computation is stable.

Now we consider the Euler equations (4.1). We test our tracking method on a nonsteady regular shock reflection problem and compare the numerical solution with the experimental results in [5]. The Euler system for γ-law gas with $\gamma = 5/3$ is considered. A planar shock with Mach number 2.05 is moving down a shock tube and impinges on a wedge with an angle of 60°. A reflected shock is formed, which extends from the reflection point to the shock tube wall, where it forms a bow shock in front of the wedge, as shown in Figure 14. The solution to this initial value problem is self-similar, i.e., $u(x,t) = u(\sigma t, \sigma x)$ for every σ. The solution consists of three regions denoted by 1, 2, and 3, two shocks of y-type, two RW boundaries, and two node



(a)

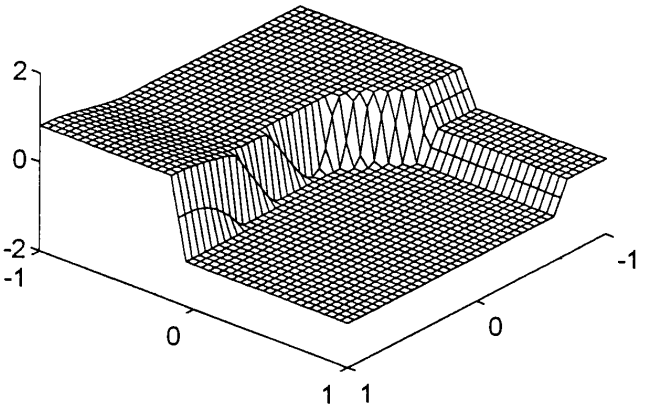


(b)

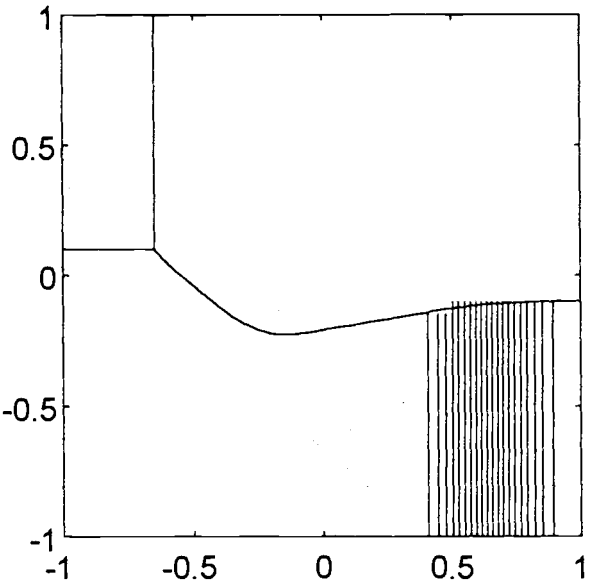
FIG. 12. *The first Riemann problem.* (a) *The surface of the solution.* (b) *The contour plot of the solution plus shock curve.*

points, the reflection point, and the foot point of the reflected shock at the shock tube.

The underlying scheme is still a second-order TVD scheme of the type described in [33], and the recovery of the discontinuity curve is of piecewise constant, so it is first-order. The wedge is set be the bottom of the region of solution as in the second case in the previous section, and the corner point that connects the wedge and the



(a)



(b)

FIG. 13. *The second Riemann problem.* (a) *The surface of the solution.* (b) *The contour plot of the solution plus the shock curves.*

shock tube is located at the grid point $(x_{i_0-1/2}, -1/2h)$, as shown in Figure 15. At the initial time the reflected shock possesses only two xy-type critical cells centered at $(x_{i_0-1}, 0)$ and $(x_{i_0}, 0)$, which are also the two node cells, the cells contain the node points. In these two critical cells, the initial data of the numerical solution in region 3 is set as follows.

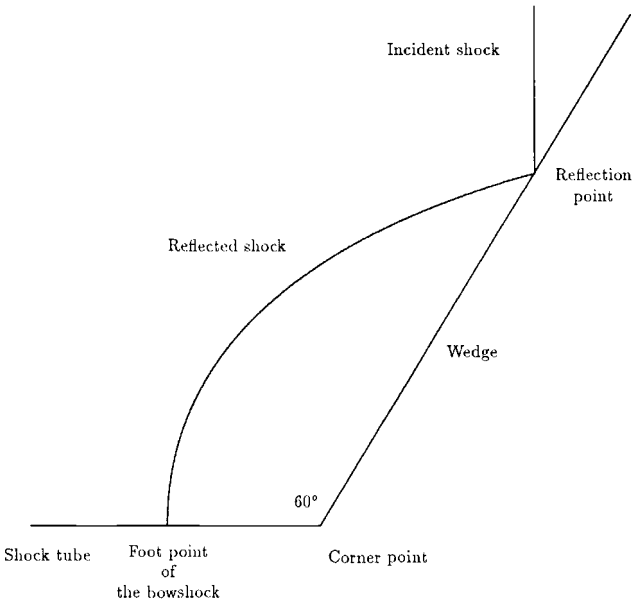


FIG. 14. *A planar shock with Mach number* 2.05 *is moving down a shock tube and impinges on a wedge with an angle of* 60°. *A reflected shock is formed, which extends from the reflection point to the shock tube wall, where it forms a bow shock in front of the wedge.*

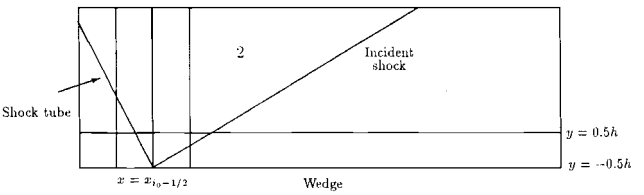
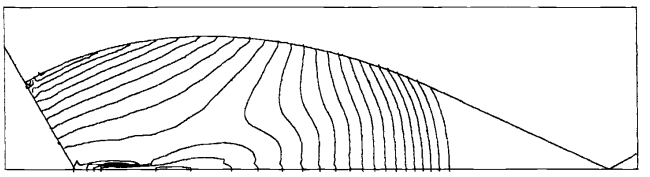


FIG. 15. *The initialization of the numerical experiment is shown as in the figure. The wedge is set to be the bottom of the solution region. The corner point connecting the shock tube and wedge is located at* $(x_{i_0-1/2}, -0.5h)$. *The region* 3, *i.e., the region between the bow shock and the wedge, starts with zero area. This region contains initially two critical cells centered at* $(x_{i_0-1}, 0)$ *and* $(x_{i_0}, 0)$, *respectively. The initial data in these cells are evaluated as described in this section.*

We denote the state in region 2. by u_2 and solve the one-dimensional Riemann problem $R(u_2, \mathcal{R}_{\vec{n}}(u_2), \vec{n})$ in the vertical direction $\vec{n} = (0, -1)$ and obtain a middle state u^*. In the grid cell centered at $(x_{i_0}, 0)$ the initial cell-average is set to be u^* under the consideration that the flow is reflected on the wedge. To define the initial data in the cell centered at $(x_{i_0-1}, 0)$ we solve the Riemann problem $R(u^*, \mathcal{R}_{\vec{n}_1}(u^*), \vec{n}_1)$ in the normal direction of the shock tube $\vec{n}_1$. Then the initial cell-average in the cell is set to be the middle state in the solution to the second Riemann problem under the consideration that the flow in this cell is reflected also by the shock tube.

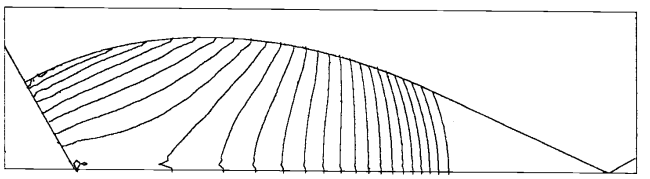
All the initial discontinuity positions of the reflected shock are set to locate at the grid point $(x_{i_0-1/2}, -1/2h)$. This means that the reflected shock is started with a "bubble" of zero area. In such a setting, both the regular cell-averages in the two critical cells are u_2.

As one can see, the above initialization is quite reasonable and arbitrary, in which we do not solve two-dimensional Riemann problems.



DENSITY

(a)



PRESSURE

(b)

FIG. 16. *Numerical results of a regular shock reflection on a grid of* 40×160. *Contour plots of physical quantities in region* 3 *plus shock curves.* (a) *Density.* (b) *Pressure.*

The numerical experiment is carried out on grids of 40×160 (which should be considered quite coarse from the viewpoint of computation). The shocks, node cells, and RW boundaries are treated by the methods described in the previous sections. Figure 16 shows the numerical solution. In the figures, picture (a) presents the contour plot of the density in region 3 plus the discontinuity curve and picture (b) presents that for the pressure. It is seen that the numerical solution agrees quite well with the physical results presented in [5].

The whole computation is quite stable, even at the very beginning. The fronts at the beginning should be considered complex because they have structures of the magnitude of mesh size. However, the small structures are not resolved accurately at this stage of the run, but are resolved roughly by the conservation properties. To show the efficiency of the algorithm in handling complex structures, we display the numerical results at two very early stages. Figure 17 displays the numerical solution at the 25th time step, the bubble of the reflected shock is about two cells in height, and Figure 18 displays the solution at the 60th time step, the bubble is about four cells in height (results are displayed on a grid of 5×20). The computation on the grid of 40×160 took in total 500 time steps. Both the results show that the numerical solution began to show its self-similar form at these early stages.

Figure 19 displays the density distribution along the wall of the numerical solution. There are small oscillations in the distribution on the shock tube part. This is because the cell-averages displayed in this part are not centered at the wall but are on the two sides of the wall with varying distances of $O(h)$.

There are some wiggles behind the reflected shocks in the density and pressure contours. There are two possible reasons for this phenomena. (1) The front tracking

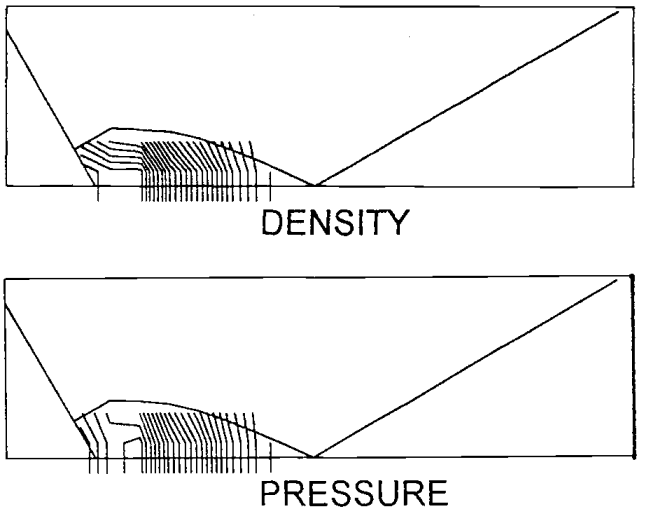


FIG. 17. *The numerical solution at the* 25*th time step; the bubble of the reflected shock is about two cells in height.*

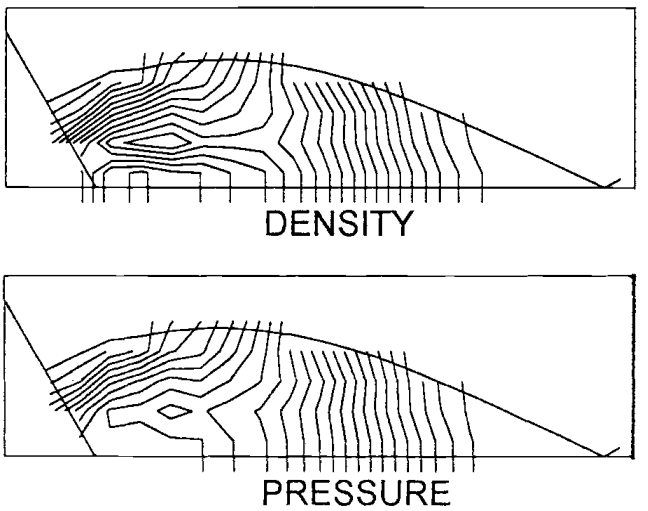


FIG. 18. *The numerical solution at the* 60*th time step; the bubble of the reflected shock is about four cells in height.*

method is only first-order since the reconstruction in the critical cell is first-order; see section 5. Therefore, there might be first-order errors related to the characteristic fields other than that of the tracked reflected shock in the area. (2) The density and pressure contours in region 3 are plotted by MATLAB in the following way. In the cells that are out of a region we set the corresponding cell-averages to be "NaN" and then plot them. In doing so, the plotted region is not regular, neither a rectangle nor a square, but a region with part of its boundary being zigzag lines, which affects the plot quality in this area. Right now we can not tell which reason is more important.

To see the gain of accuracy in using the tracking method, we refer to the last numerical example in [2]. This is of the same problem; however, its reflected shock is

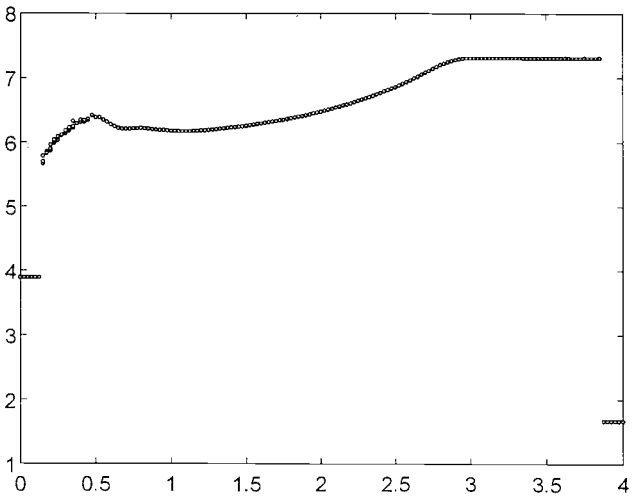


FIG. 19. *The density distribution along the wall of the numerical solution on the grid.*

computed by front capturing method. The example was run on a grid of 100×400. We see that the quality of the smooth part is almost the same as that of our example on the grid of 40×160, but our tracked reflected shock is much sharper than theirs. We also refer to the numerical examples in [11] and [1], which are of a similar problem and are computed by a traditional tracking method, to see the comparison between this tracking method and the traditional tracking methods.

The algorithm is coded in Fortran language. The Fortran used for the numerical experiment is an old version, which only partially supports the Fortran 90 features, with no pointer and module. Therefore, the realization of some data structures are very poor in the code. The numerical experiment on the grid of 40×160 ran for 330 minutes on a PC-586 Pentium 100, not a fast one. The computer does not have the function to check the CPU time consumed in each part; therefore, we cannot tell precisely the CPU time ratio. However, we manually check the CPU times for each part and found that more than 90% of the time was consumed on the computation in the smooth regions in one time step. This confirms the analysis in the end of section 3 that the complexity of the front tracking part is only that of a one-dimensional front capturing method.

Acknowledgments. The author wishes to thank the Graduate Student Office at Shanghai University of Science and Technology for allowing him to use its computer facilities. Part of the numerical experiments reported in section 7 were completed on the PC-386 computer in this office.

REFERENCES

[1] I.-L. CHERN, J. GLIMM, U. MCBRYAN, B. PLOHR, AND S. YANIV, *Front tracking for gas dynamics*, J. Comput. Phys., 62 (1986), pp. 83–110.

[2] I.-L. CHERN AND P. COLELLA, *A Conservative Front Tracking Method for Hyperbolic System of Conservation Laws*, Lawrence Livermore National Laboratory Report UCRL-97200, 1987.

[3] A.J. CHORIN AND J.E. MARSDEN, *A Mathematical Introduction to Fluid Mechanics*, Springer-Verlag, New York, Heidelberg, Berlin, 1993.

[4] P. Colella and P.R. Woodward, *The piecewise-parabolic method for gas dynamics simulation*, J. Comput. Phys., 54 (1984), pp. 174–201.
[5] R.L. Deschambault and I.I. Glass, *An update on non-stationary oblique shock-wave reflection: Actual isopycnics and numerical experiments*, J. Fluid Mech., 131 (1983), pp. 27–57.
[6] H. Forrer, *Boundary Treatment for a Cartesian Grid Method*, Eidgenössische Technische Hochschule Research Report 93-04, 1996.
[7] H. Forrer, *Second Order Accurate Boundary Treatment for Cartesian Grid Methods*, Eidgenössische Technische Hochschule Research Report 96-13, 1996.
[8] J. Glimm, *The interaction of nonlinear hyperbolic waves,* Comm. Pure Appl. Math., 41 (1988), pp. 569–590.
[9] J. Glimm, J. Grove, B. Lindquist, O. McBryan, and G. Tryggvason, *The bifurcation of tracked scalar waves*, SIAM J. Sci. Statist. Comput., 9 (1988), pp. 61–79.
[10] J. Glimm, O. McBryan, R. Menikoff, and D. Sharp, *Front tracking applied to Rayleigh-Taylor instability*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 230–251.
[11] J. Glimm, C. Klingenberg, O. McBryan, B. Plohr, D. Sharp, and S. Yaniv, *Front tracking and two-dimensional Riemann problems,* Adv. Appl. Math., 6 (1985), pp. 259–290.
[12] J. Glimm, J. Grove, X.L. Li, K.M. Shyue, Y. Zeng, and Q. Zhang, *Three-dimensional front tracking*, SIAM J. Sci. Comput., 19 (1998), pp.703–727.
[13] J. Glimm, M.J. Graham, J. Grove, X.L. Li, T.M. Smith, D. Tan, F. Tangerman, and Q. Zhang, *Front tracking in two and three dimensions*, Comput. Math. Appl. 35 (1998), pp. 1–11.
[14] J. Grove, *The interaction of shock waves with fluid interface*, Adv. Appl. Math., 10 (1989), pp. 201–227.
[15] A. Harten, *ENO schemes with subcell resolution*, J. Comput. Phys., 83 (1989), pp. 148–184.
[16] A. Harten, B. Engquist, S. Osher, and S.R. Chakravarthy, *Uniformly high order accurate essentially non-oscillatory schemes,* III, J. Comput. Phys., 71 (1987), pp. 231–303.
[17] A. Harten and S. Osher, *Uniformly high-order accurate nonoscillatory schemes.* I, SIAM, J. Numer. Anal., 24 (1987), pp. 279–309.
[18] W.D. Henshaw, *A scheme for numerical solution of hyperbolic system of conservation laws*, J. Comput. Phys., 68 (1987), pp. 25–47.
[19] C. Klingenberg and D.K. Mao, *The total variation decreasing property of a conservative front tracking technique*, Math. Comput. Modelling, 20 (1994), pp. 89–99.
[20] R.J. LeVeque and K.M. Shyue, *One-dimensional front tracking based on high resolution wave propagation methods*, SIAM J. Sci. Comput., 16 (1995), pp. 348–377.
[21] R.J. LeVeque and K.M. Shyue, *Two-dimensional front tracking based on high resolution wave propagation methods*, J. Comput. Phys., 123 (1996), pp. 354–368.
[22] R.J. LeVeque and M.J. Berger, *Stable Boundary Conditions for Cartesian Grid Calculations*, Report 90-37, ICASE, Hampton, VA, 1990.
[23] R.J. LeVeque, *Numerical Methods for Conservation Laws*, Birkhauser-Verlag, Basel, Boston, Berlin, 1990.
[24] A. Majda, *Compressible fluid flow and systems of conservation laws in several space variables*, Appl. Math. Sci. 53, Springer-Verlag, New York, 1984.
[25] D. Mao, *A shock tracking technique based on conservation in one space dimension*, SIAM J. Numer. Anal., 32 (1995), pp. 1677–1703.
[26] D. Mao, *A treatment of discontinuity for finite difference methods in the two dimensional case*, J. Comput. Phys., 104 (1993), pp. 377–397.
[27] D. Mao, *A treatment of discontinuities for finite difference methods*, J. Comput. Phys., 103 (1992), pp. 359–369.
[28] D. Mao, *A treatment of discontinuities in shock-capturing finite difference methods*, J. Comput. Phys., 92 (1991), pp. 422–455.
[29] D. Mao, *A treatment for discontinuities*, in Proceedings of the Third International Conference on Hyperbolic Problems, Uppsala, Sweden, 1990, B. Engquist and B. Gustafsson, eds., Studentlitteratur, Sweden, 1991.
[30] D. Mao, *A difference scheme for shock calculation*, J. Comput. Math., 3 (1985), pp. 356–382 (in Chinese).
[31] D. Mao', *Entropy satisfaction of a conservative shock tracking method*, SIAM J. Numer. Anal., 36 (1999), pp. 529–550.
[32] G. Moretti, *Thoughts and Afterthoughts About Shock Computations*, Report PIBAL-72-37, Polytechnic Institute of Brooklyn, Brooklyn, NY, 1972.
[33] S. Osher and S. Chakravarthy, *Very high order accurate TVD schemes*, in Oscillation Theory, Computation, and Methods of Compensated Compactness, IMA Vol. Math. Appl. 2, Springer-Verlag, New York, 1986, pp. 229–274

[34] C.-W. Shu and S.J. Osher, *Efficient implementation of ENO schemes* II. J. Comput. Phys., 83 (1987), pp 439–471.

[35] B.K. Swartz and B. Wendroff, *A front tracking code based on Godunov's method*, Appl. Numer. Math., 2 (1986), pp. 385–397.

[36] B. van Leer, *Towards the ultimate conservative difference scheme,* IV. *A new approach to numerical convection*, J. Comput. Phys., 23 (1977), pp. 276–299.

[37] D.H. Wanger, *The Riemann problem in two space dimensions for a single conservation law*, SIAM J. Math. Anal., 14 (1983), pp. 534–559.

[38] P. Woodward and P. Colella, *The numerical simulation of two-dimensional fluid flow with strong shocks*, J. Comput. Phys., 54 (1984), pp. 115–173.

[39] Y.L. Zhu, X.C. Zhong, B.M. Chen, and Z.M. Zhang, *Difference Methods for Initial-Boundary-Value Problems and Flows Around Bodies*, Springer-Verlag, New York, Science Press, Beijing, China, 1980.

SIAM J. SCI. COMPUT.
Vol. 22, No. 1, pp. 152–176
© 2000 Society for Industrial and Applied Mathematics

EFFICIENT NONPARAMETRIC DENSITY ESTIMATION ON THE SPHERE WITH APPLICATIONS IN FLUID MECHANICS*

ÖMER EĞECIOĞLU† AND ASHOK SRINIVASAN‡

Abstract. The application of nonparametric probability density function estimation for the purpose of data analysis is well established. More recently, such methods have been applied to fluid flow calculations since the density of the fluid plays a crucial role in determining the flow. Furthermore, when the calculations involve directional or axial data, the domain of interest falls on the surface of the sphere. Accurate and fast estimation of probability density functions is crucial for these calculations since the density estimation is performed at each iteration during the computation. In particular the values $f_n(X_1), f_n(X_2), \ldots, f_n(X_n)$ of the density estimate at the sampled points X_i are needed to evolve the system. Usual nonparametric estimators make use of kernel functions to construct f_n. We propose a special sequence of weight functions for nonparametric density estimation that is especially suitable for such applications. The resulting method has a computational advantage over kernel methods in certain situations and also parallelizes easily. Conditions for convergence turn out to be similar to those required for kernel-based methods. We also discuss experiments on different distributions and compare the computational efficiency of our method with kernel based estimators.

Key words. probability density, nonparametric estimation, fluid mechanics, convergence, kernel method, efficient algorithm

AMS subject classifications. 65U05, 62G05, 62G07, 65D15, 65Y20

PII. S1064827595290462

1. Introduction.

Nonparametric density estimation is the problem of the estimation of the values of a probability density function, given samples from the associated distribution. No assumption is made about the type of the distribution from which the samples are drawn. This is in contrast to parametric estimation in which the density is assumed to come from a given family, and the parameters are then estimated by various statistical methods. Early contributors to the theory of nonparametric estimation include Smirnov [21], Rosenblatt [16], Parzen [15], and Chentsov [3]. Extensive descriptions of various approaches to nonparametric estimation along with a comprehensive bibliography can be found in books by Silverman [23] and Nadaraya [14]. More recent developments are presented in books by Scott [18] and Wand and Jones [27]. Results of the experimental comparison of some widely used methods appear in [10, 25].

In addition to data analysis, an important application of nonparametric density estimation is in computational fluid mechanics. When the flow calculations are performed in a Lagrangian framework, a set of points in space are evolved through time using the governing equations. In time, points that were initially close can move apart, leading to mesh distortion and numerical difficulties. Problems with mesh distortion can be eliminated to a certain extent by the use of smoothed particle hydrodynamics (SPH) techniques [2, 13, 9, 12]. SPH treats the points being tracked as samples coming from an unknown probability distribution. These calculations often require the computation of the values of not only the unknown density, but its gradient as well.

*Received by the editors August 16, 1995; accepted for publication (in revised form) August 3, 1999; published electronically June 13, 2000.
http://www.siam.org/journals/sisc/22-1/29046.html
†Department of Computer Science, University of California, Santa Barbara, CA 93106 (omer@cs.ucsb.edu).
‡Department of Mathematics, Indian Institute of Technology, Bombay, India (ashok@math.iitb.ernet.in).

In contrast to applications concerned with the display of the density, where it is sufficient to estimate the density on some grid, in these fluid flow calculations the density estimate is required at each sample point. Another difference in these two types of applications is that when dealing with data analysis, one is usually concerned with the optimal accuracy one can get for a given sample size. In fluid flow calculations, where additional "data" can be obtained with increased discretization, one is usually more concerned with the optimal variation of the computational effort as a function of error.

In some applications, for example, in problems involving directional data [24], the samples lie on the unit circle S^1 or along the surface of the unit sphere S^2. A special case of directional data is axial data, in which the density is symmetric about the center of the circle or the sphere, that is, $f(\vec{x}) = f(-\vec{x})$.

Various methods have been proposed for nonparametric density estimation in mathematical statistics, such as the kernel [15, 1, 28] and the orthogonal series methods [17, 11]. The kernel method has been extensively studied, and it is probably the most popular scheme in applications such as SPH. In this method, the value of the density at the point x is estimated as

$$f_n(x) = \frac{1}{nA_h} \sum_{i=1}^{n} K\left(\frac{x - X_i}{h}\right), \tag{1}$$

where f_n is the estimate of the density given a sample of size n, X_i are the positions of the samples drawn from a probability distribution with an unknown density function f, K is a kernel function, h is the window width, and A_h is a normalization factor to make f_n into a probability density. One of the drawbacks of the kernel method is the computational cost involved. Even though it is possible to reduce the cost in the one-dimensional case using the expansion of a polynomial kernel and an updating strategy [19], this strategy cannot be easily extended to higher dimensions [5]. Binning methods [5] can be used in any dimension. However, since the density in this case is evaluated on a uniform grid, this method is not suitable for the fluid flow calculations in which we are interested, where an estimate is required at each sample point.

We propose a cosine-based weight function estimator $c_m(x)$ for nonparametric density estimation, which is a special case of the class of estimators that form a δ sequence [26, 28]. This estimator is similar to the kernel estimator but has the ease of evaluation of a series expansion. The role of the window width parameter h of the kernel method is replaced by a smoothing parameter m in our method, and f_n is now of the form

$$f_n(x) = \frac{1}{n} \sum_{i=1}^{n} c_m\left(x - X_i\right). \tag{2}$$

Our choice of c_m is particularly suitable for applications in fluid flow calculations where the values $f_n(X_1)$, $f_n(X_2)$, ..., $f_n(X_n)$ at the sampled directions themselves are required at each point in each time step in the flow simulation. We show that with this estimator the required n values can be computed efficiently using only $O(m^{1+d}n)$ arithmetic operations for directional data and $O(m^d n)$ arithmetic operations for axial data in d dimensions, where m need not be large as long as it increases without bound with n. This is in contrast to the $O(n^2)$ operations required by the kernel method for this computation in the worst case and an expected complexity of $O(h^d n^2)$ with kernels having bounded support. However, in the special case of $d = 1$, the complexity of the kernel method can be reduced to linear after an initial sorting step.

We derive conditions under which the sequence of estimated density functions f_n constructed in this fashion converge to the unknown density f, and experimentally verify the accuracy and the efficiency of our method in practical test cases. Experiments and theoretical analyses also indicate how m should vary with n for optimal accuracy.

The paper is organized as follows. In section 2 we define the weight function estimator c_m and give the conditions for the convergence of the mean integrated square error (MISE) when the sample space is S^1 (Theorem 3). The conditions guarantee that

$$\int E(f_n(x) - f(x))^2 dx \to 0$$

as $n \to \infty$. We also present corresponding results for S^2. In section 3 schemes for efficient computation of these estimates on S^1 and S^2 are presented. In sections 4 and 5, we describe experimental results with our estimator and compare it with the kernel method for some distributions encountered in practice. Our experiments imply a net savings on the number of operations performed over kernel methods in certain situations and also verify the formula found for the optimal choice of m. The results show that the kernel method and our estimator perform well in different settings, and thus complement each other. The main conclusions are presented in section 6. The appendix contains additional test results.

2. The cosine estimator and the convergence of MISE. In this section, we first mention some related work done on spherical data; then we define our estimator and derive conditions for its convergence for directional data on the circle, and give corresponding results for directional and axial data on the sphere and axial data on the circle.

The kernel method for nonparametric density estimation for directional and axial data is discussed in [6, 8]. While dealing with directional data, Fisher, Lewis, and Embleton [6] recommend using the following kernel:

$$W_n(P, P_i) = \left[\frac{C_n}{4\pi \sinh(C_n)}\right] \exp\left[C_n(x^T X_i)\right]. \tag{3}$$

For axial data they recommend the kernel

$$W_n(P, P_i) = A(C_n) \exp\left[C_n(x^T X_i)\right], \tag{4}$$

where $A(C_n)$ normalizes W to a probability density function, and C_n is the reciprocal of h used in the definition of kernel estimators. x and X_i are the Cartesian representation of points P and P_i, respectively, and $x^T X_i$ is the inner product of these two vectors. $W_n(P, P_i)$ plays the role of $K(x - X_i)$ of (1). Hall, Watson, and Cabrera [8] analyze estimators for directional data with the $x - X_i$ term in (1) replaced by $1 - x^T X_i$. Observe that the term $x^T X_i$ is the cosine of the angle between the points P and P_i, and therefore $1 - x^T X_i$ is a measure of the distance along the surface of the sphere between points P and P_i. Inner product plays a crucial role in these estimators. We consider an estimator that can be written in terms of powers of the inner product, the power playing the role of the smoothing parameter. This enables us to expand the estimator in a series and facilitates fast computation.

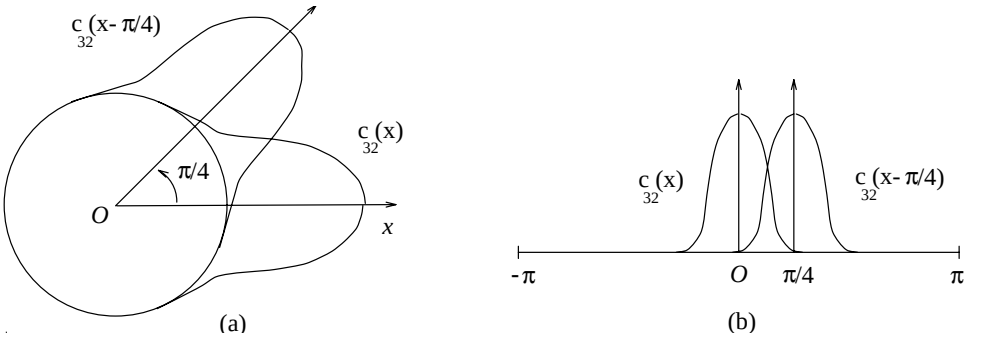


FIG. 1. *The functions* $c_{32}(x)$ *and* $c_{32}(x - \frac{\pi}{4})$ *on* S^1 *and on* $[-\pi, \pi]$.

2.1. The case of S^1. We first define our estimator on S^1. Assume X_j, $j = 1, 2, \ldots, n$, is a sequence of independently and identically distributed (i.i.d.) random variables (observations) for directional data on $[-\pi, \pi]$ with probability density function $f \in C^2[-\pi, \pi]$. We impose the additional condition that $f(-\pi) = f(\pi)$ since the random variables X_j are defined on the unit circle S^1.

As an estimator of the density of directional data $f(x)$, $x \in [-\pi, \pi]$, we consider a nonparametric estimator of the form given by (2) with

$$c_m(x) \;=\; \frac{1}{A_m} \cos^{2m}\left(\frac{x}{2}\right) \tag{5}$$

on $[-\pi, \pi]$. The normalization factor A_m given below makes $c_m(x)$ integrate to 1 on $[-\pi, \pi]$:

$$A_m = \int_{-\pi}^{\pi} \cos^{2m}\left(\frac{x}{2}\right) dx.$$

Making use of a table of integrals such as Gradshteyn and Ryzhik [7] and by using Stirling's formula, it can be shown that

$$A_m \;=\; \frac{\pi}{2^{2m-1}} \binom{2m}{m} \;\sim\; \frac{2\sqrt{\pi}}{\sqrt{m}}\ . \tag{6}$$

As examples, the functions $c_m(x)$ and $c_m(x - \frac{\pi}{4})$ for $m = 32$ are shown on S^1 in Figure 1(a) and on the interval $[-\pi, \pi]$ in Figure 1(b).

We wish to find sufficient conditions under which the sequence of estimators f_n converges to f in the MISE sense. In order to do this, we first show the convergence of the bias and then derive the conditions under which the variance converges to 0. We shall then use these results to prove convergence of MISE on S^1.

First we show that as $m \to \infty$, the expected value of the estimate $f_n(x)$ approaches the actual density $f(x)$ uniformly for any given n.

LEMMA 1. *Suppose* $f \in C^2[-\pi, \pi]$ *and let* $f_n(x)$ *be as given in* (2). *Then* $Ef_n(x) \to f(x)$ *as* $m \to \infty$ *uniformly, independently of* n.

Proof.

$$Ef_n(x) = \int_{-\pi}^{\pi} c_m(x - s) f(s) ds, \tag{7}$$

as shown in Silverman [20] and Whittle [29]. By a change of variable

$$\int_{-\pi}^{\pi} c_m(x-s)f(s)ds = \int_{x-\pi}^{x+\pi} c_m(-y)f(x+y)dy = \int_{x-\pi}^{x+\pi} c_m(y)f(x+y)dy \tag{8}$$

since $c_m(y) = c_m(-y)$. By using the periodicity of c_m and f, along with (7), (8), and the mean value theorem,

$$Ef_n(x) = \int_{-\pi}^{\pi} c_m(y)f(x+y)dy \ = \int_{-\pi}^{\pi} c_m(y)(f(x) + f'(x)y + f''(\xi_x(y))y^2/2)dy,$$

where $\xi_x(y)$ is some point between x and y. Therefore

$$Ef_n(x) = f(x)\int_{-\pi}^{\pi} c_m(y)dy + \int_{-\pi}^{\pi} c_m(y)f'(x)ydy + \int_{-\pi}^{\pi} c_m(y)f''(\xi_x(y))y^2/2dy.$$

From (6), the first integral evaluates to 1, and since $yc_m(y)$ is an odd function, the second integral evaluates to 0. Let $2M' = \max_{x\in[-\pi,\pi]} |f''(x)|$. We then have the following estimate for the bias:

$$\begin{aligned} |Ef_n(x) - f(x)| &\le \left| \int_{-\pi}^{\pi} c_m(y)f''(\xi_x(y))y^2dy \right| \\ &\le M' \int_{-\pi}^{\pi} c_m(y)y^2dy = M' \int_{-\pi}^{\pi} \frac{1}{A_m}\cos^{2m}(y/2)y^2dy. \end{aligned}$$

For any δ such that $0 < \delta \le \pi$,

$$\begin{aligned} |Ef_n(x) - f(x)| &\le M' \int_{|y|<\delta} \frac{1}{A_m}\cos^{2m}(y/2)y^2dy \ + \ M' \int_{|y|\ge\delta} \frac{1}{A_m}\cos^{2m}(y/2)y^2dy \\ &\le M'\delta^2 \int_{|y|<\delta} \frac{1}{A_m}\cos^{2m}(y/2)dy \ + \ \frac{2\pi^3 M'}{3}\frac{1}{A_m}\cos^{2m}(\delta/2) \end{aligned} \tag{9}$$

since $\cos y$ decreases as $|y|$ increases on the interval under consideration. Furthermore, the integral in (9) is bounded above by 1. Therefore,

$$\begin{aligned} |Ef_n(x) - f(x)| &\le M'\delta^2 \ + \ \frac{2\pi^3 M'}{3} \ \frac{\cos^{2m}(\delta/2)}{A_m} \\ &\le M'\delta^2 \ + \ \frac{2\pi^3 M'}{3} \ \frac{\left(1 - \delta^2/8 + \delta^4/384\right)^{2m}}{A_m} \\ &= M'\delta^2 \ + \ \frac{2\pi^3 M'}{3} \ \frac{\left(1 - \delta^2(1 - \delta^2/48)/8\right)^{2m}}{A_m}. \end{aligned} \tag{10}$$

In order to get a bound, we will choose δ as a function of m. If we take $\delta \to 0$ as $m \to \infty$, then $\left(1 - \delta^2(1 - \delta^2/48)/8\right)^{2m} \to \exp(-m\delta^2(1 - \delta^2/48)/4)$. Observe that if $m\delta^2 \to \infty$ as $m \to \infty$, then this term decays exponentially. The second expression in (10) is the product of this term and $1/A_m = O(\sqrt{m})$, and thus the product approaches 0 since the exponential decay dominates. In order to get a good bound on the first term of (10), we wish to choose δ satisfying the condition that $m\delta^2 \to \infty$ such that the δ^2 is as small as possible. We can choose $\delta = 1/m^{\frac{1}{2}-\epsilon}$, where $\epsilon > 0$ is arbitrarily small. Thus M'/m is an asymptotic bound on the bias. Furthermore, the bound is independent of x; hence, the convergence is uniform. ∎

LEMMA 2. *Suppose* $f \in C^2[-\pi,\pi]$ *and let* $f_n(x)$ *be as given in* (2). *Then* $var\ f_n(x) \to 0$ *uniformly as* $n \to \infty$, *provided* $m \to \infty$ *as* $n \to \infty$, *and* $m = o(n^2)$.

Proof.

$$var\ f_n(x) = \frac{1}{n}\int_{-\pi}^{\pi} c_m(x-s)^2 f(s)ds - \frac{1}{n}\left\{\int_{-\pi}^{\pi} c_m(x-s)f(s)ds\right\}^2$$

as shown in Whittle [29]. As a consequence of Lemma 1, the second integral approaches $f(x)$ asymptotically, and hence, the second term approach 0 since f is bounded. It thus suffices to show the convergence of the first integral to 0.

Let $M = \max_{x\in[-\pi,\pi]} |f(x)|$. As in Lemma 1, making a change of variable $y = x - s$ and using periodicity as in (8), we get

$$\frac{1}{n}\int_{-\pi}^{\pi} c_m(y)^2 f(x+y)dy \ \le\ \frac{MA_{2m}}{nA_m^2} \to \frac{M}{\sqrt{8\pi}}\frac{\sqrt{m}}{n},$$

where the expression on the right-hand side is a consequence of the asymptotic expression for A_m in (6). Since $m/n^2 \to 0$ as $n \to \infty$, the above integral converges to 0. Since m is independent of x, the convergence is uniform. Therefore the variance of $f_n(x)$ converges uniformly to 0 under the conditions of the lemma. □

Note that the bound on the bias for the cosine method given by Lemma 1 is of the form

$$|Ef_n(x) - f(x)| \le \frac{1}{2}\max|f''(x)|/m, \tag{11}$$

and the bound for the variance given by Lemma 2 is

$$var\ f_n(x) \le \frac{1}{\sqrt{8\pi}}\max|f(x)|\sqrt{m}/n\ . \tag{12}$$

Therefore, the role played by m for the cosine method is the same as h^{-2} of the kernel based methods, where h is the window width of the kernel estimator. In other words when $m \sim h^{-2}$, the bounds on the bias and the variance of the cosine estimator are in accordance with the asymptotic behavior of the kernel method found in Silverman [23]. Such similarity of rates of convergence is to be expected since the cosine estimator is essentially like the kernel estimator, though the forms of the functions differ. It will be shown later that the main advantage of the cosine estimator lies in its computational efficiency.

THEOREM 3. *Suppose* $f \in C^2[-\pi,\pi]$ *and* $f_n(x)$ *is as given in* (2). *If* $m \to \infty$ *as* $n \to \infty$, *and* $m = o(n^2)$, *then*

$$\text{MISE} \ =\ \int_{-\pi}^{\pi} E(f_n(x) - f(x))^2 dx \to 0$$

as $n \to \infty$.

Proof.

$$\int E(f_n(x) - f(x))^2 dx = \int (Ef_n(x) - f(x))^2 dx + \int var\ f_n(x)dx$$

as shown in Whittle [29]. From Lemmas 1 and 2, each of the integrals approaches 0. Hence, the MISE converges to 0. □

In fact, the MISE is of the form

$$\text{MISE} = O(1/m^2) + O(\sqrt{m}/n), \tag{13}$$

where the asymptotic bounds on the constants are as given in (11) and (12). However, as we shall explain later, the exact asymptotic constants are not all that important for practical computations.

Conditions for the convergence of the estimates and its derivatives on the real line instead of S^1 can be found in [4]. Next we consider the case when the directional data lie along the surface of the unit sphere.

2.2. The case of S^2. Let $X_j,\ j = 1, 2, \ldots, n$, be a sequence of i.i.d. random variables (observations) with values on the surface of the unit sphere S^2 centered at the origin in $\mathbb{R}^3$. Suppose that the probability density function $f(x)$ of the X_j has bounded second derivatives. We consider a nonparametric estimator of the form

$$f_n(x) = \frac{1}{n}\sum_{i=1}^{n} c_m(x, X_i) \tag{14}$$

for some m to be determined as a function of n. The c_m are defined in this case as follows. If α_{xX} denotes the angle between points x and X, then

$$c_m(x, X) \ = \ \frac{1}{A_m}\cos^{2m}\left(\frac{\alpha_{xX}}{2}\right). \tag{15}$$

The normalizing factor A_m is given below:

$$A_m \ = \ \frac{4\pi}{m+1}\ .$$

Through a derivation along the lines of the case of the circle, the following theorem can be proved for the convergence of the estimators.

THEOREM 4. *Suppose $f \in C^2(S^2)$ and let $f_n(x)$ be as given in* (14). *If $m \to \infty$ as $n \to \infty$ and $m = o(n)$, then*

$$\text{MISE} \ = \ \int E(f_n(x) - f(x))^2 dx \to 0.$$

Analogous to (13), the form of the MISE is found to be

$$\text{MISE} = O\left(\frac{1}{m^2}\right) + O\left(\frac{m}{n}\right). \tag{16}$$

From this expression for MISE we see that as in the case of S^1, $m \sim h^{-2}$, where h is the window width of the kernel estimator.

When dealing with axial data, we can consider the following axial estimator for spherical data:

$$c_m(x, X) \ = \ \frac{1}{A_{2m}}\cos^{2m}(\alpha_{xX}).$$

We can also define a corresponding estimator on the circle, where we take the cosine of the arc length between two points, instead of the cosine of half the arc length as in the case of directional data. The relationship between the asymptotic MISE, m and n, is the same as in (13) and (16) for the cases of the circle and the sphere, respectively.

3. Efficient evaluation of the density estimator. In this section, we shall describe an efficient algorithm for the computation of the density estimates $f_n(x)$ evaluated at a set of n observed points $X_1, X_2, \ldots, X_n$ on the unit circle S^1 ($d = 1$ case) and on the unit sphere S^2 ($d = 2$ case). We also show that if the value of f_n at some arbitrary x is desired, then this is also easily accomplished once $f_n(X_1), f_n(X_2), \ldots, f_n(X_n)$ have been computed. The efficiency of our method is based on the fact that $f_n(x)$ is defined in terms of the functions $c_m(x)$ as given in (5) or (15).

Suppose we represent the positions of the observed points $X_1, X_2, \ldots, X_n$ by their Cartesian coordinates. We show that for any x, $f_n(x)$ can be expressed as a polynomial of total degree m in the coordinates of x. The coefficients of this polynomial can be determined in turn from the coordinates of the X_i. Moreover, the coefficients are the sums of the contributions due to each X_i independently.

First we consider the case of directional data on S^1. From (2), (5), and the half-angle formula for cosine we get

$$f_n(x) = \frac{1}{nA_m} \sum_{i=1}^{n} \left(\frac{1 + \cos(x - X_i)}{2} \right)^m . \tag{17}$$

Denote the points on S^1 corresponding to the angles x and X_i for $i = 1, 2, \ldots, n$, by $\vec{x} = (x_1, x_2)$ and $\vec{X}_i = (X_{i1}, X_{i2})$ in Cartesian coordinates and let $\langle , \rangle$ represent the standard inner product on $\mathbb{R}^2$. Then $\cos(x - X_i) = \langle \vec{x}, \vec{X}_i \rangle$. Substituting this into (17) we get

$$f_n(x) = \frac{1}{n2^m A_m} \sum_{i=1}^{n} \left(1 + \langle \vec{x}, \vec{X}_i \rangle \right)^m = \frac{1}{n2^m A_m} \sum_{i=1}^{n} \left(1 + x_1 X_{i1} + x_2 X_{i2} \right)^m . \tag{18}$$

The expression (18) is a polynomial of degree m in x_1 and x_2. For a fixed m, we can compute the coefficients by adding the contribution of each X_i as follows. Using the multinomial theorem and (18)

$$f_n(x) = \frac{1}{n2^m A_m} \sum_{i=1}^{n} \sum_{r+s \le m} \binom{m}{r, s} x_1^r x_2^s \, X_{i1}^r X_{i2}^s, \tag{19}$$

where the inner summation is over all $r + s \le m$ with $r, s \ge 0$. Changing the order of summation

$$f_n(x) = \frac{1}{2^m A_m} \sum_{r+s \le m} \binom{m}{r, s} M(r, s) \, x_1^r x_2^s, \tag{20}$$

where

$$M(r, s) = \frac{1}{n} \sum_{i=1}^{n} X_{i1}^r X_{i2}^s,$$

and A_m is as given by (6). If we use the asymptotic expression for A_m in (6) for computational ease, then (20) simplifies to

$$f_n(x) = \frac{\sqrt{m}}{2^{m+1}\sqrt{\pi}} \sum_{r+s \le m} \binom{m}{r, s} M(r, s) \, x_1^r x_2^s \tag{21}$$

TABLE 1
Computational complexity of the cosine estimator.

	Circle ($d = 1$)	Sphere ($d = 2$)
Axial data	mn	m^2n
Directional data	m^2n	m^3n

for large m. Now we consider the number of operations required for the evaluation of $f_n(x)$ given the observations $X_1, X_2, \ldots, X_n$. The powers X_{i1}^r and X_{i2}^r for a fixed i and $r = 1, 2, \ldots, m$ can be computed with $O(m)$ multiplications. Doing this for $i = 1, 2, \ldots, n$ requires $O(mn)$ multiplications. After the conclusion of this step, each of the $O(m^2)$ averages $M(r, s)$ for a given r and s can be computed with an additional $O(n)$ operations. Since there are a total of $O(m^2)$ terms in $f_n(x)$ corresponding to the pairs r, s with $0 \leq r, s, \leq m$, this means that the coefficients of the polynomial in (20) or (21) can be computed with a total of $O(m^2n)$ arithmetic operations.

Once the coefficients of $f_n(x)$ have been computed, to evaluate $f_n(x)$ with $\vec{x} = (x_1, x_2)$ we calculate the powers x_1^r and x_2^r for $r = 1, 2, \ldots, m$ in $O(m)$ operations. Since the coefficients are already available, the remaining computation in (20) requires only an additional $O(m^2)$ multiplications and additions. The results for different cases are summarized in Table 1.

Remark. For our MISE convergence condition for S^1 (Theorem 3) to be satisfied, m must increase without bound with n. Theoretically, we can take m to be as slowly increasing as we like. Then the above result implies that the computation of the density at all of the sample points can be accomplished using only about $O(n)$ arithmetic operations if the magnitude of m gives acceptable accuracy for $f_n(x)$. The problem with taking m to be too slowly growing is that the magnitude of m controls the error terms in our convergence proofs.

An efficient algorithm for the evaluation of $f_n(\vec{x})$ for directional data on S^2 is constructed similarly. When $\vec{X}_1, \vec{X}_2, \ldots, \vec{X}_n$ are observations on S^2 drawn from an unknown density, it can be shown [4] that

$$f_n(x) = \frac{m+1}{\pi 2^{m+2}} \sum_{r+s+t \leq m} \binom{m}{r, s, t} M(r, s, t)\, x_1^r x_2^s x_3^t, \tag{22}$$

where the summation is over all $r + s + t \leq m$ with $r, s, t \geq 0$, and

$$M(r, s, t) = \frac{1}{n} \sum_{i=1}^{n} X_{i1}^r X_{i2}^s X_{i3}^t \ .$$

This time the coefficients of the polynomial in (22) can be computed with a total of $O(m^3n)$ arithmetic operations. After this preprocessing, each evaluation of $f_n(x)$ for $x \in S^2$ requires only an additional $O(m^3)$ operations.

Corresponding results can be derived for axial data as summarized in Table 1.

It should also be noted that we needed the Cartesian representation of the data. If the data are given in spherical coordinates, then there will be an additional overhead for obtaining the Cartesian representation. However, this overhead takes only linear time and so will be negligible for sufficiently large data. Furthermore, it has been shown [24] that for an important class of applications, Cartesian coordinates are preferable to spherical coordinates, as the latter system is not numerically stable for solving the differential equations that arise.

In the subsequent parts of this section we shall compare the computational efficiency of our scheme with that of the kernel method.

3.1. Parallelization. One of the advantages of the computational strategy described above is the ease of parallelization. Parallelization is required in many fluid flow calculations due to the large sizes of the systems. The kernel method is somewhat difficult to parallelize. If we use an efficient kernel implementation that performs kernel evaluations only for those points which are within a cut-off distance h of the given sample, then an efficient implementation of the parallelization requires periodic load balancing and domain decomposition so that points that are close by remain on the same processor, and so that each processor has roughly the same load in terms of the computational effort. Also, the communication pattern for the kernel method is not very regular. In contrast, parallelization for the cosine estimator can easily be accomplished by a global reduction operation, for which efficient implementations are usually available. This method requires the same computational effort for each point, and so the load is easily balanced by having the same number of points in each processor. Domain decomposition does not play an important role in this scheme, since the points can be on any processor.

3.2. Theoretical comparison of the kernel and the cosine estimators. Now we analyze the computational efficiency of the kernel and the cosine density estimation methods. An important measure of the efficiency of the algorithms is not just the convergence rate of the error with sample size n, but the relationship of the computational effort C required as a function of the error E. For the kernel estimator, we can write the asymptotic MISE E as

$$E = O\left(h^4 + \frac{1}{h^d n}\right), \tag{23}$$

where h is the smoothing parameter, $d = 1, 2$ is the dimension, and n is the sample size. The computational effort required for this nonparametric density estimation can be expressed as

$$C = O(h^\beta n^\gamma), \tag{24}$$

where $0 \le \beta \le d$ and $\gamma = 1$ or 2 depending on the details of the algorithm used. For a given sample size, (23) gives the optimal h as $h \sim n^{-1/(d+4)}$. However, since the equation for the computational complexity also depends on h, we need to consider the possibility that a value of h smaller than this optimal value may actually result in a lower computational effort.

Let us consider a variation of h with n of the form

$$h = O(n^{-\alpha}), \quad 1/(d+4) \le \alpha < 1/d\ . \tag{25}$$

From (25), (24), and (23) we obtain

$$C = n^{-\alpha\beta+\gamma} \Rightarrow n = C^{\frac{1}{\gamma-\beta\alpha}} \Rightarrow E = C^{\frac{-4\alpha}{\gamma-\beta\alpha}} + C^{\frac{d\alpha-1}{\gamma-\beta\alpha}}.$$

For minimum error, the exponent of both the terms on the right should be the same, otherwise the error due to the higher term will dominate. This leads to $\alpha = 1/(d+4)$ which is the same as the value of optimal α for a given n. If we let h_{optn} represent the optimal h minimizing the MISE for a given n, and h_{optC} represent the optimal

h minimizing the computational effort as a function of the error, then the expression derived above for α does not necessarily imply that $h_{optn} = h_{optC}$ since the relation $h_{optn} = kh_{optC}$ would still satisfy the expression for α for some constant k. If $k < 1$, then we can choose a suboptimal value of h in order to improve the speed of the algorithm.

The optimal variation of error with computational complexity using this value of α is given by

$$C = E^{\frac{\beta}{4} - \frac{(d+4)\gamma}{4}}.$$

Let us now consider the cosine estimator. We can write the asymptotic MISE as follows:

$$E = O\left(m^{-2} + \frac{m^{d/2}}{n}\right)$$

where E is the MISE, m is the smoothing parameter, d is the dimension ($d = 1$ for the circle and $d = 2$ for the sphere), and n is the sample size. The computational effort required for this estimator can be expressed as $C = O(m^{\xi} n)$, where ξ can be determined from Table 1. The expression for C above is the same as (24) with $\gamma = 1$, and $\beta = -2\xi$, recalling that m behaves as h^{-2}. By an analysis similar to the previous case we can show that the optimal variation of error with computational complexity is given by

$$C = E^{\frac{-(d/2+\xi)}{2} - 1}.$$

As examples, for the cosine estimator on the circle ($d = 1$) with axial data ($\xi = 1$) and directional data ($\xi = 2$), the computational complexity and the error are related by $C = E^{-1.75}$ and $C = E^{-2.25}$, respectively.

The complexity of the kernel estimator is the same for axial and directional data. However, several different possibilities exist depending on how efficient the implementation of the estimator is. If we consider estimators of the form given by (3) and (4), then we have $d = 1$, $\beta = 0$, $\gamma = 2$, and therefore

$$C = E^{-2.5}. \tag{26}$$

However, if we consider a kernel with bounded support, and use an efficient implementation of the algorithm that computes the kernel only for those points that have a nonzero contribution, then the *expected* value of β is 1 and we get $C = E^{-2.25}$ for data on the circle. Note that the worst case complexity remains as in (26). For the $d = 1$ case, we can consider an efficient algorithm using polynomial kernels and updating [19], which uses a linear time after an initial $O(n \log n)$ sorting step. In this case $\beta = 0$, $\gamma = 1$, and $C = E^{-1.25}$, which means that the kernel method has a better asymptotic complexity than the cosine kernel. However there appears to be no natural generalization of this update strategy to higher dimensions [5].

Results for the different cases can be determined in the manner demonstrated above and are presented in Table 2. We wish to mention that the exact asymptotic constants in Theorem 3 are not quite that important (compared with the exponent on E), since asymptotically the slowdown incurred by the cache misses dominates the overall running time. We can expect that the simpler memory access pattern of our estimator will make it advantageous over the kernel method in the asymptotic case.

TABLE 2

*Variation of the optimal computational effort versus MISE. The numbers in the table represent η, where the relationship between the computational effort C and $E = MISE$ is given by $C = E^{-\eta}$. *Does not take into account an initial sorting step.*

Estimator	Circle ($d = 1$)	Sphere ($d = 2$)
Cosine, axial data	1.75 ($\xi = 1$)	2.5 ($\xi = 2$)
Cosine, directional data	2.25 ($\xi = 2$)	3.0 ($\xi = 3$)
Kernel, worst case	1.25 * ($\beta = 0$, $\gamma = 1$)	3.0 ($\beta = 0$, $\gamma = 2$)
Kernel, expected case	1.25 * ($\beta = 0$, $\gamma = 1$)	2.5 ($\beta = 2$, $\gamma = 2$)

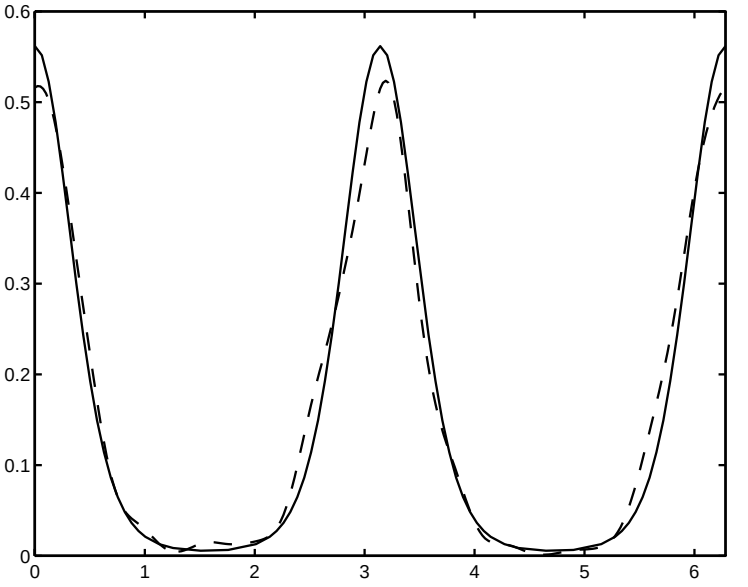
Since the worst case complexities of the kernel method and the cosine estimator for directional data on the sphere have the same order, the relative efficiencies of the methods can be tested only through experiments. Similarly, since the worst case complexity of the cosine estimator for axial data on the sphere is the same as the expected case for an efficient implementation of the kernel estimator, we need to perform experiments to test the relative merits of the two estimators.

4. Experimental results. We performed numerical experiments for axial and directional data on the circle and sphere in order to test the effectiveness of our estimator. We first plot estimates for known distributions and then demonstrate that the MISE follows expected trends for certain distributions. We finally compare the computational efficiency of our estimator with that of kernel methods. More empirical results are presented in the appendix.

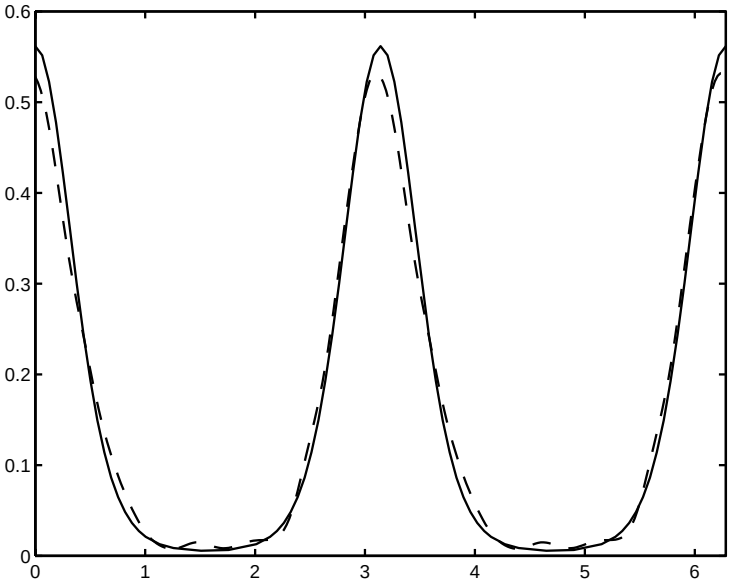
We consider the function $\psi(\phi, \theta) = \exp(US\cos^2(\theta))/A$ where A normalizes the function to be a density on the surface of a sphere and S is a known function of U. The angles ϕ and θ are the azimuth and the elevation in spherical coordinates. This is the solution to a particular problem in fluid mechanics. In Figure 2(a) we present a typical estimate for the $d = 1$ version of the above density function where ψ was taken to be $\psi(x) = \exp(US\cos^2(x))/A$. In this figure, we take the data to be directional. However, since this function is symmetric with respect to the center of the circle, we can consider the data as axial and use the axial estimator. We can see from Figure 2(b) that this requires a much smaller value of m.

In Figure 3(a) the MISE is compared versus m and n for the one-dimensional ψ distribution using the axial cosine estimator. We also compare with one case of the directional estimator in order to show the benefit of using the axial estimator. In Figure 3(b) the MISE is compared versus m and n for the two-dimensional version of the ψ distribution on the surface of a sphere using the directional estimator.

We next present results for experiments comparing the speed of the cosine and the kernel estimators. We consider the optimal variation of the computational effort with MISE. In order to get the optimal computational effort for a given MISE, we allow for the possibility that we may require different sample sizes for the kernel and the cosine estimators. This is justified because in iterative calculations one can easily change the "sample" size by changing the discretization of the system. We have performed these comparisons only for spherical data. The case of data on the circle was not considered because of the asymptotic analyses of the previous section which clearly indicate that the linear kernel algorithm in the one-dimensional case will outperform the cosine estimator. However, in a parallel implementation, the sorting step for the linear kernel algorithm may be slow, and then one may wish to consider the cosine estimator.



(a)

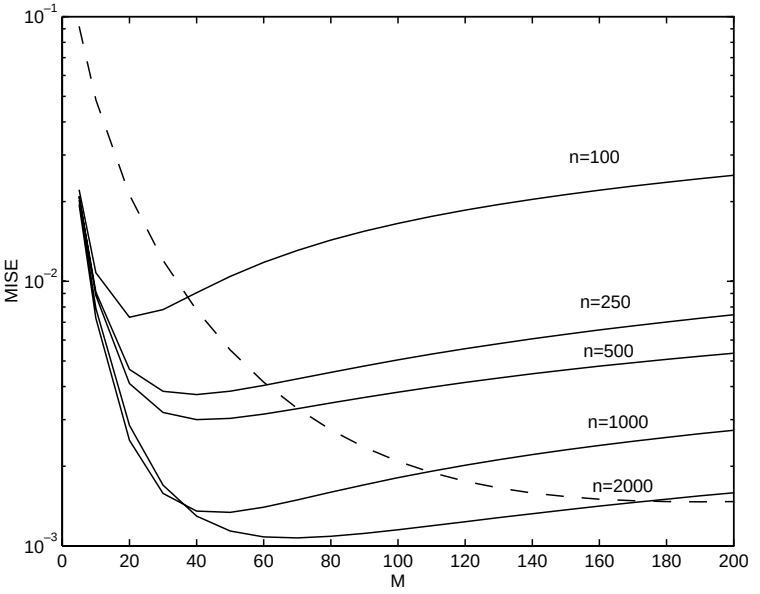


(b)

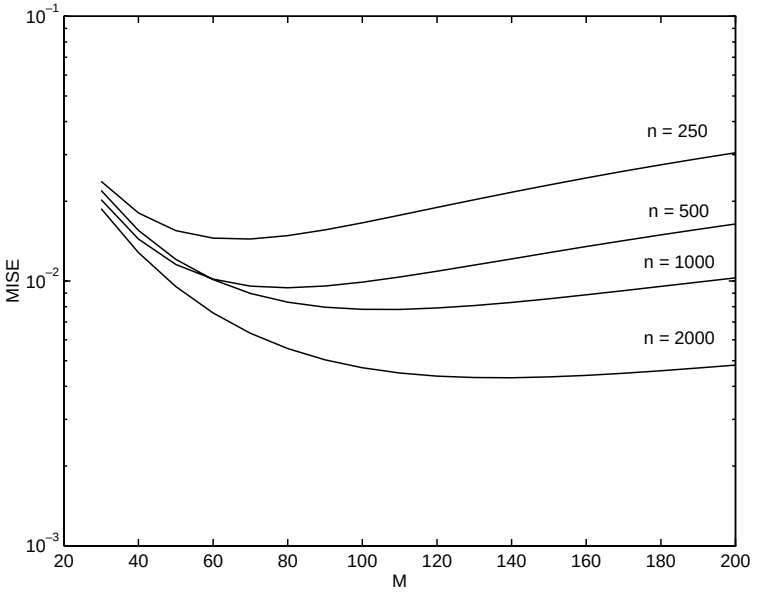
FIG. 2. *Cosine estimates for* $\psi(x) = \exp(US\cos^2(x))/A$*, the one-dimensional case of the* ψ *function defined above. The solid line represents the true density.* (a) *The dashed line represents the directional estimate with* $m = 100$*,* $n = 500$*.* (b) *The dashed line represent the axial estimate with* $m = 40$*,* $n = 500$*.*

The following kernel was chosen for the comparisons:

$$K(x) = \begin{cases} \frac{1}{A}(1 - 1.5x^2 + 0.75x^3), & x \in [0,1], \\ \frac{1}{A}0.25(2-x)^3, & x \in [1,2], \\ 0 & \text{otherwise}, \end{cases}$$



(a)



(b)

FIG. 3. *MISE versus m and n for the ψ density.* (a) *One-dimensional (on the circle): $\psi(x) = \exp(US\cos^2(x))/A$; the solid line shows the results for the axial cosine estimator and the dashed line for the directional estimator (with $n = 2000$).* (b) *Two-dimensional (on the sphere): $\psi(\phi, \theta) = \exp(US\cos^2(\theta))/A$ as defined above; MISE for the directional cosine estimator.*

where A is the normalization constant, and the ratio of the distance between two points along the surface of the sphere to h is given as the argument to the kernel function. The use of this kernel for comparisons can be justified by its popular use

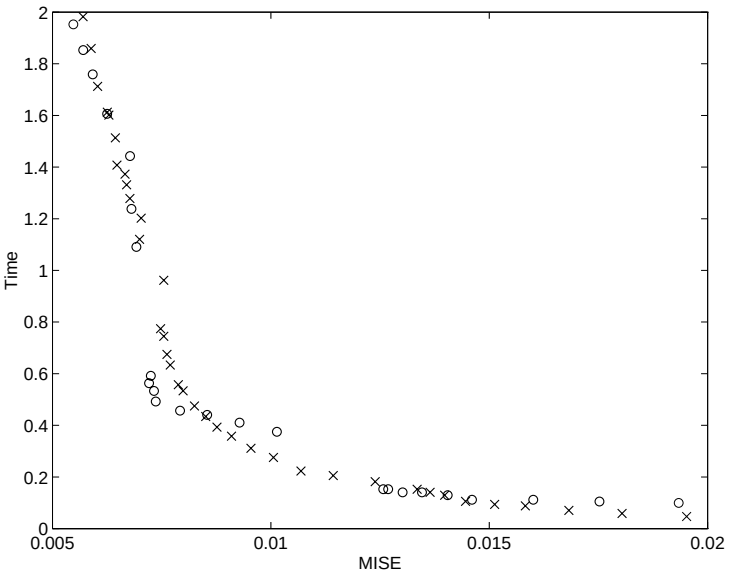


FIG. 4. *Comparison of time (in seconds) versus the MISE for the cosine and kernel estimation of axial data sampled from* $\psi(\phi, \theta) = \exp(US\cos^2(\theta))/A$ *as defined above. The points marked in* o *represent the kernel estimate. The points marked in* x *represent the cosine estimate.*

in fluid mechanics calculations [12]. Furthermore, we cannot expect any other kernel to give a significantly better performance, for the following reasons: (i) It is well known that most kernels are equally good [23, Table 3.1] with respect to "efficiency," as defined in [23]. (ii) Given that the efficiency is about the same, the only other consideration is the computational effort involved. Our kernel takes between 6 and 10 floating point operations for a nonzero evaluation (including the cost of computing the square of the distance). Any other reasonable kernel would require at least 6 floating point operations. Apart from this, the memory access times and zero-evaluations would add the same constant to all kernels.

Figures 4 and 5 compare the computational effort required for the cosine weight function estimator and the kernel estimator for certain densities. We obtained the data using the following procedure. We performed estimates for various values of n, m, and h and obtained the MISE and the time for the calculations. For the cosine and the kernel estimates, we separately plotted the data for the time required for the calculations versus the error. We chose the lower envelope of the data as the curve for that particular estimator since the values of m, n for the data on the lower envelope give the best obtainable speed for a given MISE.

In the implementation of the kernel estimator, we divided the sphere into cells such that the sides of the cells had length at least $2h$ (since the kernel defined above has window width $2h$, rather than h). We placed each sample in the appropriate cell. When computing the density for a point in a particular cell, we need to search over points in only a few cells. The expected complexity of this scheme is $O(n^2h^2)$.

We first consider data estimated using the axial cosine estimator and an axial variant of the kernel estimator. Figure 4 shows the results for the two-dimensional ψ distribution. This is an example of a highly nonuniform distribution. We can see that the kernel and the cosine estimators are about equally fast.

We next consider a more uniform distribution given by $f(\phi, \theta) \propto \cos(\theta) + 1/(8\pi)$, where ϕ is the azimuth and θ is the elevation. The results presented in Figure 5(a) show that the cosine estimator outperforms the kernel estimator by more than an order of magnitude when the axial forms of both the estimators are used.

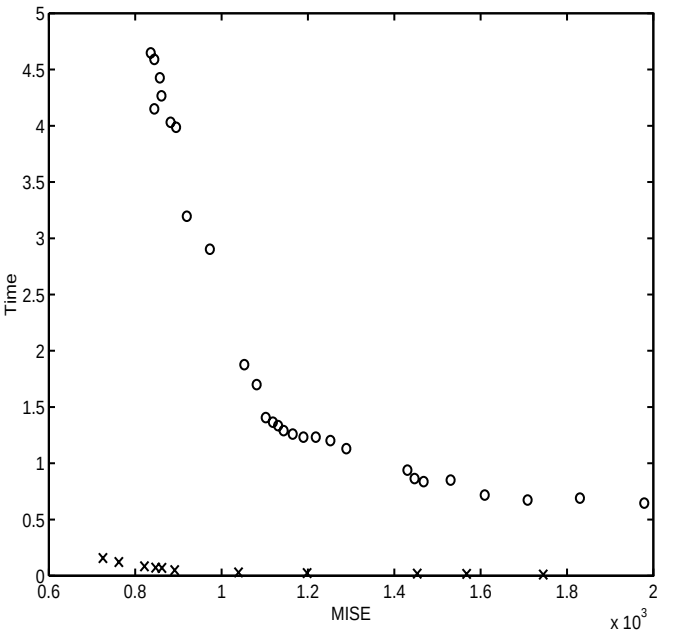
After this we considered the two densities mentioned above but treated the data as directional and estimated them using the directional variants of the kernel and the cosine estimators. For the ψ distribution, the cosine estimator performed very poorly in computational efficiency, and we do not present results for this case. Figure 5(b) presents the results for the distribution given by $f(\phi, \theta) \propto \cos(\theta) + 1/(8\pi)$. We can see that the cosine estimator still outperforms the kernel estimator, though only slightly.

5. Discussion. The comparison of the estimates with the true density indicate that the cosine estimator produces accurate results for the distributions tested. Plots of MISE versus m and n follow the expected trends. As the sample size increases, the error for the optimal m decreases. Besides, the optimal value of m increases as the sample size increases. It can also be seen that as the number of points increases, the range of m over which the estimate performs well also increases. We can use this to our advantage by choosing a suboptimal value of m which decreases the computational effort significantly but increases the error only slightly.

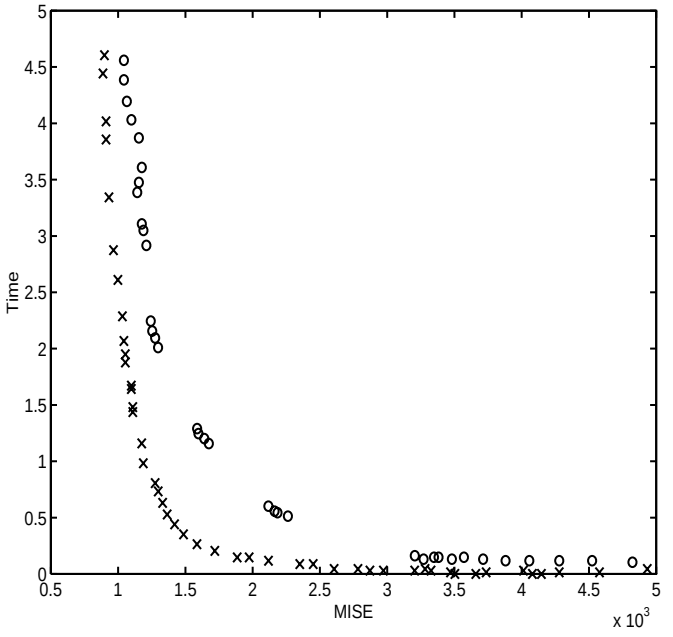
The experiments comparing the computational efficiencies show that the cosine estimator outperforms the kernel estimator for axial data when the distribution is moderately uniform. If the distribution is highly nonuniform, then the two estimators give comparable performance for axial data. The cosine estimator outperforms the kernel estimator slightly for directional data when the distribution is moderately uniform. However, the timing results for the cosine estimator are poor for highly nonuniform directional data. In general, when the data is not very nonuniform, smoother weight functions are used. This gives a low value of m which implies a fast evaluation using the cosine estimator. However, this leads to a higher h for the kernel estimator, which implies that more samples contribute to the kernel evaluation of each sample point and, hence, this leads to more computational effort. Conversely, when the distribution is highly nonuniform, especially for directional data, the kernel method is to be preferred. The empirical test results presented in the appendix further demonstrate this point.

We also analyzed our experimental data to estimate an optimum variation of m with n. Using the results of our experiments, we can perform a least squares analysis and approximate m as $kn^{1/2.5}$ for one-dimensional density estimation which is the same as that expected based on the asymptotic expression for the MISE. Choosing $m = kn^{1/3}$ appears to give reasonable estimates for density on the surface of a sphere. This result is also consistent with the theoretical predictions. Here, the magnitude of k depends on the complexity of the density function. It varies between 1 and 10 for the distributions considered here.

We also noted the values of m, n, and h, which gave the optimal computational effort for a given MISE, and compared the results for the kernel and the cosine estimators. We observed that the values of h were close to the values which gave the minimum MISE for the given sample size. However, the values of m were significantly lower than the values which gave the minimum MISE for the given sample size, though the error involved itself was not much higher than the minimum MISE. Thus, it appears that we can choose a suboptimal smoothing parameter in order to increase the speed in the case of the cosine estimator.



(a)



(b)

FIG. 5. *Plot of time (in seconds) versus the MISE for the cosine and the kernel estimation of data sampled from* $f(\phi,\theta) \propto \cos(\theta) + 1/(8\pi)$. *The points marked in* o *represent the kernel estimate. The points marked in* x *represent the cosine estimate.* (a) *Data treated as axial.* (b) *Data treated as directional.*

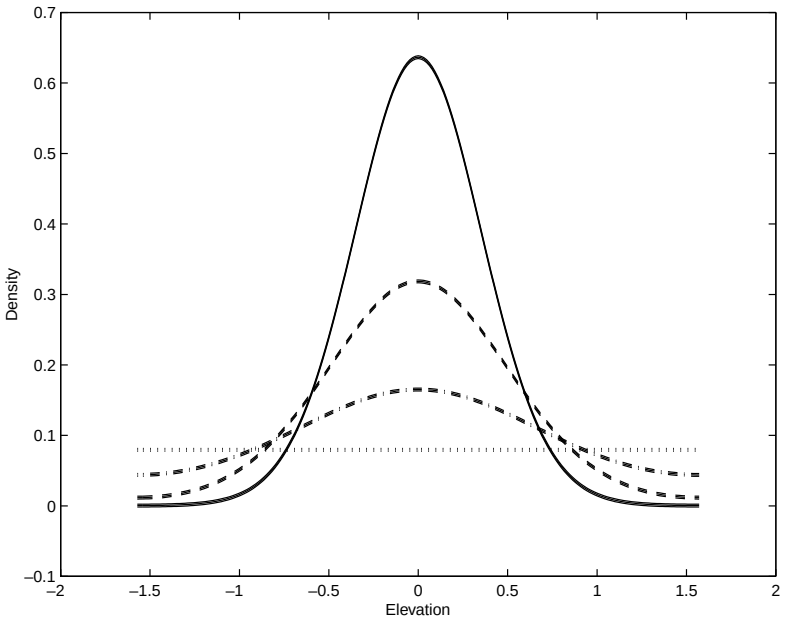


FIG. 6. *Plot of density functions* $g(\phi; s) = k \cosh(s \sin\phi)$ *for different values of* s*, where* ϕ *is the elevation. The solid line denotes* $s = 8.0$*, the dashed line* $s = 4.0$*, the dash-dotted line* $s = 2.0$*, and the dotted line* $s = 0.0$*.*

6. Conclusions. In this paper, we have described a weight function estimator for nonparametric estimation of probability density functions based on cosines, and we provided conditions under which the estimate and its derivatives converge to the actual functions. We have developed a scheme for the efficient computation of the density and presented experimental results to check the performance of the estimator for practical problems. These results are particularly relevant to certain fluid mechanics calculations and, in general, to situations where the sample size can be controlled, for example, though refinement of the discretization. We have also given an empirical formula for choosing the weight function exponent parameter of the estimator. Our experimental results suggest that the cosine estimator outperforms the kernel estimator for both directional and axial data that are moderately uniform. It gives performance comparable to the kernel estimator for highly nonuniform axial data, while the kernel method is preferable for highly nonuniform directional data. There is potential for further theoretical study of our estimator.

Appendix. Further test results. We present more test results in this section in order to study the relative efficiencies of the cosine and the kernel techniques, as the density function is varied systematically from being relatively uniform to being sharply peaked on the unit sphere.

For these tests, we chose density functions $g(\phi; s) = k \cosh(s \sin\phi)$, where s is a constant that governs the sharpness of the density function, ϕ is the elevation, and $k = s/(4\pi \sinh s)$ normalizes this to a probability density function. Figure 6 shows the density as a function of the elevation alone for different values of the parameter s. This function is symmetric about the center of the sphere, and thus we can use the axial estimators, as in section 4. We can also ignore our knowledge of this symmetry and use the general directional estimators.

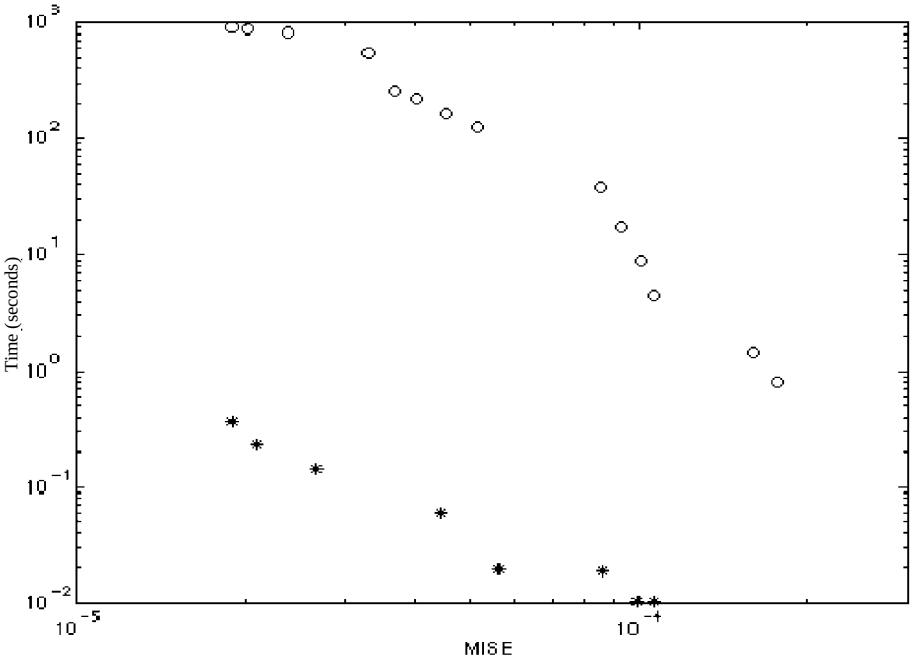
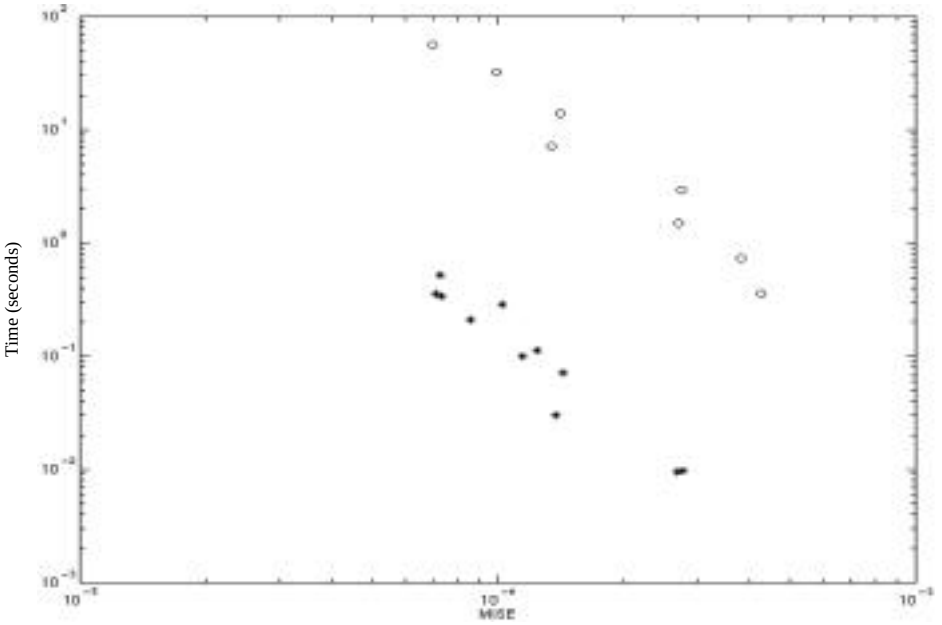
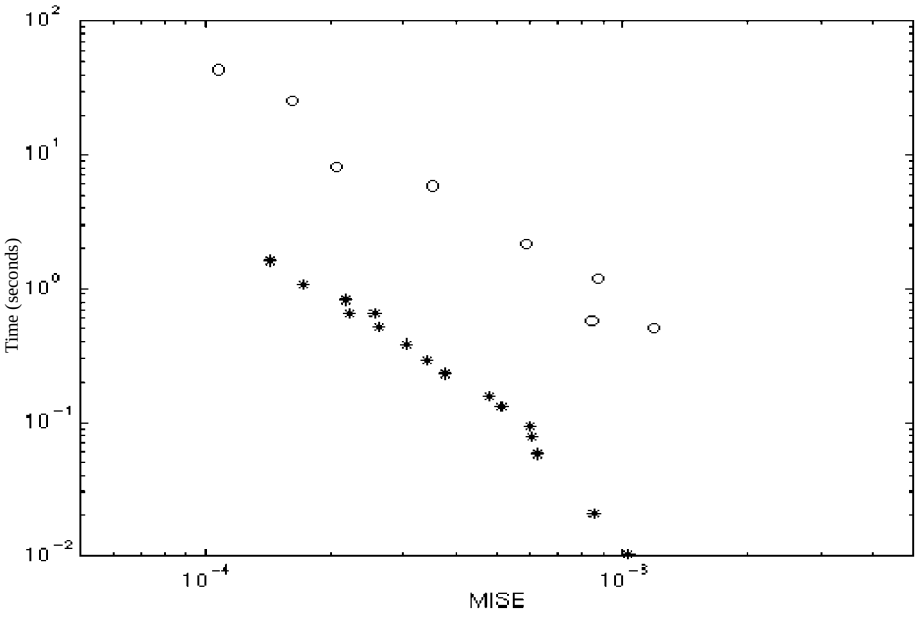


FIG. 7. *Plot of MISE versus time for the density function* $g(\phi; 0.5) = k \cosh(0.5 \sin\phi)$. *The points marked by* * *are for the cosine estimator, and the points markeds by* o *are for the kernel estimator.* (a) *Axial estimator.* (b) *Directional estimator.*



(a)



(b)

FIG. 8. *Plot of MISE versus time for the density function* $g(\phi; 1.0) = k \cosh(1.0 \sin\phi)$. *The points marked by* * *are for the cosine estimator, and the points markeds by* o *are for the kernel estimator.* (a) *Axial estimator.* (b) *Directional estimator.*

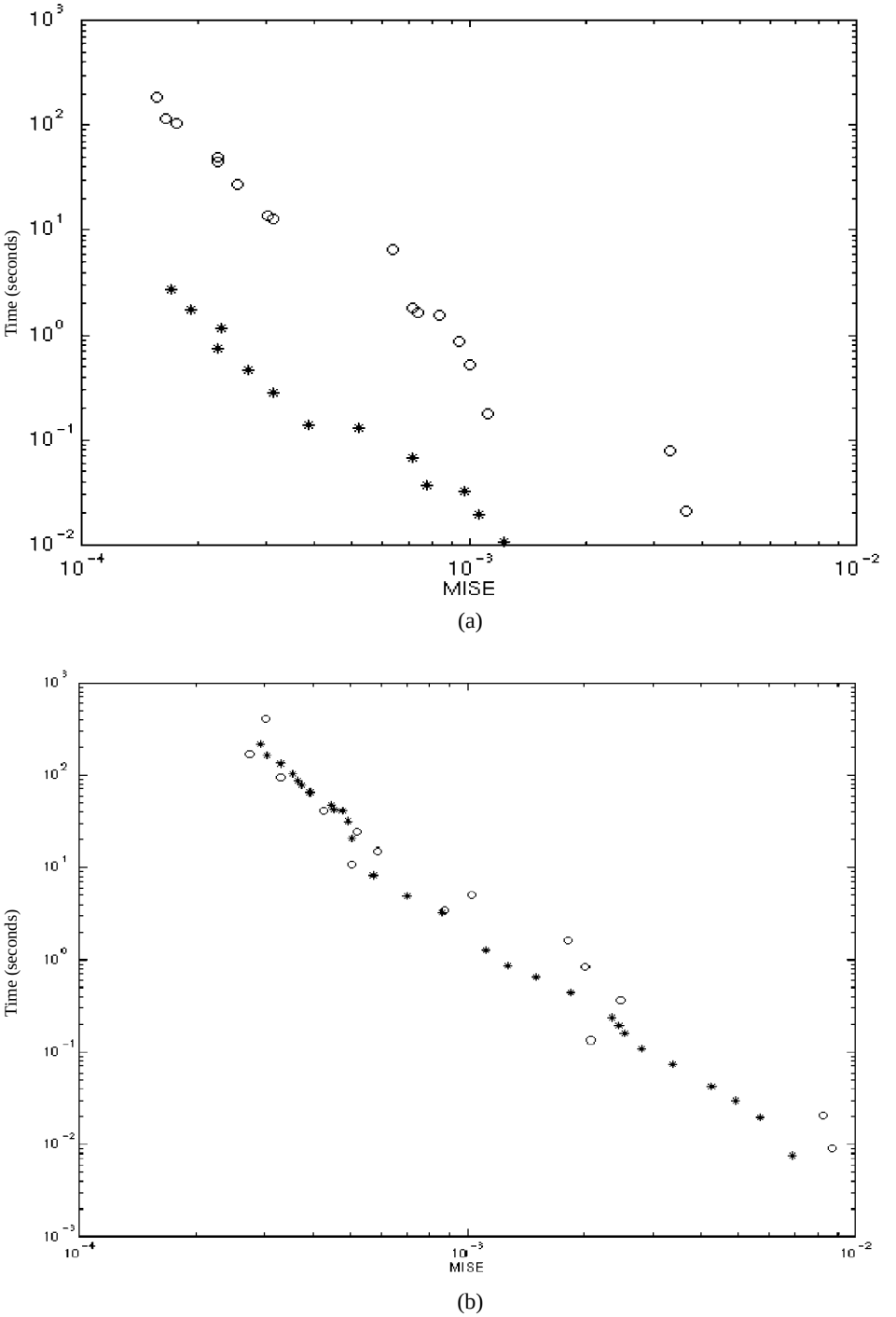
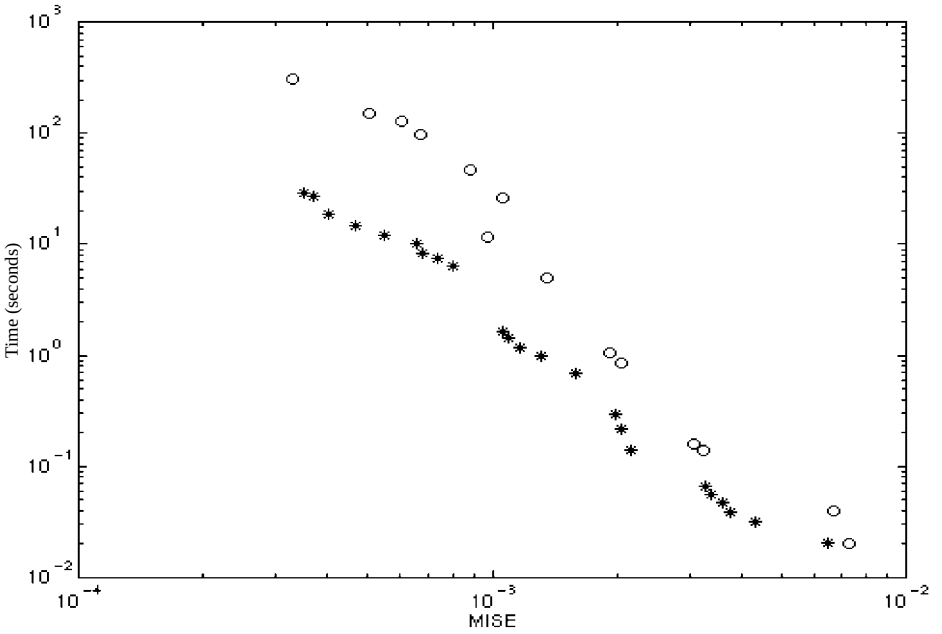


FIG. 9. *Plot of MISE versus time for the density function* $g(\phi; 2.0) = k \cosh(2.0 \sin\phi)$. *The points marked by* * *are for the cosine estimator, and the points markeds by* o *are for the kernel estimator.* (a) *Axial estimator.* (b) *Directional estimator.*



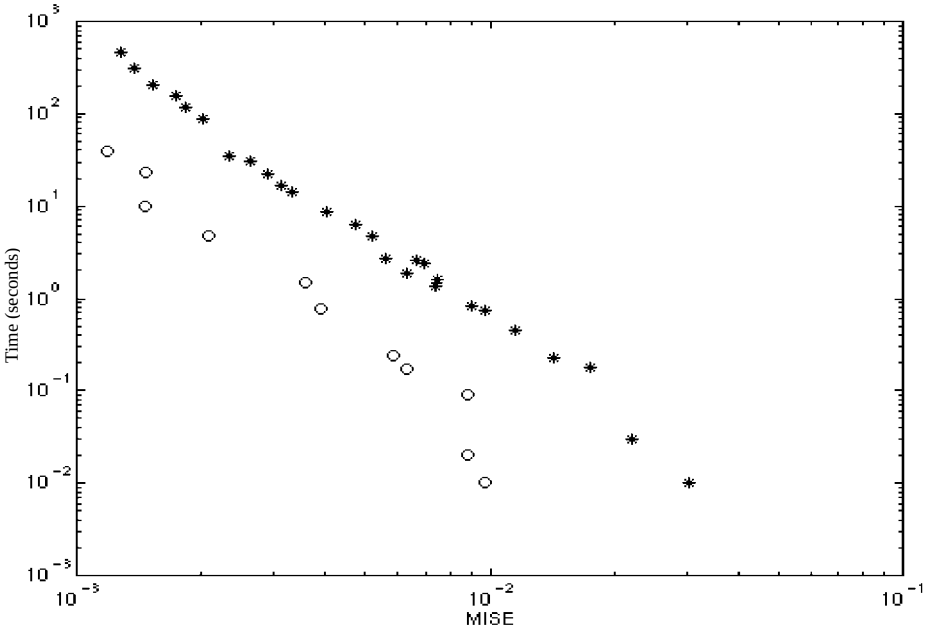
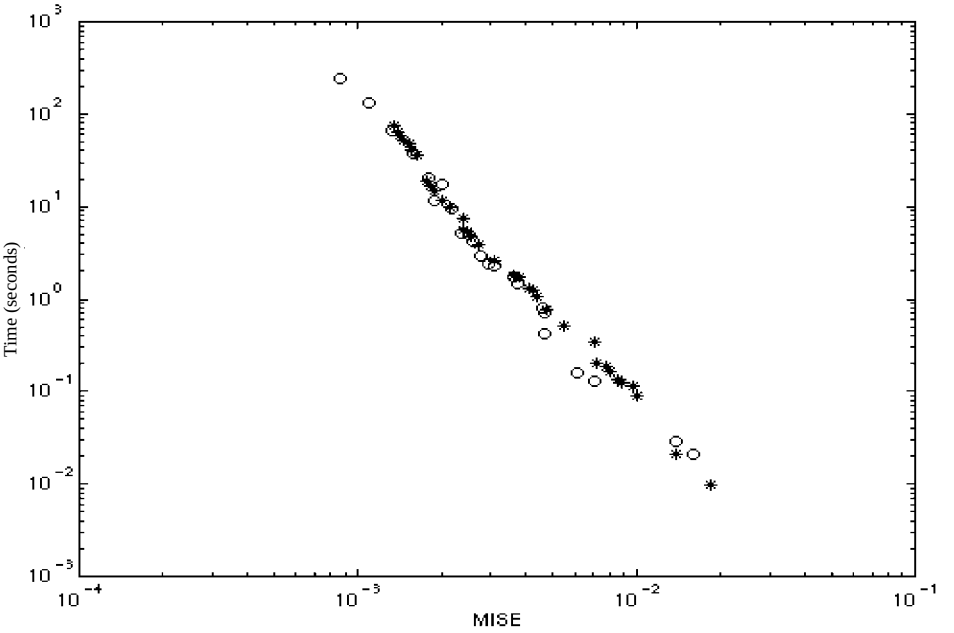
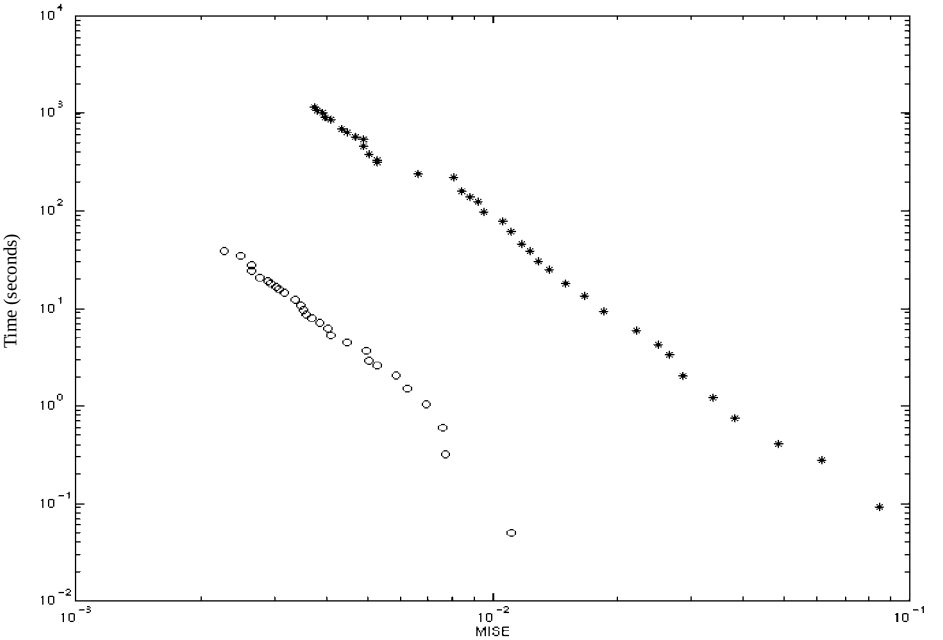


FIG. 10. *Plot of MISE versus time for the density function* $g(\phi; 4.0) = k \cosh(4.0 \sin\phi)$. *The points marked by* * *are for the cosine estimator, and the points markeds by* o *are for the kernel estimator.* (a) *Axial estimator.* (b) *Directional estimator.*



(a)



(b)

FIG. 11. *Plot of MISE versus time for the density function* $g(\phi; 8.0) = k \cosh(8.0 \sin \phi)$. *The points marked by* * *are for the cosine estimator, and the points markeds by* o *are for the kernel estimator.* (a) *Axial estimator.* (b) *Directional estimator.*

We present the results of the experiments as plots of time versus MISE for the kernel estimator versus the cosine estimator, for both axial and directional data. The kernel estimator is the one used in the empirical tests in section 4. The tests were performed on an Intel Celeron 300MHz processor with 64 MB memory. The C code was compiled with the gcc compiler at optimization level $-O3$.

We can see from Figures 7, 8, 9, 10, and 11 that when the density function is not very sharp, the cosine estimator outperforms the kernel estimator for both axial and directional data. As the density becomes sharper, the kernel method starts outperforming the cosine estimator for directional data, though the latter is still better for axial data. When the density becomes extremely sharp, the kernel method becomes better for both types of data, though for axial data the two methods are still comparable to a certain extent in terms of speed. These results follow the theoretically predicted trends and demonstrate that these two methods complement each other for different types of data.

Appendix. We thank the referees for their detailed comments and advice, especially for directing our attention to the current literature.

REFERENCES

[1] P. J. Bickel and M. Rosenblatt, *On some global measures of the deviations of density function estimates*, Ann. Stat., 1 (1973), pp. 1071–1095.

[2] C. Chaubal, Ö. Eğecioğlu, G. Leal, and A. Srinivasan, *Smoothed particle hydrodynamics techniques for the solution of kinetic theory problems. Part* 1: *Method*, J. Non-Newtonian Fluid Mech., 70 (1997), pp. 125–154.

[3] N. N. Chentsov, *Estimation of unknown probability density based on observations*, Dokl. Akad. Nauk SSSR, 147 (1962), pp. 45–48 (in Russian).

[4] Ö. Eğecioğlu and A. Srinivasan, *Efficient Nonparametric Estimation of Probability Density Functions*, Technical Report TRCS95-21, University of California at Santa Barbara, 1995.

[5] J. Fan and J. S. Marron, *Fast implementations of nonparametric curve estimators*, J. Comput. Graph. Statist., 3 (1994), pp. 35–56.

[6] N. I. Fisher, T. Lewis, and B. J. J. Embleton, *Statistical Analysis of Spherical Data*, Cambridge University Press, Cambridge, 1993.

[7] I. S. Gradshteyn and I. M. Ryzhik, *Table of Integrals, Series, and Products*, Academic Press, New York, London, 1980, pp. 374.

[8] P. Hall, G. S. Watson, and J. Cabrera, *Kernel density estimation with spherical data*, Biometrika, 74 (1987), pp. 751–762.

[9] L. Hernquist and N. Katz, *TREESPH: A unification of SPH with the hierarchical tree method*, Astrophys. J. Suppl., 70 (1989), pp. 419–446.

[10] J. Hwang, *Non-parametric multivariate density estimation: A comparative study*, IEEE Trans. Signal Process., 42 (1994), pp. 2795–2810.

[11] R. Kronmal and M. Tarter, *The estimation of probability densities and cumulatives by Fourier series methods*, Amer. Statist., (1968), pp. 925–952.

[12] J. J. Monaghan and J. C. Lattanzio *A refined particle method for astrophysical problems*, Astron. Astrophys., 149 (1985), pp. 135–143.

[13] J. J. Monaghan, *Smoothed particle hydrodynamics*, Annu. Rev. Astron. Astrophys., 30 (1992), pp. 543–574.

[14] E. A. Nadaraya, *Non-parametric Estimation of Probability Densities and Regression Curves*, Mathematics and Applications (Soviet Series)1, Kluwer Academic Publishers, Boston, 1989.

[15] E. Parzen, *On estimation of a probability density function and mode*, Ann. Math. Statist., 33 (1962), pp. 1065–1076.

[16] M. Rosenblatt, *Remarks on some non-parametric estimates of a density function*, Ann. Math. Statist., 27 (1956), pp. 832–837.

[17] S. C. Schwartz, *Estimation of probability density by an orthogonal series*, Ann. Math. Statist., 38 (1967), pp. 1261–1265.

[18] D. W. Scott, *Multivariate Density Estimation*, John Wiley, New York, 1992.

[19] B. Seifert, M. Brockmann, J. Engel, and T. Gasser, *Fast algorithms for nonparametric curve estimation*, J. Comput. Graph. Statist., 3 (1994), pp. 192–213.
[20] B. W. Silverman, *Kernel density estimation using the fast Fourier transform*, Appl. Statist., 31 (1982), pp. 93–99.
[21] N. V. Smirnov, *On the approximation of probability densities of random variables*, Scholarly Notes of Moscow State Polytechnical Institute, 16 (1951), pp. 69–96 (in Russian).
[22] G. D. Smith, *Numerical Solution of Partial Differential Equations: Finite Difference Methods*, Oxford University Press, London, 1978.
[23] B. W. Silverman, *Density Estimation for Statistics and Data Analysis*, Chapman and Hall, London, 1986.
[24] A. Szeri and L. G. Leal, *A new computational method for the solution of flow problems of microstructured fluids. Part* 2. *Inhomogeneous shear flow of a suspension*, J. Fluid Mech., 262 (1994), pp. 171–204.
[25] R. Vio, G. Fasano, M. Lazzarin, and O. Lessi, *Probability density estimation in astronomy*, Astron. Astrophys., 289 (1994), pp. 640–648.
[26] G. Walter and J. Blum, *Probability density estimation using delta sequences*, Ann. Statist., 7 (1979), pp. 328–340.
[27] M. P. Wand and M. C. Jones, *Kernel Smoothing*, Chapman and Hall, London, 1995.
[28] G. S. Watson and M. R. Leadbetter, *On the estimation of the probability density,* I, Ann. Math. Statist., 34 (1963), pp. 480–491.
[29] P. Whittle, *On the smoothing of probability density functions*, J. Roy. Statist. Soc. Ser. B, 20 (1958), pp. 334–343.

SIAM J. SCI. COMPUT.
Vol. 22, No. 1, pp. 177–197
© 2000 Society for Industrial and Applied Mathematics

A DATA-BOUNDED QUADRATIC INTERPOLANT ON TRIANGLES AND TETRAHEDRA*

MARTIN BERZINS†

Abstract. Many real-world problems are successfully modeled by partial differential equations. Many numerical solvers for these problems use triangular and tetrahedral meshes to accurately model complex geometries. Such problems often involve shocks and discontinuities, and it is important to devise interpolation methods that can accurately approximate solutions containing such features. These interpolants are required for the postprocessing of the solution, e.g., for visualization and to recover solution values at arbitrary points over the numerical domain. This paper describes a triangle-based quadratic interpolant that is "data-bounded" and so will not create any values outside of the range of the existing data points. The method is compared with the standard quadratic interpolant and extended to the case of quadratic tetrahedral elements.

Key words. interpolation, triangles, tetrahedra, quadratic elements

AMS subject classifications. 65M20, 65M15

PII. S1064827597317636

1. Introduction.

In scientific computing, the visualization of the solution to real-world problems is an essential aid to the understanding of the physical problem being modeled. Interpolation schemes that will respect the physical properties of the underlying data are thus needed, one example being to preserve positivity. Many data sets that require such interpolants result from the numerical solution of partial differential equations (PDEs). Many of the methods used to solve such PDE problems compute solutions on rectangular or hexahedral meshes. A good survey of a number of interpolants for such meshes which are appropriate for scientific visualization in that they provide values that are bounded by the data values (data bounded) and possibly preserve the shape of data values is given by Brodlie and Mashwama [6]. These interpolants are piecewise linear bilinear and trilinear and are piecewise cubic and bicubic. One such interpolant is a bounded bicubic interpolant on a rectangular mesh [6].

In two and three spatial dimensions a number of general purpose PDE solvers employ triangular and tetrahedral elements in conjunction with triangular and tetrahedral mesh generators to solve problems defined on complex domains. The best example of this being the finite element method [12]. Alternatively, a number of authors use a cell-centered or cell-vertex finite volume spatial discretization schemes to solve convection-dominated PDEs; see [5], [7], [11] for details. An important feature of these problems is that initial smooth conditions may develop into steep gradients or even shocks and discontinuities.

Standard quadratic interpolation techniques [8], [12] can be used to fit six shape functions over each triangle to give an approximation to the surface. Section 2 below and [10] both show that this method can easily produce interpolated values outside the physical range of the data values. Similar problems arise with the standard quadratic interpolant for tetrahedra [12].

*Received by the editors October 23, 1997; accepted for publication (in revised form) April 12, 1999; published electronically June 13, 2000.
http://www.siam.org/journals/sisc/22-1/31763.html
†School of Computer Studies, The University of Leeds, Leeds LS2 9JT, U.K. (martin@csc.leeds.ac.uk).

There have been many methods which have attempted to overcome some of the problems associated with such interpolants. Some of the earliest work is that of Barnhill and co-authors [2], [3]. This work will be described in outline form and used as a starting point for the new triangular interpolant developed here. Two other important interpolation methods in this area are those of Abgrall [1] and Barth [4]. Both these schemes have the common approach of using adaptive multitriangle stencils to achieve high-order accuracy for problems which may have shocks and discontinuities. The problem is that the number of possible combinations grows rapidly, the stencil used is potentially large, and the points considered may be some distance from the original node. Abgrall's good results provide a more than adequate justification of the scheme however.

A recent paper by Pratt and Berzins [10] showed that by adopting a relatively simple approach which involved sacrificing accuracy at the midpoints of edges it was possible to achieve positivity for problems with steep gradients. The aim here is to overcome this deficiency and to consider a simpler alternative to Abgrall's scheme. This will be done by decomposing a standard quadratic interpolant into a combination of four one-dimensional quadratics. Replacing each of these one-dimensional interpolants by ones which preserve data-boundedness, possibly by replacing one quadratic by two piecewise quadratics, enables a new data-bounded interpolant to be devised. This interpolant is bounded by the minimum and maximum data values defining it and may therefore be utilized to preserve positivity. It can also be used for the visualization of the solution and by the numerical solver to recover values over the numerical domain. A novel feature of the new scheme is that the method is local to each triangle or tetrahedral element. The price that is paid for this is that when the interpolant is modified to preserve data-boundedness the resulting piecewise polynomial is only C^0 continuous rather than C^1 continuous.

The extension of the approach to tetrahedra proves to be straightforward by using a combination of positivity preserving one-dimensional and triangular interpolants.

2. The standard quadratic triangular interpolation scheme. A two-dimensional quadratic triangular interpolant of an unknown function $u(x, y)$ needs six data points: these points are usually at the vertices of the triangle and the midpoints of the sides [12]. These can be mapped to area coordinates (L_1, L_2, L_3). Six shape functions can be fitted to these points such that each is unified at one point and zero at the others. These shape functions are shown in (1):

$$\begin{aligned} \phi_1 &= (2L_1 - 1)L_1, & \phi_2 &= (2L_2 - 1)L_2, \\ \phi_3 &= (2L_3 - 1)L_3, & \phi_4 &= 4L_2L_1, \\ \phi_5 &= 4L_3L_2, & \phi_6 &= 4L_1L_3. \end{aligned} \tag{1}$$

The area coordinates of points, (L_1, L_2, L_3), 1 to 6 being $(1, 0, 0)$, $(0, 1, 0)$, $(0, 0, 1)$, $(\frac{1}{2}, \frac{1}{2}, 0)$, $(0, \frac{1}{2}, \frac{1}{2})$, and $(\frac{1}{2}, 0, \frac{1}{2})$, respectively. The interpolated value U_I is then defined by

$$U_I(L_1, L_2, L_3) = \sum_{i=1}^{6} \phi_i(L_1, L_2, L_3)U_i. \tag{2}$$

The position of the points U_i is shown in Figure 1. $U_i, i = 1, 2, 3$ being the points at which the corresponding $L_i = 1$.

The problem with the standard interpolation formula is that the shape functions associated with the three vertex values are negative over large parts of the triangle.

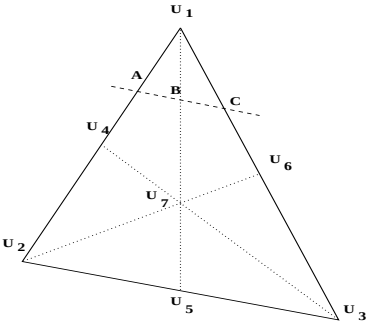


FIG. 1. *Example triangle.*

Thus it is possible that new and unphysical extrema may be introduced [10]. Consider, e.g., the centroid of the triangle marked as U_7 in Figure 1. In the case when standard quadratic interpolation based on the values $U_i, i = 1, 6$ is used, then

$$U_7 = U\left(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right) = \frac{4}{9}(U_4 + U_5 + U_6) - \frac{1}{9}(U_1 + U_2 + U_3). \tag{3}$$

Hence if all the values $U_i, i = 1, 6$ are positive but the values U_1, U_2, and U_3 are in some sense large compared to U_4, U_5, and U_6, then U_7 may be negative. More generally, possible extrema of the standard quadratic interpolant polynomial U_I defined by (2) lie at the points at which (after replacing L_3 with $1 - L_1 - L_2$) $\frac{\partial U}{\partial L_1} = 0$ and $\frac{\partial U}{\partial L_2} = 0$. Differentiating U_I and collecting terms together gives a pair of equations for L_1, L_2, the critical points:

$$4L_1(U_1 + U_3 - 2U_6) + 4L_2(U_3 + U_4 - U_5 - U_6) = (U_1 + 3U_3 - 4U_6),$$
$$4L_1(U_3 + U_4 - U_5 - U_6) + 4L_2(U_3 + U_2 - 2U_5) = (U_2 + 3U_3 - 4U_5).$$

The standard multivariable calculus test for determining extrema is that $D < 0$, where $D = (\frac{\partial^2 U_I}{\partial x \partial y} - \frac{\partial^2 U_I}{\partial x^2}\frac{\partial^2 U_I}{\partial y^2})$. The case when $D > 0$ defines saddle points, or if $D = 0$ the test fails, and there may be no maximum or minimum or a line of critical points. A lengthy but straightforward derivation shows that D has the same sign as $-Det$, where Det is the determinant of the preceding pair of equations.

Although this test provides useful help in understanding why the standard interpolant may not be data-bounded, it does not directly help to construct a data-bounded interpolant. It is this problem, that of constructing a data-bounded quadratic interpolant, that will be addressed in sections 4 and 5 below. This will be achieved by showing that the original polynomial may be interpreted as a combination of one-dimensional quadratics. A geometrical outline of the approach is indicated by Figure 1 which indicates how a point on the line ABC is defined in terms of quadratics using U_2, U_4, and U_1 to define a value at A, U_5, U_7, and U_1 to define a value at B, and using U_3, U_6, and U_1 to define a value at C. A further quadratic interpolation using ABC then defines points on that line. The problems are thus reduced to that of finding data-bounded quadratic polynomials in one space dimension and of finding a suitable centroid value.

3. Two data-bounded piecewise quadratic polynomials. In this section two data-bounded one-dimensional piecewise quadratic polynomials are devised. These polynomials will play an important role in the triangular interpolant in section 4 and

in the tetrahedral interpolant in section 8. The approach taken to ensure data boundedness will be, when necessary, to decompose the original quadratic polynomial into two piecewise polynomial quadratics.

3.1. Polynomial P_1. Consider the case of the standard one-dimensional quadratic interpolant for the function $u(x)$ defined on the interval $[0,h]$ mapped onto $[0,1]$ with data points at $0, \frac{1}{2}$, and 1 given by $U_0, U_{\frac{1}{2}}$, and U_1. Let this polynomial be defined by

$$q_1(U_0, U_{\frac{1}{2}}, U_1, L_1) = U_0\hat{\phi}_1 + U_{\frac{1}{2}}\hat{\phi}_2 + U_1\hat{\phi}_3, \tag{4}$$

where

$$\hat{\phi}_1 = (1-L_1)(1-2L_1), \quad \hat{\phi}_2 = 4L_1(1-L_1), \quad \hat{\phi}_3 = L_1(2L_1-1),$$

and $L_1 + L_2 = 1$. Differentiating with respect to L_1 gives

$$\frac{dq_1}{dL_1} = 0 \text{ at } L_1 = \frac{1}{2} + r, \quad \text{where } \; r \; = \frac{U_1 - U_0}{4(2U_{\frac{1}{2}} - U_1 - U_0)} \tag{5}$$

and differentiating again with respect to L_1 gives

$$\frac{d^2q_1}{dL_1^2} = 4(-2U_{\frac{1}{2}} + U_1 + U_0), \tag{6}$$

which is also a second-order difference approximation to $h^2\frac{d^2u}{dx^2}(\frac{1}{2})$. This result shows that the polynomial may have extrema at nonnodal points. If the second derivative is zero then as q is linear and hence takes its extrema at the ends of the interval.

In order to get a data-bounded interpolant for which any extrema lie at data points, the key observation in modifying the polynomial is that on the interval $[0, \frac{1}{2}]$ a completely different value of U_1 may be used from the original. Similarly, on the interval $[\frac{1}{2}, 1]$ a completely different value of U_0 from the original may be used without destroying C^0 but not C^1 continuity of the solution. Thus the original polynomial is replaced by two piecewise quadratic polynomials.

Let these new values of U_0 and U_1 be denoted by U_0^* and U_1^*, respectively. The new piecewise polynomial is then defined by

$$p_1(U_0, U_{\frac{1}{2}}, U_1, L_1) = U_0\hat{\phi}_1 + U_{\frac{1}{2}}\hat{\phi}_2 + U_1^*\hat{\phi}_3, \quad 0 \le L_1 \le \frac{1}{2}, \tag{7}$$

$$= U_0^*\hat{\phi}_1 + U_{\frac{1}{2}}\hat{\phi}_2 + U_1\hat{\phi}_3, \quad \frac{1}{2} < L_1 \le 1. \tag{8}$$

These new values of U_0^* and U_1^* will be defined by moving extrema to the closest nodal point, denoted by L_{ext}^{new}, by using (5). In the case when $r \le -\frac{1}{2}$ or $r \ge \frac{1}{2}$, then $U_0^* = U_0$ and $U_1^* = U_1$ and the original polynomial is unchanged

$$L_{ext}^{new} = 0, -\frac{1}{2} < r \le -\frac{1}{4}, U_1^* = 4U_{\frac{1}{2}} - 3U_0, \tag{9}$$

$$L_{ext}^{new} = \frac{1}{2}, \quad -\frac{1}{4} < r \le 0, \;\; U_1^* = U_0, \tag{10}$$

$$L_{ext}^{new} = \frac{1}{2}, \quad 0 < r \le \frac{1}{4}, \quad U_0^* = U_1, \tag{11}$$

$$L_{ext}^{new} = 1, \quad \frac{1}{4} < r \le \frac{1}{2}, \quad U_0^* = 4U_{\frac{1}{2}} - 3U_1. \tag{12}$$

In all the above four cases $U_1 - U_1^*$ and $U_0 - U_0^*$ may be rewritten in a more convenient form as

$$L_{ext}^{new} = 0, \quad U_1^* - U_1 = 4U_{\frac{1}{2}} - 3U_0 - U_1 = \frac{(1+2r)}{2} 4\ (2U_{\frac{1}{2}} - U_0 - U_1), \tag{13}$$

$$L_{ext}^{new} = \frac{1}{2}, \quad U_1^* - U_1 = \quad U_0 - U_1 \quad = \ - \ r \quad 4\ (2U_{\frac{1}{2}} - U_0 - U_1), \tag{14}$$

$$L_{ext}^{half} = \frac{1}{2}, \quad U_0^* - U_0 = \quad U_1 - U_0 \quad = \quad r \quad 4\ (2U_{\frac{1}{2}} - U_0 - U_1), \tag{15}$$

$$L_{ext}^{new} = 1, \quad U_0^* - U_0 = 4U_{\frac{1}{2}} - 3U_1 - U_0 = \frac{(1-2r)}{2}\ 4\ (2U_{\frac{1}{2}} - U_0 - U_1). \tag{16}$$

From the last four equations and (6)

$$U_1 - U_1^* = \alpha(r)\ \frac{d^2 q_1}{dL_1^2}, \quad \text{where} \quad -1/2 \le r \le 0, \tag{17}$$

$$U_0 - U_0^* = \alpha(r)\ \frac{d^2 q_1}{dL_1^2}, \quad \text{where} \quad 0 \le r \le 1/2, \tag{18}$$

and where $0 \le \alpha(r) \le 1/4$ is the piecewise linear function defined as in (13)–(16). Thus the algorithm adds a nonlinear multiple of the second derivative in order to get a data-bounded interpolant. The extra term introduced by this approach is given by

$$q_1(\ldots, L_1) - p_1(\ldots, L_1) = (U_1^* - U_1)\hat{\phi}_3, \quad 0 \le L_1 \le \frac{1}{2}, \tag{19}$$

$$= (U_0^* - U_0)\hat{\phi}_1, \quad \frac{1}{2} < L_1 \le 1. \tag{20}$$

As a consequence of this, the error as defined on $[\frac{1}{2}, 1]$ is no longer defined by the standard form on $[h/2, h]$ given by

$$u(x) - q_1(U_0, U_{\frac{1}{2}}, U_1, L_1) = x(x - h/2)(x - h)\ \frac{1}{6}\frac{d^3 u}{dx^3}(\xi_1), \quad \xi_1 \epsilon [0, h], \tag{21}$$

where $u(x)$ is defined at the start of this section, but by

$$u(x) - p_1(U_0, U_{\frac{1}{2}}, U_1, L_1) = x(x - h/2)(x - h)\ \frac{1}{6}\frac{d^3 u}{dx^3}(\xi_1)$$

$$- \frac{2}{h^2}(U_0^* - U_0)(x - h/2)(x - h). \tag{22}$$

Substituting from (15) and (16) for $U_0^* - U_0$ gives

$$u(x) - p_1(U_0, U_{\frac{1}{2}}, U_1, L_1) =$$

$$(x - h/2)(x - h)\ \left[4\ \alpha(r)\ \frac{4(-2U_{\frac{1}{2}} + U_0 + U_1)}{2\ h^2}\ +\ \frac{x}{6}\frac{d^3 u}{dx^3}(\xi_1)\right]. \tag{23}$$

Thus in the same way as limiter schemes in the solution of hyperbolic PDEs vary the order of the method to preserve positivity (see [5]) the function $\alpha(r)$ varies the order between second order $|\alpha(r)| = 1/4$ and third order $\alpha = 0$ to preserve data-boundedness. The case $\alpha(r) = 1/4$ occurs only if $U_0 = U_{\frac{1}{2}}$ or $U_1 = U_{\frac{1}{2}}$ and then the use of a linear polynomial is unavoidable.

In contrast, when linear interpolation is used, e.g., on the subinterval $[h/2, h]$, to substitute for the polynomial q_1 the approximation polynomial is

$$l_1(U_{\frac{1}{2}}, U_1, L_1) = U_{\frac{1}{2}} 2(1 - L_1) + U_1(2L_1 - 1) \tag{24}$$

and the modified form of the error may, for comparison purposes, be written as $u(x) - l_1(.) = u(x) - q_1(.) - (l_1(.) - q_1(.))$. After some manipulation this may be written as

$$u(x) - l_1(U_{\frac{1}{2}}, U_1, L_1) = \\ (x - h/2)(x - h) \left[\frac{4(-2U_{\frac{1}{2}} + U_0 + U_1)}{2\ h^2} + \frac{x}{6} \frac{d^3u}{dx^3}(\xi_1) \right]. \tag{25}$$

A comparison between (23) and (25) shows that when the term $\frac{d^3u}{dx^3}(\xi_1)$ vanishes then the ratio of the two errors is $4\alpha(r) : 1$. This is not generally the case. In the examples below, for example, the ratios of the errors in the two cases are obtained by comparing the bracketed terms ([ ]) in (23) and (25). Furthermore, if the original polynomial $q_1(.)$ violates the positivity of the data values, then in order for the modified polynomial to be data bounded there must be a significant cancellation in the two terms in [ ] in (23) and (25). This is also shown in the examples below.

3.1.1. Numerical examples. In this section the cases $U_0 = 100, U_{\frac{1}{2}} = 0.3$, and $U_1 = 0.1$, and $U_0 = 1.0, U_{\frac{1}{2}} = 0.3$, and $U_1 = 0.1$ are considered as data points for the function $u(x)$ defined by

$$u(x) = U_1\ e^{a(1-x)^b}, \tag{26}$$

where

$$a = \log(U_0/U_1), \quad c = \log(U_{\frac{1}{2}}/U_1), \quad \text{and} \quad b = (\log(a) - \log(c))/\log(2).$$

In the first case $U_0 = 100$ and the new and original polynomials are identical in $[0, 0.5]$. Table 1 shows the values of the interpolants at points in $(0.5, 1.0)$ and Table 2 shows the errors in the interpolants. In particular, the original quadratic polynomial has a minimum of about -12, whereas the new polynomial remains positive. In this case the new value of U_0^* is 0.9, $U_0^* - U_0 = -99.1$ $\alpha(r) = 0.246$, and the change to the original polynomial has to be substantial to achieve data-boundedness. Also shown are the results from the linear polynomial l_1 defined by (24).

Now consider the case when the value of U_0 is changed to $U_0 = 1, U_{\frac{1}{2}} = 0.3$, and $U_1 = 0.1$. The new and original polynomials are identical in $[0, 0.5]$. Table 1 shows the differences at points in $(0.5, 1.0)$. In particular, the original quadratic polynomial q_1 has a minimum of about 0.0975, whereas the new polynomial p_1 remains within the range of the data. In this case the new value of U_0^* is 0.9, $U_0^* - U_0 = -0.1$, and $\alpha(r) = 0.05$. Thus the change to the original polynomial is quite small to ensure that the data values stay within the range of the data. It should be noted that the new polynomial is the same for both values of U_0 as the value of r in both cases defines the same value of U_0^*. Also shown are the results from the linear polynomial l_1 defined by (24).

TABLE 1
Results for the standard, modified quadratic, and linear interpolants.

U_0	L_1	0.5	0.55	0.60	0.65	0.70	0.75	0.80	0.85	0.90	0.95
100.0	$u(x)$	0.3	0.230	0.184	0.153	0.133	0.119	0.110	0.105	0.102	0.100
100.0	p_1	0.3	0.262	0.228	0.198	0.172	0.150	0.132	0.118	0.108	0.102
100.0	l_1	0.3	0.280	0.260	0.240	0.220	0.200	0.180	0.160	0.140	0.120
100.0	q_1	0.3	-4.197	-7.700	-10.21	-11.72	-12.24	-11.76	-10.29	-7.820	-4.357
1.0	$u(x)$	0.3	0.267	0.238	0.212	0.189	0.169	0.151	0.136	0.122	0.110
1.0	p_1	0.3	0.262	0.228	0.198	0.172	0.150	0.132	0.118	0.108	0.102
1.0	l_1	0.3	0.280	0.260	0.240	0.220	0.200	0.180	0.160	0.140	0.120
1.0	q_1	0.3	0.257	0.220	0.178	0.160	0.137	0.120	0.107	0.100	0.098

TABLE 2
Errors in the standard, modified quadratic, and linear interpolants.

U_0	L_1	0.55	0.60	0.65	0.70	0.75	0.80	0.85	0.90	0.95
100.0	$p_1 - u(x)$	0.032	0.044	0.045	0.039	0.031	0.022	0.013	0.006	0.002
100.0	$l_l - u(x)$	0.050	0.076	0.087	0.087	0.081	0.070	0.055	0.038	0.020
100.0	$q_1 - u(x)$	-4.42	-7.880	-10.36	-11.85	-12.35	-11.87	-10.39	-7.920	-4.458
1.0	$p_1 - u(x)$	-0.005	-0.010	-0.014	-0.017	-0.019	-0.019	-0.018	-0.014	-0.008
1.0	$l_1 - u(x)$	0.013	0.022	0.028	0.031	0.031	0.029	0.024	0.018	0.010
1.0	$q_1 - u(x)$	-0.009	-0.018	-0.024	-0.029	-0.031	-0.031	-0.028	-0.022	-0.012

3.2. Polynomial P_2. Consider now the case of the one-dimensional quadratic interpolant to the function $u(x)$ on $[0,1]$ with data points at $0, \frac{1}{3}$, and 1 given by $U_0, U_{\frac{1}{3}}$, and U_1 defined by

$$q_2(U_0, U_{\frac{1}{3}}, U_1, L_1) = U_0\bar{\phi}_1 + U_{\frac{1}{3}}\bar{\phi}_2 + U_1\bar{\phi}_3, \tag{27}$$

where

$$\bar{\phi}_1 = (1-L_1)(1-3L_1), \quad \bar{\phi}_2 = 4.5L_1(1-L_1), \quad \bar{\phi}_3 = \frac{1}{2}L_1(3L_1-1),$$

and $L_1 + L_2 = 1$. Differentiating with respect to L_1 gives

$$\frac{dq_2}{dL_1} = 0 \text{ at } L_1 = \frac{1}{2} + s, \quad \text{where } s = \frac{U_1 - U_0}{3(3U_{\frac{1}{3}} - U_1 - 2U_0)}. \tag{28}$$

Differentiating again with respect to L_1 gives

$$\frac{d^2q_2}{dL_1^2} = 3(-3U_{\frac{1}{3}} + U_1 + 2U_0). \tag{29}$$

As above, this polynomial may have extrema at nonnodal points which may be removed by modifying the polynomial on the interval $[0, \frac{1}{3}]$ by using a completely different value of U_1 from the original or on the interval $[\frac{1}{3}, 1]$ by using a completely different value of U_0 from the original. This approach preserves C^0 but not C^1 continuity of the solution. The new polynomial is defined by

$$p_2(U_0, U_{\frac{1}{3}}, U_1, L_1) = U_0\bar{\phi}_1 + U_{\frac{1}{3}}\bar{\phi}_2 + U_1^*\bar{\phi}_3, \quad 0 \le L_1 \le \frac{1}{3} \tag{30}$$

$$= U_0^*\bar{\phi}_1 + U_{\frac{1}{3}}\bar{\phi}_2 + U_1\bar{\phi}_3, \quad \frac{1}{3} < L_1 \le 1. \tag{31}$$

Define the values of U_0^* and U_1^* by moving extrema to the closest nodal point, denoted by L_{ext}^{new}. In the case when $s \leq -\frac{1}{2}$ or $s \geq \frac{1}{2}$, then $U_0^* = U_0$ and $U_1^* = U_1$ and the original polynomial is unchanged. Define the values of U_0^* and U_1^1 by again moving extrema to the closest nodal point:

$$L_{ext}^{new} = 0, -\frac{1}{2} < s \leq -\frac{1}{3},\ U_1^* = 9U_{\frac{1}{3}} - 8U_0, \tag{32}$$

$$L_{ext}^{new} = \frac{1}{3}, -\frac{1}{3} < s \leq -\frac{1}{6},\ U_1^* = 4U_0 - 3U_{\frac{1}{3}}, \tag{33}$$

$$L_{ext}^{new} = \frac{1}{3},\ -\frac{1}{6} < s \leq \frac{1}{6},\ \ U_0^* = (U_1 + 3U_{\frac{1}{3}})/4, \tag{34}$$

$$L_{ext}^{new} = 1,\ \ \frac{1}{6} < s \leq \frac{1}{2},\quad U_0^* = (9U_{\frac{1}{3}} - 5U_1)/4. \tag{35}$$

The extra error introduced by this approach is given by

$$q_2(\ldots, L_1) - p_2(\ldots, L_1) = (U_1 - U_1^*)\hat{\phi}_3,\ \ 0 \leq L_1 \leq \frac{1}{3}, \tag{36}$$

$$= (U_0 - U_0^*)\hat{\phi}_1,\ \ \frac{1}{3} < L_1 \leq 1, \tag{37}$$

where

$$\begin{aligned}
L_{ext}^{new} = 0,\ U_1^* - U_1 &= \ \ 9U_{\frac{1}{3}} - 8U_0 - U_1 \ \ = (1+2s) \quad 3(3U_{\frac{1}{3}} - U_1 - 2U_0),\\
L_{ext}^{new} = \frac{1}{3},\ U_1^* - U_1 &= \ \ 4U_0 - 3U_{\frac{1}{3}} - U_1 \ \ = -\left(\frac{1}{3} + 2s\right)\ 3(3U_{\frac{1}{3}} - U_1 - 2U_0),\\
L_{ext}^{new} = \frac{1}{3},\ U_0^* - U_0 &= (U_1 + 3U_{\frac{1}{3}})/4 - U_0 = \left(\frac{1}{12} + \frac{s}{2}\right)\ \ 3(3U_{\frac{1}{3}} - U_1 - 2U_0),\\
L_{ext}^{new} = 1,\ U_0^* - U_0 &= (9U_{\frac{1}{3}} - 5U_1)/4 - U_0 = \left(\frac{1}{4} - \frac{s}{2}\right) \quad 3(3U_{\frac{1}{3}} - U_1 - 2U_0).
\end{aligned}$$

From the last four equations and (5) $U_1 - U_1^* = \beta(s)\ \frac{d^2 q_2}{dL_1^2}$, where $0 \leq \beta(s) \leq 1/3$ and similarly for $U_0 - U_0^*$. Hence similar arguments as put forward by (21)–(25) in section 3.1 apply.

4. A data-bounded two-dimensional quadratic interpolant. In extending the ideas behind the one-dimensional schemes above to two dimensions there are a number of possible ways to proceed. A direct two-dimensional analogy for the previous section would be to notice that the value of U_1 could be changed to U_1^* for values of $L_1 < \frac{1}{2}$ and similarly for U_2 and U_3. Although this approach can be made to work for a single triangle, there is a problem in enforcing continuity of the solution along exterior edges in a mesh of triangles. For a positive interpolant in a mesh of triangles it is thus important for the scheme to treat edges independently.

The starting point for the two-dimensional quadratic scheme is the observation that the ordinary quadratic interpolant may be written as a combination of four one-dimensional quadratic interpolants; two along exterior edges, one through the centroid, and the final one across the other three. There are three such interpolants which will be denoted by $U_{I,j}, j = 1, 2, 3$; the subscript I being used to avoid confusion

with data points. For example, referring to Figure 1, let

$$\begin{aligned} U_A &= q_1(U_2, U_4, U_1, L_1), \\ U_B &= q_2(U_5, U_7, U_1, L_1), \\ U_C &= q_1(U_3, U_6, U_1, L_1), \\ U_{I,1} &= q_1\left(U_A, U_B, U_C, \frac{L_3}{L_2 + L_3}\right), \end{aligned} \tag{38}$$

where $L_3/(L_2+L_3)$ represents the position along the line ABC in that $L_3 = 0$ at A and $L_2 = 0$ at C. As each polynomial is exact for a quadratic this interpolant will reproduce a quadratic function. This approach has some similarities with that of Barnhill and co-authors [2], [3] except that they use three combinations of two quadratics along edges only with linear interpolation between them and subtracted a multiple of linear interpolation using the values U_1, U_2, and U_3. The main differences here are the use of the centroid value U_7 and the use throughout of quadratic polynomials. The two other ways of writing the interpolant are

$$\begin{aligned} U_D &= q_1(U_2, U_5, U_3, L_3), \\ U_E &= q_2(U_4, U_7, U_3, L_3), \\ U_F &= q_1(U_1, U_6, U_3, L_3), \\ U_{I,2} &= q_1\left(U_D, U_E, U_F, \frac{L_1}{L_2 + L_1}\right); \end{aligned} \tag{39}$$

$$\begin{aligned} U_G &= q_1(U_1, U_4, U_2, L_2), \\ U_H &= q_2(U_6, U_7, U_2, L_2), \\ U_K &= q_1(U_3, U_5, U_2, L_2), \\ U_{I,3} &= q_1\left(U_G, U_H, U_K, \frac{L_3}{L_1 + L_3}\right). \end{aligned} \tag{40}$$

In the case of the standard quadratic interpolant the centroid value, U_7 is computed using (3). This value, as has already been shown in section 2, will not preserve data-boundedness.

Using the notation of section 2 it is now straightforward to describe the bounded positive quadratic interpolant. Let $\hat{U}_A$ be the value corresponding to U_A above but replacing the polynomial q_1 with p_1 and letting the values $\hat{U}_B, \hat{U}_C, \hat{U}_D, \hat{U}_E, \hat{U}_F, \hat{U}_G, \hat{U}_H$, $\hat{U}_K$ be similarly defined using the data-bounded polynomials p_1 and p_2, and using a centroid value U_7 that is itself bounded by the data values $U_j, j = 1, \ldots, 6$. The method used to compute this centroid value will be given in the next section.

Define the polynomials $\hat{U}_{I,j}, j = 1, 3$ by

$$\hat{U}_{I,1} = p_1\left(\hat{U}_A, \hat{U}_B, \hat{U}_C, \frac{L_3}{L_2 + L_3}\right), \tag{41}$$

$$\hat{U}_{I,2} = p_1\left(\hat{U}_D, \hat{U}_E, \hat{U}_F, \frac{L_1}{L_2 + L_1}\right), \tag{42}$$

$$\hat{U}_{I,3} = p_1\left(\hat{U}_G, \hat{U}_H, \hat{U}_K, \frac{L_3}{L_1 + L_3}\right), \tag{43}$$

each of these polynomials being data-bounded and piecewise quadratic. The effect of using the different combinations of bounded quadratics means that the polynomials

are no longer identical and may be directionally biased. For these reasons the bounded positive interpolant used is $\hat{U}_I$, where

$$\hat{U}_I(L_1, L_2, L_3) = \frac{1}{3}(\hat{U}_{I,1} + \hat{U}_{I,2} + \hat{U}_{I,3}). \tag{44}$$

5. Choosing the centroid value, U_7. In the case of the cell-centered finite volume schemes [5], [7] values at the centroids of triangles are the primary ones generated by the method. In this case interpolation techniques are used to compute the values at the midpoints of edges [5] and at the nodes [10]. In the case of triangular quadratic finite element based schemes [12], however, the centroid values are not available and must be computed.

The primary requirement is that the centroid value is itself data-bounded and that any error introduced is comparable to other errors already present. In the case when the standard quadratic value U_7, defined by (3), is data-bounded, then this value is used unchanged.

Let $U_{\max}$ and $U_{\min}$ be the maximum and minimum of the values $U_1, \ldots, U_6$. In the case when U_7 lies outside of the range of these values, then U_7 is set to either $U_{\max}$ or $U_{\min}$ using the following procedure. For convenience, consider the case when $U_7 > U_{\max}$, the other case follows without difficulty. Let the (data-bounded) linear interpolant value at the centroid be denoted by U_7^L and defined by

$$U_7^L = \frac{1}{3}(U_1 + U_2 + U_3); \tag{45}$$

then from (3)

$$U_7 = U_7^L - \frac{2}{9}\left[(U_1 - 2U_4 + U_2) + (U_2 - 2U_5 + U_3) + (U_3 - 2U_6 + U_1)\right]. \tag{46}$$

Let h_1, h_2, and h_3 be the lengths of the three edges connecting U_1 and U_2, U_2, and U_2, and U_3 and U_1, respectively; then

$$U_1 - 2U_4 + U_2 \quad = \quad h_1^2 \frac{\partial^2 U_I}{\partial z_1^2}, \tag{47}$$

where z_i is the local coordinate along the ith edge with length h_i. A similar interpretation of the other terms in (46) gives

$$U_7 - U_7^L = \frac{2}{9}\left[h_1^2 \frac{\partial^2 U_I}{\partial z_1^2} + h_2^2 \frac{\partial^2 U_I}{\partial z_2^2} + h_3^2 \frac{\partial^2 U_I}{\partial z_3^2}\right]. \tag{48}$$

As the linear value U_7^L is data-bounded and the quadratic one is not, it follows that we can find a constant $0 \leq \gamma \leq 1$ such that

$$\gamma = \frac{U_{\max} - U_7^L}{U_7 - U_7^L} \tag{49}$$

and hence in replacing U_7 by $U_{\max}$ an extra source of error is introduced, which will be denoted by e_7, where $e_7 = U_7 - U_{\max}$, and note that

$$e_7 = (1 - \gamma)\ \frac{2}{9}\ \left[h_1^2 \frac{\partial^2 U_I}{\partial z_1^2} + h_2^2 \frac{\partial^2 U_I}{\partial z_2^2} + h_3^2 \frac{\partial^2 U_I}{\partial z_3^2}\right]. \tag{50}$$

6. Error and continuity analysis. The interpolation error of the standard quadratic triangular interpolant is given by Johnson [9], for example. In order to estimate the value of the extra error incurred by using the one-dimensional data-bounded polynomials consider the case of the interpolant defined by (38). The values $\hat{U}_A, \hat{U}_B, \hat{U}_C$ each have an error of the type considered in sections 3.1 and 3.2. Let these errors be denoted by $e\hat{U}_A, e\hat{U}_B$, and $e\hat{U}_C$, respectively. In addition there is a possible extra error due to the use of the approximate centroid value. From (27) this can be written as $e_7\bar{\phi}_2(L_1)$. Hence the additional error due to preserving data-boundedness in $\hat{U}_{I,1}$, which is denoted by $e\hat{U}_{I,1}$, is given by

$$\begin{aligned} e\hat{U}_{I,1} &= q_1\left(U_A, U_B, U_C, \frac{L_3}{L_2+L_3}\right) \\ &\quad - p_1\left(U_A + e\hat{U}_A, U_B + e\hat{U}_B + e_7\bar{\phi}_2(L_1), U_C + e\hat{U}_C, \frac{L_3}{L_2+L_3}\right). \end{aligned} \tag{51}$$

Adding and subtracting the term $q_1(U_A+e\hat{U}_A, U_B+e\hat{U}_B+e_7\bar{\phi}_2(L_1), U_C+e\hat{U}_C, \frac{L_3}{L_2+L_3})$ and simplifying using the results in section 3.1 gives

$$\begin{aligned} e\hat{U}_{I,1} &= -q_1\left(e\hat{U}_A, e\hat{U}_B + e_7\bar{\phi}_2(L_1), e\hat{U}_C, \frac{L_3}{L_2+L_3}\right) + \\ &\bar{\phi}_1\left(\frac{L_3}{L_2+L_3}\right)\alpha(\hat{r})\left[(U_A - 2U_B + U_C) + (e\hat{U}_A - 2(e\hat{U}_B + e_7\bar{\phi}_2(L_1)) + e\hat{U}_C)\right], \end{aligned} \tag{52}$$

where $\alpha(\hat{r})$ is calculated by using the modified solution values, i.e., $\hat{U}_A$ instead of U_A, etc. The errors $e\hat{U}_{I,2}$ and $e\hat{U}_{I,3}$ may be similarly estimated. This expression shows how the additional errors introduced by the one-dimensional interpolations, $e\hat{U}_A, e\hat{U}_B$, and $e\hat{U}_C$ combine with the centroid error to introduce an additional error, of the same order as the one-dimensional errors, into the interpolant. The first term in (52) is the quadratic interpolant of the errors at points A, B, and C while the second term consists of a term similar to that arising from ensuring that the polynomial q_1 is data-bounded, e.g., see (23), but with errors in the data values.

Regarding the continuity of the interpolant defined in this way, the polynomials defined along the edges of each triangle and through its centroid are clearly continuous. The question remains as to whether or not the dependency of the modified polynomials on r and s might cause discontinuities. Both r and s depend in a fixed well-defined way on the nodal data values and so the new interpolant is a composition of continuous functions of (x, y) and so is continuous.

7. Numerical examples. In order to illustrate the properties of the interpolant three simple examples defined on a triangle with vertices at $(0,0), (0,1)$, and $(1,0)$ are used. The first problem has a maximum at (x_0, y_0) the second problem has a minimum at (x_0, y_0) while the third problem has a ridge of maximum values across the triangle.

Problem 1.

$$u(x,y) = \frac{1}{0.1 + (x-x_0)^2 + (y-y_0)^2}, \quad x_0 = 0.25, \quad y_0 = 0.1. \tag{53}$$

Problem 2.

$$u(x,y) = 10[(x-x_0)^2 + (y-y_0)^2], \quad x_0 = \frac{1}{3}, \quad y_0 = \frac{1}{3}. \tag{54}$$

Problem 3.

$$u(x,y) = 2 - \frac{1}{2}(x + y_0 y)^2 + x_0\ (x + y_0 y), \quad x_0 = -\frac{1}{3}, \quad y_0 = -2.0. \tag{55}$$

Figures 2–10 display the results for these three problems, and the associated numerical tables in Appendices A, B, and C show the solution values to two significant figures at the mesh points $(0.1i, 0.1j)$, where $i = 0, \ldots, 10$, $j = 0, \ldots, 10$, and $i+j \leq 10$. For each of the three problems the original quadratic and new quadratic interpolants are shown. For Problems 2 and 3 the exact solution is identical to the standard quadratic interpolant and so is not shown. In the case of Problem 1 the exact solution is shown.

In the case of Problem 1 the original interpolant has a maximum value of about 9.8 along the line $x = 0$, whereas the new polynomial does not allow the solution to rise above the largest nodal value of 5.7971, thus giving rise to the large maximum error shown in Figure 4.

For Problem 2 the data and original interpolant are both zero at the centroid $x = \frac{1}{3}, y = \frac{1}{3}$ but the new interpolant (correctly given its intent) does not allow the solution values to dip below the smallest data value of 0.56.

In the case of Problem 3, the original quadratic interpolant peaks at 2.1, but the new polynomial remains bounded by the maximum nodal value of 2.041 at point 6. Overall, these results show that the new interpolant remains bounded between the maximum and minimum data values.

8. Extension to tetrahedral elements. The method described above is readily extended to the case of the standard quadratic interpolant on tetrahedra [12] in which each exterior triangular face has the same data points as the triangular quadratic interpolant in Figure 1. The data points are shown in Figure 11 and are numbered 1 to 10. Suppose that we wish to find the value of the interpolant at a point lying on the QRS triangle defined by the points Q, R, and S on which the volume coordinate L_3 is constant. Let QR, RS, and QS be the midpoints of the lines between Q and R, R and S, and between Q and S, respectively. Furthermore let c_1 be the centroid of the triangular face defined by points 1, 2, 3; c_2 be the centroid of the triangle defined by the points 1, 3, 8; and c_3 be the centroid of the triangle defined by the points 2, 3, 8.

The central idea is to use the monotone positive polynomials p_1 and p_2 defined in section 2 to compute sufficient values on the triangle defined by Q, R, and S so that the triangular interpolant described in section 3 may be used to find the value of the interpolant. The values at Q, R, and S are computed using the p_1 polynomials along the tetrahedral edges, while the values at the midpoints QR, RS, and QS are computed using the p_2 polynomial and the centroid values U_{c_1}, U_{c_2}, and U_{c_3}, respectively, on the exterior faces of the tetrahedron. The three centroid values are calculated in the same way as in section 5. In other words,

$$\begin{aligned}
U_Q &= p_1(U_2, U_5, U_3, QL_3),\\
U_S &= p_1(U_8, U_7, U_3, SL_3),\\
U_R &= p_1(U_1, U_6, U_3, RL_3),\\
U_{QR} &= p_2(U_4, U_{c_1}, U_3, QRL_3),\\
U_{RS} &= p_2(U_9, U_{c_2}, U_3, RSL_3),\\
U_{QS} &= p_2(U_{10}, U_{c_3}, U_3, QSL_3),
\end{aligned}$$

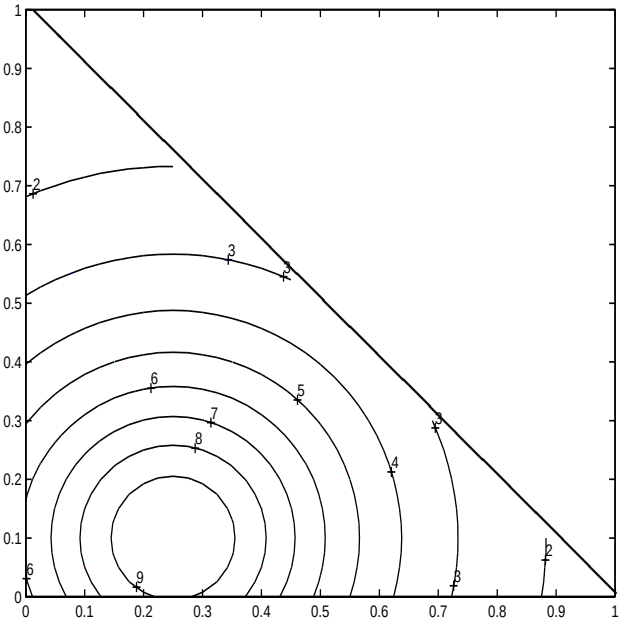


FIG. 2. *Problem* 1. *Original values.*

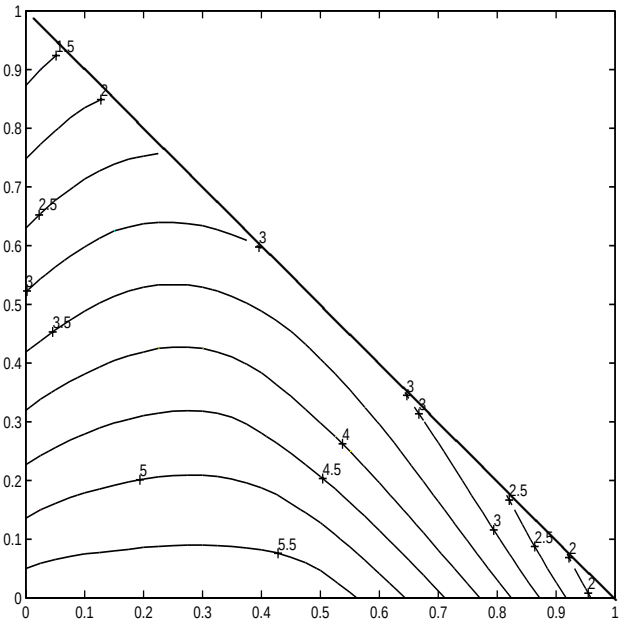


FIG. 3. *Problem* 1. *New interpolant.*

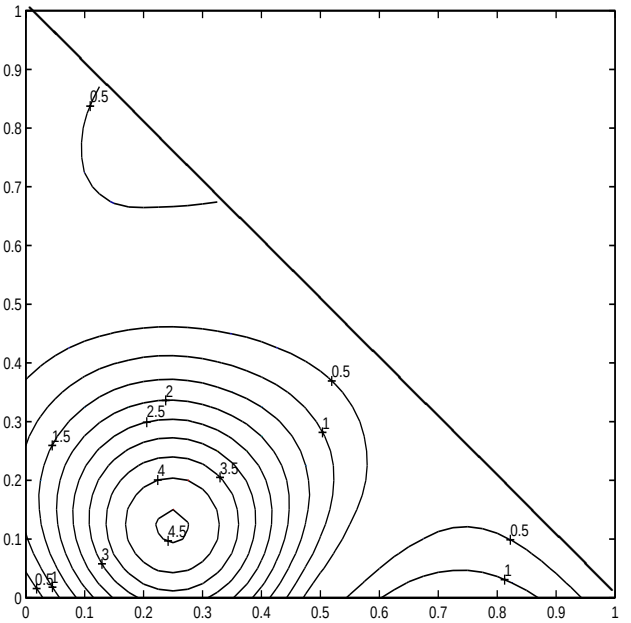


FIG. 4. *Problem* 1. *Errors in new interpolant.*

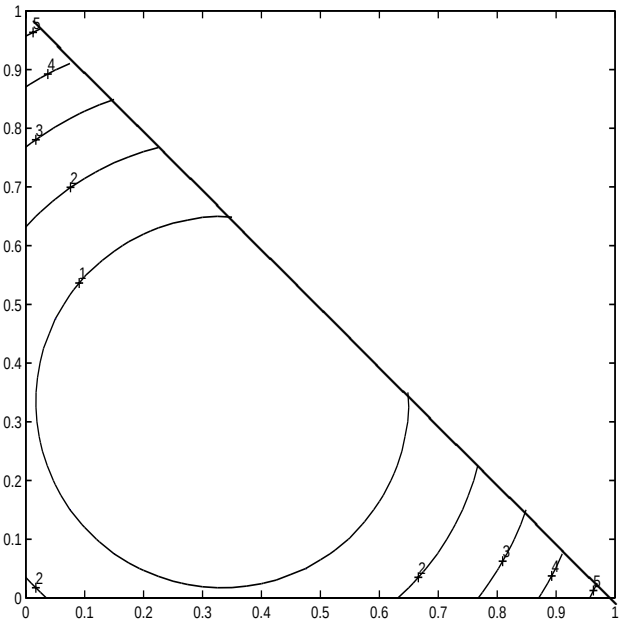


FIG. 5. *Problem* 2. *Data values and original interpolant.*

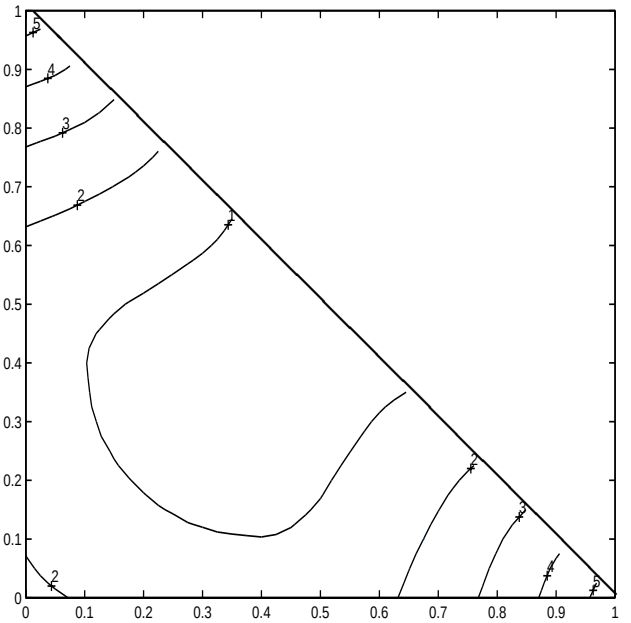


FIG. 6. *Problem* 2. *New positive interpolant.*

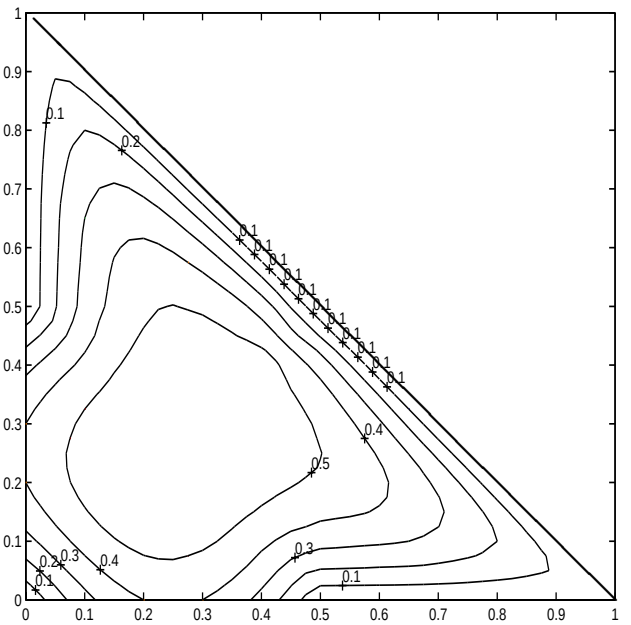


FIG. 7. *Problem* 2. *Errors in new interpolant.*

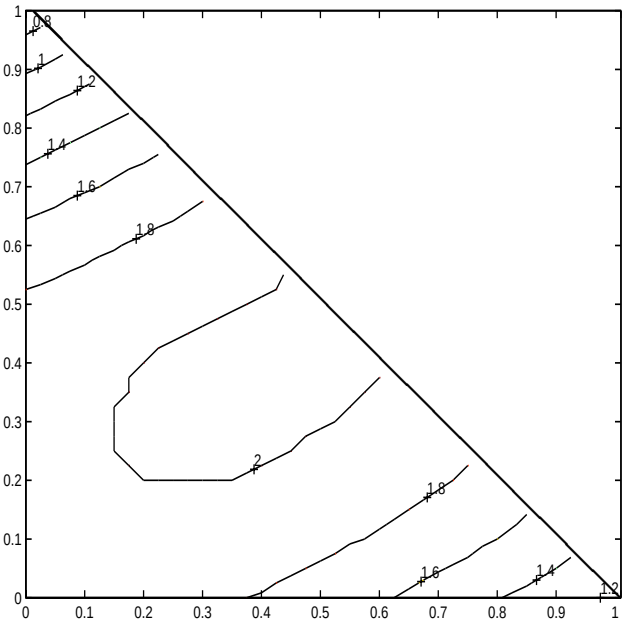


FIG. 8. *Problem* 3. *Data values and original interpolant.*

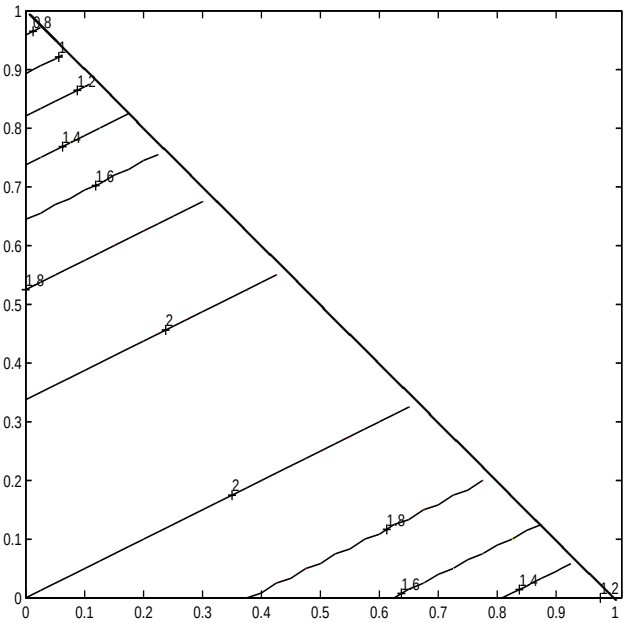


FIG. 9. *Problem* 3. *New interpolant.*

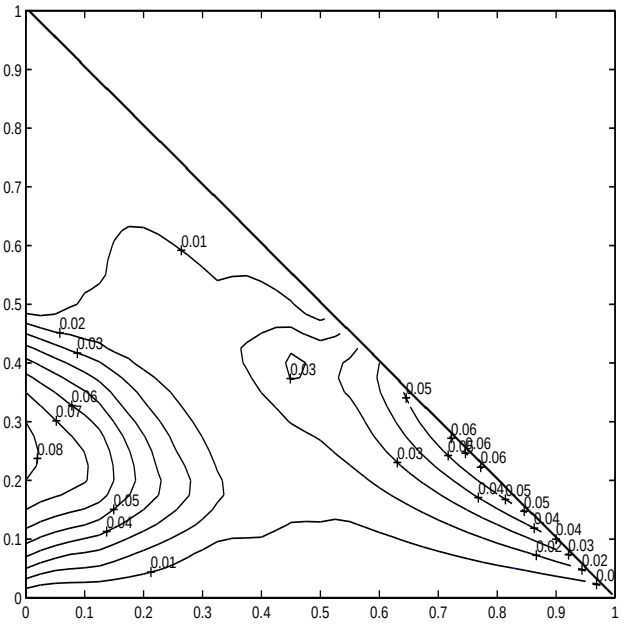


FIG. 10. *Problem* 3. *Errors in new interpolant.*

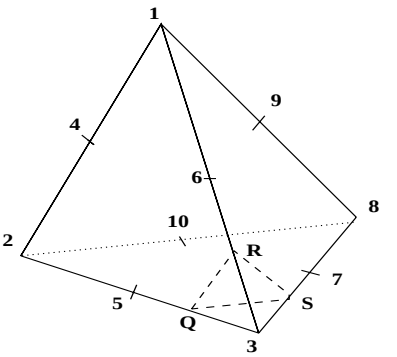


FIG. 11. *Example tetrahedron.*

where QL_3 is the L_3 coordinate of point Q, and similarly for the other five values of L_3 .

The final step is to use the values $U_Q, U_R, U_S, U_{QR}, U_{RS}, U_{QS}$ in the positive triangular interpolant described in section 6 to compute the required value. The accuracy and positivity properties follow from the properties of the individual linear and triangular interpolants.

As in the case of triangles where for any point there are three such interpolants of this type there are four such tetrahedral interpolants, the one described being associated with the volume coordinate L_3. Again as in the triangular case, providing that the standard quadratics q_1, q_2, and the standard quadratic triangular interpolant defined by (1) are used, then all four interpolants will give the same answer. When the positivity preserving polynomials p_1 and p_2 and the positivity preserving triangular

interpolant defined by (36) are used the four values will no longer be identical, though all will be in the range of the data. One solution is to average the four values. Alternatively the closest value to the original quadratic could be used.

9. Conclusions and extensions. This new method for quadratic interpolation on triangles and tetrahedra, based on combinations of linear monotone quadratic polynomials, will preserve positivity in the data in that the interpolant created will be bounded by the maximum and minimum function values used to define it.

An obvious extension of the approach is to consider the interpolant values on a triangle by global rather than local data values, e.g., [6]. This may be done by locating extrema on each edge, if necessary replacing them by global extremal data values, and using a data-bounded quadratic to interpolate in between these new values on that part of the subedge where the extremal value lies. In this way, if a nonnodal extremal value outside the data bounds was detected, each quadratic edge polynomial would be decomposed into three piecewise quadratic polynomials.

Appendix A. *Problem* 1. *Data values.*

```
1.0
1.2  1.3
1.5  1.6  1.7
1.9  2.1  2.2  2.2
2.4  2.7  2.8  2.8  2.7
3.1  3.5  3.8  3.8  3.5  3.1
4.0  4.7  5.2  5.2  4.7  4.0  3.2
4.9  6.2  7.0  7.0  6.2  4.9  3.8  2.9
5.8  7.5  8.9  8.9  7.5  5.8  4.3  3.2  2.4
6.2  8.2  9.8  9.8  8.2  6.2  4.5  3.3  2.5  1.9
5.8  7.5  8.9  8.9  7.5  5.8  4.3  3.2  2.4  1.9  1.5
```

Problem 1. *Original quadratic interpolant values.*
Data values are shown as [ ].

```
[1.0]
1.4  1.7
1.8  2.1  2.3
2.2  2.5  2.7  2.7
2.6  3.0  3.2  3.2  3.0
[3.1] 3.4  3.6  3.6  3.4 [3.1]
3.6  3.9  4.1  4.1  3.9  3.6  3.1
4.1  4.4  4.6  4.6  4.4  4.1  3.6  2.9
4.6  5.0  5.2  5.2  5.0  4.6  4.1  3.4  2.6
5.2  5.6  5.7  5.7  5.6  5.2  4.7  4.0  3.1  2.1
[5.8] 6.1  6.3  6.3  6.1 [5.8] 5.3  4.6  3.7  2.7 [1.5]
```

Problem 1. *New positive quadratic interpolant values.*
Data values are shown as [ ].

```
[1.0]
1.4  1.7
1.8  2.1  2.3
2.2  2.6  2.7  2.7
2.6  3.0  3.2  3.2  3.0
[3.1] 3.5  3.6  3.6  3.4 [3.1]
```

```
3.6  3.9  4.1  4.1  3.9  3.5  3.0
4.1  4.4  4.5  4.6  4.4  4.0  3.5  2.8
4.6  4.9  5.0  5.0  4.9  4.5  4.0  3.3  2.5
5.2  5.4  5.4  5.5  5.4  5.2  4.6  3.9  3.0  2.1
[5.8] 5.8  5.8  5.8  5.8 [5.8] 5.3  4.6  3.7  2.7 [1.5]
```

Appendix B. *Problem* 2. *Data values and original interpolant values.*
Data values used are shown as [ ].

```
[5.6]
4.3   3.8
3.3   2.7   2.4
2.5   1.9   1.5   1.4
1.8   1.3   .89   .72   .76
[1.4]  .82   .46   .29   .32  [.56]
1.2   .59   .22   .06   .09   .32   .76
1.1   .56   .19   .02   .06   .29   .72   1.4
1.3   .72   .36   .19   .22   .46   .89   1.5   2.4
1.7   1.1   .72   .56   .59   .82   1.3   1.9   2.7   3.8
[2.2]  1.7   1.3   1.1   1.2  [1.4]  1.8   2.5   3.3   4.3  [5.6]
```

Problem 2. *New positive quadratic interpolant values.*
Data values used are shown as [ ].

```
[5.6]
4.3   3.8
3.3   3.0   2.4
2.5   2.2   1.9   1.4
1.8   1.6   1.3   1.0   .76
[1.4]  1.2   .94   .77   .67  [.56]
1.4   1.0   .75   .63   .67   .67   .76
1.5   1.1   .77   .60   .61   .77   1.0   1.4
1.7   1.3   .94   .76   .75   .94   1.3   1.8   2.4
1.9   1.6   1.3   1.1   1.0   1.2   1.6   2.2   3.0   3.8
[2.2]  1.9   1.7   1.5   1.4  [1.4]  1.8   2.5   3.3   4.3  [5.6]
```

Problem 2. *Errors in new positive quadratic interpolant.*

```
[0.0]
0.0   0.0
0.0   .20   0.0
0.0   .27   .28   0.0
0.0   .32   .42   .31   0.0
[0.0]  .33   .48   .48   .35  [0.0]
.27   .42   .53   .54   .54   .35   0.0
.40   .51   .58   .58   .55   .48   .32   0.0
.40   .53   .59   .58   .53   .48   .42   .27   0.0
.27   .43   .53   .51   .42   .33   .31   .27   .20   0.0
[0.0]  .27   .40   .40   .27  [0.0]  0.0   0.0   0.0   0.0  [0.0]
```

Appendix C. *Problem* 3. *Data values and original interpolant.*
Data values used are shown as [ ].

```
[.67]
 .98   1.1
 1.3   1.4   1.5
 1.5   1.6   1.7   1.8
 1.7   1.8   1.8   1.9   1.9
[1.8]  1.9   1.9   2.0   2.0  [2.0]
 1.9   2.0   2.0   2.0   2.1   2.1   2.0
 2.0   2.0   2.1   2.1   2.0   2.0   2.0   2.0
 2.1   2.1   2.0   2.0   2.0   2.0   1.9   1.9   1.8
 2.0   2.0   2.0   2.0   1.9   1.9   1.8   1.7   1.6   1.5
[2.0]  2.0   1.9   1.9   1.8  [1.7]  1.6   1.5   1.4   1.3  [1.2]
```

Problem 3. *New positive quadratic interpolant values.*
Data values used are shown as [ ].

```
[.67]
 .98   1.1
 1.3   1.4   1.5
 1.5   1.6   1.7   1.8
 1.7   1.8   1.8   1.9   1.9
[1.8]  1.9   1.9   2.0   2.0  [2.0]
 1.9   2.0   2.0   2.0   2.0   2.0   2.0
 1.9   2.0   2.0   2.0   2.0   2.0   2.0   1.9
 2.0   2.0   2.0   2.0   2.0   1.9   1.9   1.8   1.7
 2.0   2.0   2.0   1.9   1.9   1.8   1.8   1.7   1.6   1.5
[2.0]  2.0   1.9   1.9   1.8  [1.7]  1.6   1.5   1.4   1.3  [1.2]
```

Problem 3. *Errors in new positive quadratic interpolant.*

```
[0.0]
 0.0   0.0
 0.0   .005  0.0
 0.0   .007  .007  0.0
 0.0   .008  .01   .008  0.0
[0.0]  .01   .01   .01   .01  [0.0]
 .05   .03   .02   .01   .02   .03   .04
 .08   .06   .04   .02   .01   .02   .03   .06
 .08   .07   .05   .03   .01   .01   .02   .04   .06
 .05   .04   .03   .01   .01   .008  .008  .01   .02   .04
[0.0]  0.0    0.0  0.0   0.0  [0.0]  0.0   0.0   0.0   0.0  [0.0]
```

In this case the original quadratic interpolant peaks at 2.1, but the new polynomial remains bounded by the maximum nodal value of 2.041 at point 6.

Acknowledgements. The author would like to thank Ken Brodlie for pointing out the work of Barnhill et al. at a critical moment. The referees are also thanked for their thoughtful comments including suggesting the use of the term "data-bounded."

REFERENCES

[1] R. Abgrall, *Design of an Essentially Non-Oscillatory Reconstruction Procedure on Finite-Element Type Meshes*, ICASE report 91-84, 1991; revised form INRIA report 1592, 1992.
[2] R. E. Barnhill, G. Birkhoff, and W. J. Gordon, *Smooth interpolation on triangles*, J. Approx. Theory, 8 (1973), pp. 114–128.

[3] R. E. Barnhill and L. Mansfield, *Error bounds for smooth interpolation on triangles*, J. Approx. Theory, 11 (1974), pp. 306–318.
[4] T. G. Barth, *On Unstructured Grids and Solvers*, Technical report, von Karman Institute for Fluid Dynamics, Lecture Series 1990-03, 1990.
[5] M. Berzins and J. M. Ware, *Positive cell centered finite volume discretization methods for hyperbolic equations on irregular meshes*, Appl. Numer. Math., 16 (1995), pp. 417–438.
[6] K. W. Brodlie and P. Mashwama, *Controlled interpolation for scientific visualization*, in Scientific Visualization, in Proceedings of IEEE Conference on Visualization, G. M. Nielson et al., eds., IEEE Computer Society, Los Alamitos, CA, 1997, pp. 253–275.
[7] L. J. Durlofsky, B. Enquist, and S. Osher, *Triangle based adaptive stencils for the solution of hyperbolic conservation laws*, J. Comput. Phys., 98 (1992), pp. 64–73.
[8] R. H. Gallagher, *Finite Element Analysis Fundamentals*, Prentice-Hall, London, 1975.
[9] C. Johnson, *Numerical Solutions of p.d.e.s by the Finite Element Methods*, Cambridge University Press, Cambridge, UK, 1988.
[10] P. Pratt and M. Berzins, *Shock preserving quadratic interpolation for visualization on triangular meshes*, Comput. Graphics, 385 (1997), pp. 723–730.
[11] V. Venkatakrishnan, *Perspective on unstructured grid flow solvers*, AIAA Journal, 34 (1996), pp. 533–547.
[12] O. C. Zienkiewicz and R. L. Taylor, *The Finite Element Method*, McGraw-Hill, London, 1989.

SIAM J. SCI. COMPUT.
Vol. 22, No. 1, pp. 198–217
© 2000 Society for Industrial and Applied Mathematics

ARE BILINEAR QUADRILATERALS BETTER THAN LINEAR TRIANGLES?*

E. F. D'AZEVEDO†

Abstract. This paper compares the theoretical effectiveness of bilinear approximation over quadrilaterals with linear approximation over triangles. Anisotropic mesh transformation is used to generate asymptotically optimally efficient meshes for piecewise linear interpolation over triangles and bilinear interpolation over quadrilaterals. For approximating a convex function, although bilinear quadrilaterals are more efficient, linear triangles are more accurate and may be preferred in finite element computations; whereas for saddle-shaped functions, quadrilaterals may offer a higher-order approximation on a well-designed mesh. A surprising finding is different grid orientations may yield an order of magnitude improvement in approximation accuracy

Key words. mesh generation, interpolation, quadrilateral patch, triangulation, bilinear approximation

AMS subject classifications. 65D05, 65L50

PII. S106482759630406X

1. Introduction.

This paper compares the theoretical effectiveness of bilinear approximation over quadrilaterals with linear approximation over triangles. The novelty is in the use of anisotropic mesh transformation to generate asymptotically optimally efficient meshes in the comparison. Elementary analysis based on a simple quadratic data model is used. Although both linear and bilinear interpolants are $O(h^2)$ accurate, the results suggest linear triangles are *always* more accurate than general convex bilinear quadrilaterals in approximating a convex function but bilinear approximation may offer a higher-order approximation for saddle-shaped functions on a well-designed mesh. A surprising finding is different grid orientations may yield an order of magnitude "superconvergence" improvement in approximation accuracy. This work is a basic study on optimal meshes with the intention of gaining insight into the more complex meshing problems in finite element analysis.

We consider the problem of interpolating a given smooth data function with continuous piecewise linear triangles or bilinear quadrilaterals over a domain to satisfy a given error tolerance. A mesh that achieves this error tolerance with the *fewest* elements is defined to be optimally efficient. Intuitively, one would expect smaller and denser elements in regions where the function has sharp peaks or large variations. Since each convex quadrilateral can be split across either one of the diagonals into two triangles, one can imagine embedding a refined triangular mesh within the quadrilateral mesh. A practical question arises as to whether the bilin-

*Received by the editors May 20, 1996; accepted for publication (in revised form) July 30, 1999; published electronically June 13, 2000. This work was supported by the Applied Mathematical Sciences subprogram of the Office of Energy Research, U.S. Department of Energy, under contract DE-AC05-84OR21400 with Lockheed Martin Energy Systems, Inc., and in part by the Information Technology Research Centre, which is funded by the Province of Ontario. This work was performed by an employee of the U.S. Government or under U.S. Government contract. The U.S. Government retains a nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or allow others to do so, for U.S. Government purposes. Copyright is owned by SIAM to the extent not limited by these rights.

http://www.siam.org/journals/sisc/22-1/30406.html

†Mathematical Sciences Section, Oak Ridge National Laboratory, P. O. Box 2008, Building 6012, Oak Ridge, TN 37831–6367 (e6d@ornl.gov).

ear approximation over quadrilaterals or linear approximation over triangles is more effective.

To make a fair comparison, we need to compare bilinear approximation over an "optimal" quadrilateral mesh versus linear approximation over an "optimal" triangular mesh. Provably optimal triangular meshes [7, 9] have been produced by anisotropic mesh transformation.

Anisotropic mesh generation using a Riemannian metric is emerging as an effective technique for unstructured grid generation where the vertex distribution is highly nonuniform. The central idea is to control the element shapes and sizes by specifying a symmetric metric tensor that measures the approximation error. The metric tensor determines the corresponding anisotropic transformation. The anisotropic mesh is then the image of a uniform mesh of optimal shape elements under the anisotropic transformation. Simpson [18] gives a survey on anisotropic meshes.

Nadler [14], D'Azevedo and Simpson [8, 9], and D'Azevedo [7] have studied *local* anisotropic transformation for the generation of optimally efficient triangular meshes. Peraire et al. [15], Fortin and coworkers [10, 1], Borouchaki and coworkers [2, 3, 4] have used a local metric in anisotropic mesh generation for dynamic remeshing in solving compressible flow problems. In these works, a local metric is used to control the size and shape of elements on the computational mesh.

Several techniques are used to generate anisotropic meshes. Bossen [5] uses a physically-based model of interacting "particles" to define vertices of a triangulation. The attractive (repulsive) forces are derived using a local Riemannian metric. The particles are triangulated to produce a modified Delaunay mesh in the Riemannian space defined by the metric. Shimada [16, 17] uses a similar physically-based close packing of elliptic "bubbles" for generating isotropic meshes. The ellipses are nearly circular bubbles over the transformed space. Ait-Ali-Yahia et al. [1] use a physical spring analogy to perturb points in a structured mesh to generate an anisotropic quadrilateral mesh. George, and George and Hecht [11, 12] use a point insertion process and mesh smoothing to generate an anisotropic triangular mesh. Triangles can be further merged to form a quadrilateral mesh. Common to the above described approaches, the metric tensor is interpolated from piecewise constant values defined over an underlying "background" mesh.

The approach used in this investigation has two major differences. First, we assume a *continuous* Hessian metric is available and we derive a *continuous* global coordinate transformation to the *isotropic* space. The underlying theory relies on a classical result in differential geometry described in section 2.3. A well-shaped (regular) mesh over the isotropic space is transformed back to the original physical space to produce an anisotropic mesh. Second, the Hessian "metric" need not be positive definite. In fact, it is precisely the nonpositive definite case that yields the interesting superconvergent result.

An outline of the paper follows. In section 2, we review the key ideas in [7] for generating optimally efficient triangular meshes. In section 3, we consider error properties of bilinear interpolation. We consider the optimal geometry for quadrilateral patches in section 3.2. We compare the effectiveness of quadrilaterals versus triangular meshes using the local quadratic model in section 4. Numerical experiments and the results are described in section 5. Finally section 6 gives a brief summary.

2. Triangular patch. This section is a brief review of the basic ideas in [7] for determining optimal triangle geometry. We show a linear transformation of a regular mesh of optimal-shape triangles yields an optimally efficient mesh for interpolating a

quadratic function.

2.1. Quadratic model. We shall consider a local analysis where we assume the data function $f(x,y)$ in the neighborhood of (x_c, y_c) is well approximated by its quadratic Taylor expansion,

$$\begin{aligned} f(x,y) &= f(x_c + dx, y_c + dy) \\ &\approx f(x_c, y_c) + \nabla f(x_c, y_c)[dx, dy] + \frac{1}{2}[dx, dy]H[dx, dy]^t. \end{aligned} \tag{2.1}$$

Let the error formula be $E_T(x,y) = p_\ell(x,y) - f(x,y)$, where $p_\ell(x,y)$ is the linear interpolant. By our assumption, $E_T(x,y)$ is a quadratic function and level curves for $E_T(x,y) = c$ form a family of conics with a common center at (x_c, y_c). They form a family of ellipses if $\det(H) > 0$ and hyperbolas if $\det(H) < 0$. Note by the interpolation condition, the curve $E_T(x,y) = 0$ passes through all vertices of the triangle. If $\det(H) > 0$ (conic is an ellipse), then $E_T(x,y)$ attains the local maximum at the center (x_c, y_c); otherwise, $\det(H) < 0$ (conic is a hyperbola) the maximum error is attained at the midpoint of an edge. The error at a displacement from the center is given by

$$E_T(x_c + dx, y_c + dy) = \mathcal{E}_T - \frac{1}{2}[dx, dy]H[dx, dy]^t, \quad \mathcal{E}_T = E_T(x_c, y_c). \tag{2.2}$$

The key insight in [7] is in interpreting the Hessian matrix H in (2.2) as a symmetric metric tensor. Let the symmetric Hessian matrix be diagonalizable as

$$\begin{aligned} H &= Q^t \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} Q = S^t \begin{bmatrix} 1 & 0 \\ 0 & \epsilon \end{bmatrix} S, \quad \text{where } \epsilon = \operatorname{sign}(\det(H)), \\ S &= \begin{bmatrix} \sqrt{|\lambda_1|} & 0 \\ 0 & \sqrt{|\lambda_2|} \end{bmatrix} Q, \quad \text{and } Q \text{ is orthogonal, } Q^t Q = I. \end{aligned} \tag{2.3}$$

Note that transformation S is essentially a rotation to align eigenvectors along the coordinate axes then followed by a simple scaling. Under this transformation S, the expression $[dx, dy]H[dx, dy]^t$ reduces to $(d\tilde{x})^2 + \epsilon(d\tilde{y})^2$, where $[\tilde{x}, \tilde{y}]^t = S[x, y]^t$. The error function can be rewritten as

$$\begin{aligned} E_T(x_c + dx, y_c + dy) &= \mathcal{E}_T - \frac{1}{2}[dx, dy]H[dx, dy]^t \\ &= \mathcal{E}_T - \frac{1}{2}\left((d\tilde{x})^2 + \epsilon(d\tilde{y})^2\right) \\ &= \tilde{E}_T(\tilde{x}_c + d\tilde{x}, \tilde{y}_c + d\tilde{y}), \end{aligned} \tag{2.4}$$

where $\tilde{E}_T(\tilde{x}, \tilde{y})$ denotes the corresponding error function under transformation S in $(\tilde{x}, \tilde{y})$-space. The error expression $\tilde{E}_T(\tilde{x}, \tilde{y})$ has no preferred direction (except for the sign), hence we shall call the $(\tilde{x}, \tilde{y})$-space the "isotropic" space.

2.2. Optimal shape. In the following, we shall determine the best triangle shape that minimizes the interpolation error. We can determine the "efficiency" of the elements by computing their ratio of error to area. A small ratio indicates a more efficient element, i.e., one can achieve a lower error tolerance and tile the domain with about the same number of elements.

We consider first the case $f(x,y)$ is convex ($\det(H) > 0, \epsilon = 1$) and level curves or contours of $\tilde{E}_T(\tilde{x},\tilde{y})$ are concentric circles given by

$$\tilde{E}_T(\tilde{x}_c + d\tilde{x}, \tilde{y}_c + d\tilde{y}) = \mathcal{E}_T - \frac{1}{2}\left((d\tilde{x})^2 + (d\tilde{y})^2\right). \tag{2.5}$$

Let $\tilde{T}$ be the transformed image of triangle T over the isotropic space, with vertices at $(\tilde{x}_1,\tilde{y}_1)$, $(\tilde{x}_2,\tilde{y}_2)$, and $(\tilde{x}_3,\tilde{y}_3)$. The circumcircle of $\tilde{T}$ corresponds to the level curve of value zero. Hence the radius of this circumcircle is $\mathrm{sqrt}(2|\mathcal{E}_T|)$ and relates directly to the maximum error attainable (at the center). If this center is exterior to triangle T, the maximum error is attained at the midpoint of the longest edge (of length L) with value $L^2/8$. We can easily see that an equilateral triangle covers the most area for a fixed circumcircle, therefore an equilateral triangle for $\tilde{T}$ is of optimal shape.

If $f(x,y)$ is not convex but has a saddle-shaped graph ($\det(H) < 0$, $\epsilon = -1$), then

$$\begin{aligned}\tilde{E}_T(\tilde{x},\tilde{y}) &= \tilde{E}_T(\tilde{x}_c + d\tilde{x}, \tilde{y}_c + d\tilde{y}) \\ &= \mathcal{E}_T - \frac{1}{2}\left((d\tilde{x})^2 - (d\tilde{y})^2\right) \\ &= \mathcal{E}_T - \frac{1}{2}\left((\tilde{x}-\tilde{x}_c)^2 - (\tilde{y}-\tilde{y}_c)^2\right).\end{aligned} \tag{2.6}$$

We note that the error function $\tilde{E}_T(\tilde{x},\tilde{y})$ is a harmonic function and thus attains its extrema on the boundary of $\tilde{T}$. By calculus, we can show that the local extrema along edge $(\tilde{x}_i,\tilde{y}_i)$, $(\tilde{x}_j,\tilde{y}_j)$ is attained at the midpoint with value

$$\tilde{E}\left(\frac{\tilde{x}_i+\tilde{x}_j}{2}, \frac{\tilde{y}_i+\tilde{y}_j}{2}\right) = \frac{1}{8}\left|(\tilde{x}_i-\tilde{x}_j)^2 - (\tilde{y}_i-\tilde{y}_j)^2\right|.$$

The details for finding the optimal-shape triangle in this case are found in [7]. The optimal-shape triangle geometry that minimizes the efficiency ratio (error/area) is not unique, but the same maximum error is attained at the midpoint of each edge.

From the above two results on optimal-shape triangles, we see that a *regular* mesh of optimal-shape triangles over the isotropic $(\tilde{x},\tilde{y})$-space corresponds to an optimally efficient mesh over the original (x,y)-space. Every triangle attains the same maximum error; moreover, these triangles cover the most area for the error attained and so are optimally efficient. Since the linear transformation S is basically a rotation followed by a rescaling of coordinate axes, we find the areas of triangles are scaled accordingly. Hence the inverse transformation S^{-1} maps this regular mesh to produce an optimally efficient mesh in the original (x,y)-space.

2.3. Differential geometry. The *constant* Hessian matrix H in (2.1) determines the coordinate transformation S that maps $[\tilde{x},\tilde{y}]^t = S[x,y]^t$ so that

$$[dx,dy]H[dx,dy]^t = (d\tilde{x}^2 + \epsilon d\tilde{y}^2).$$

We may view the Hessian matrix $H(x,y)$ as a local metric for measuring the interpolation error $[dx,dy]H[dx,dy]^t$. If a function is approximated by piecewise quadratic patches, then each patch would define a constant Hessian matrix and a corresponding coordinate transformation. In the limit where each quadratic patch is infinitestimally small, we need to determine a *continuous* transformation $(\tilde{x}(x,y),\tilde{y}(x,y))$, that satisfies $[dx,dy]H(x,y)[dx,dy]^t = d\tilde{x}^2 + \epsilon d\tilde{y}^2$, where $H(x,y)$ is the Hessian matrix of a general function.

The necessary and sufficient conditions for finding the anisotropic coordinate transformation $(\tilde{x}(x,y), \tilde{y}(x,y))$ are given by a classical result in differential geometry for characterizing a "flat" space [19]: that the Riemann–Christoffel tensor formed from the metric tensor $H = \{h_{ij}\}$ is identically zero. This condition reduces to

$$0 = \Gamma = \det \begin{pmatrix} h_{11} & \frac{\partial h_{11}}{\partial x} & \frac{\partial h_{11}}{\partial y} \\ h_{12} & \frac{\partial h_{12}}{\partial x} & \frac{\partial h_{12}}{\partial y} \\ h_{22} & \frac{\partial h_{22}}{\partial x} & \frac{\partial h_{22}}{\partial y} \end{pmatrix} . \tag{2.7}$$

In this case, a sufficient condition is for $H = \{h_{ij}\}$ to satisfy

$$K_1 h_{11} + K_2 h_{12} + K_3 h_{22} = 0$$

for some constants K_1, K_2, K_3. The coordinate transformation $(\tilde{x}(x,y), \tilde{y}(x,y))$ may be found by solving an initial value ordinary differential equation. Again, the inverse transformation $(x(\tilde{x},\tilde{y}), y(\tilde{x},\tilde{y}))$ maps a regular mesh of optimal shaped triangles to yield an optimally efficient mesh.

3. Quadrilateral patch. In this section, we follow the approach of using a simple quadratic data function in deriving the error term for a triangular patch to derive the error term for bilinear approximation over a quadrilateral patch.

3.1. Bilinear interpolant. We shall use the isoparametric formulation (commonly used in finite element analysis) by considering basis functions over the normalized (p,q)-space over the unit square, $0 \le p, q \le 1$. Basis functions are

$$\begin{aligned} &\phi_1(p,q) = (1-p)(1-q), && \phi_2(p,q) = p(1-q), \\ &\phi_3(p,q) = pq, && \phi_4(p,q) = (1-p)q \end{aligned} \tag{3.1}$$

that satisfy $\phi_i(x_j, y_j) = \delta_{ij}$ and sum to one, $1 = \sum_{i=1}^{i=4} \phi_i(p,q)$.

Mapping from (p,q) to the original (x,y)-space is by

$$\begin{aligned} x(p,q) &= x_1\phi_1(p,q) + x_2\phi_2(p,q) + x_3\phi_3(p,q) + x_4\phi_4(p,q), \\ y(p,q) &= y_1\phi_1(p,q) + y_2\phi_2(p,q) + y_3\phi_3(p,q) + y_4\phi_4(p,q) \end{aligned} \tag{3.2}$$

that maps vertex $(0,0)$ to (x_1,y_1), $(1,0)$ to (x_2,y_2), $(1,1)$ to (x_3,y_3), and $(0,1)$ to (x_4,y_4). The bilinear interpolant (over (p,q)-space) is given by

$$p_b(x(p,q), y(p,q)) = \sum_{i=1}^{i=4} f(x_i,y_i)\phi_i(p,q) \ . \tag{3.3}$$

3.2. Optimal shape. In the following, we shall determine the best quadrilateral shape that minimizes the interpolation error. The error function for quadratic interpolation over a *parallelogram* can be shown by direct algebraic expansion (see Appendix A) to be

$$\begin{aligned} E_Q(p,q) &= p_b(x(p,q), y(p,q)) - f(x(p,q), y(p,q)) \\ &= \mathcal{E}_Q - \frac{1}{2}\left(\mu_1(p-p_c)^2 + \mu_2(q-q_c)^2\right), \end{aligned} \tag{3.4}$$

with centroid at $[p_c, q_c] = [1/2, 1/2]$,

$$[u_x, u_y] = [x_2 - x_1, y_2 - y_1], \quad [v_x, v_y] = [x_4 - x_1, y_4 - y_1],$$

$$\begin{aligned}
\mathcal{E}_Q &= E_Q(p_c, q_c) = \frac{1}{8}\left(\mu_1 + \mu_2\right), \\
0 &= \frac{\partial}{\partial p} E_Q(p_c, q_c) = \frac{\partial}{\partial q} E_Q(p_c, q_c), \\
\mu_1 &= [u_x, u_y] H [u_x, u_y]^t, \quad \mu_2 = [v_x, v_y] H [v_x, v_y]^t.
\end{aligned} \tag{3.5}$$

For a convex function ($\det(H) > 0$), μ_1 and μ_2 are positive, hence the maximum error is attained at the centroid $[p_c, q_c]$.

For this convex case, we can show a square over the isotropic space is of optimal shape by minimizing the efficiency ratio (error/area). Since the isoparametric bilinear interpolant (3.3) exactly fits linear functions [13], the error attained at the centroid (x_c, y_c) (which is a lower bound on the maximum error) can be written as

$$\begin{aligned}
\mathcal{E}_M &= \frac{1}{4}\left(\sum_{i=1}^{i=4} \frac{1}{2}[x_i, y_i] H [x_i, y_i]^t\right) - \frac{1}{2}[x_c, y_c] H [x_c, y_c]^t \\
&= \frac{1}{2}\left(\sum_{i=1}^{i=4} \left(\frac{1}{4}[x_i, y_i] H [x_i, y_i]^t\right) - [x_c, y_c] H [x_c, y_c]^t\right),
\end{aligned} \tag{3.6}$$

$$[x_c, y_c] = [(x_1 + x_2 + x_3 + x_4)/4, (y_1 + y_2 + y_3 + y_4)/4] \ . \tag{3.7}$$

This expression can be further simplified over the isotropic space where H is the identity

$$\begin{aligned}
\mathcal{E}_M &= \frac{1}{8}\left(\sum_{i=1}^{i=4} \left((\tilde{x}_i^2 + \tilde{y}_i^2) - (\tilde{x}_c^2 + \tilde{y}_c^2)\right)\right) \\
&= \frac{1}{8}\left((\tilde{x}_1^2 + \tilde{x}_2^2 + \tilde{x}_3^2 + \tilde{x}_4^2) - 4\tilde{x}_c^2 + (\tilde{y}_1^2 + \tilde{y}_2^2 + \tilde{y}_3^2 + \tilde{y}_4^2) - 4\tilde{y}_c^2\right) \\
&= \frac{1}{8}(L_1^2 + L_2^2 + L_3^2 + L_4^2), \quad \text{with } L_i^2 = (\tilde{x}_i - \tilde{x}_c)^2 + (\tilde{y}_i - \tilde{y}_c)^2,
\end{aligned}$$

where $[\tilde{x}_i, \tilde{y}_i]^t = S[x_i, y_i]^t$ and $[\tilde{x}_c, \tilde{y}_c]^t = S[x_c, y_c]^t$ are the corresponding coordinates over the isotropic space. The area of this transformed convex quadrilateral is (see Figure 1)

$$\text{Area} = \frac{1}{2}\left(L_1 L_2 \sin(\theta_1) + L_2 L_3 \sin(\theta_2) + L_3 L_4 \sin(\theta_3) - L_4 L_1 \sin(\theta_1 + \theta_2 + \theta_3)\right).$$

Since the isotropic transformation S in (2.3) is a rotation followed by a rescaling of the coordinate axis, the area of quadrilateral over the isotropic space is scaled by $\text{sqrt}(|\lambda_1 \lambda_2|) = \text{sqrt}(\det(H))$ (intrinsic to H). By calculus, we can show this ratio of $\mathcal{E}_M$/area is minimized and *attained* by a square with $L_1 = L_2 = L_3 = L_4$ and $\theta_1 = \theta_2 = \theta_3 = \pi/4$. Hence the most efficient shape among *all* general convex bilinear quadrilaterals is a square over the isotropic space with an efficiency ratio of 1/4.

If $f(x, y)$ is saddle-shaped ($\det(H) < 0$), the error expression for a parallelogram is still

$$E_Q(p, q) = \frac{1}{8}(\mu_1 + \mu_2) - \frac{1}{2}(\mu_1 (p - p_c)^2 + \mu_2 (q - q_c)^2) \ .$$

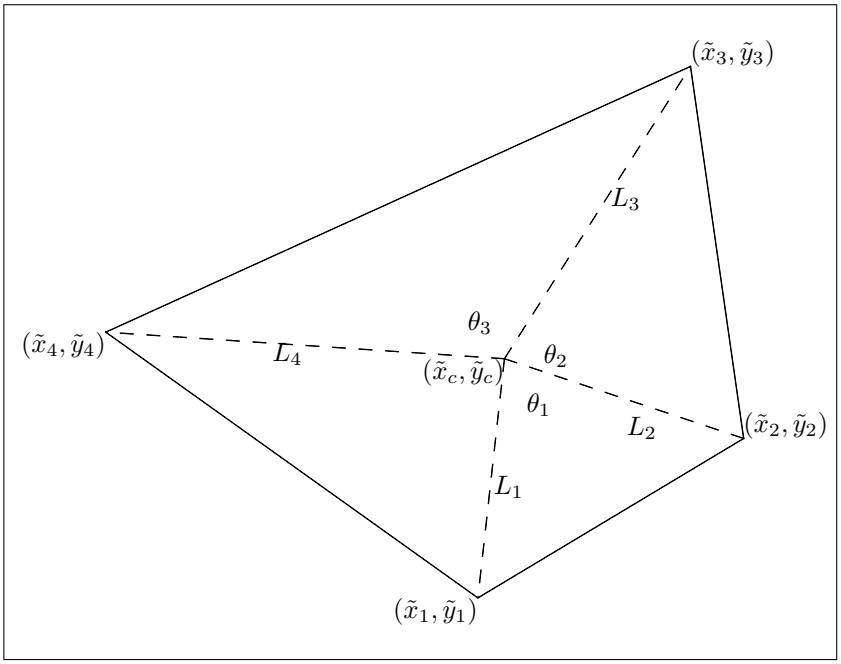


FIG. 1. *Convex quadrilateral over isotropic space.*

Under the anisotropic transformation S,

$$\mu_1 = \tilde{u}_x^2 - \tilde{u}_y^2, \quad \mu_2 = \tilde{v}_x^2 - \tilde{v}_y^2 \ , \quad \begin{bmatrix} \tilde{u}_x & \tilde{v}_x \\ \tilde{u}_y & \tilde{v}_y \end{bmatrix} = S \begin{bmatrix} u_x & v_x \\ u_y & v_y \end{bmatrix}.$$

Now both μ_1 and μ_2 vanish for

$$[\tilde{u}_x, \tilde{u}_y] = [L, L], \quad [\tilde{v}_x, \tilde{v}_y] = [-L, L], \tag{3.8}$$

which corresponds to a square rotated by $\pi/4$. The above indicates an "exact fit" ($E_Q(p,q) = 0$) if $\mu_1 = \mu_2 = 0$. This suggests bilinear approximation is a better interpolant than linear interpolation and the simple quadratic model is inadequate to fully capture the error properties in this case.

To summarize, a square over the isotropic space in any orientation is optimal for the elliptic case and a square rotated by $\pi/4$ is optimal for the hyperbolic case.

4. Comparison of quadrilaterals versus triangles. In this section, we shall show a refined triangulation produced by the Delaunay triangulation (DT) will always produce better accuracy for approximating a *convex* quadratic function. We shall apply the geometric interpretation of the maximum interpolation error over the transformed isotropic space.

THEOREM 4.1. *Any convex quadrilateral over the isotropic space can be decomposed into two triangles with no increase in maximum interpolation error for approximating a convex quadratic.*

Proof. We shall use the DT [8] in selecting the diagonal for decomposing the general convex quadrilateral into two triangles. The DT has an interesting property

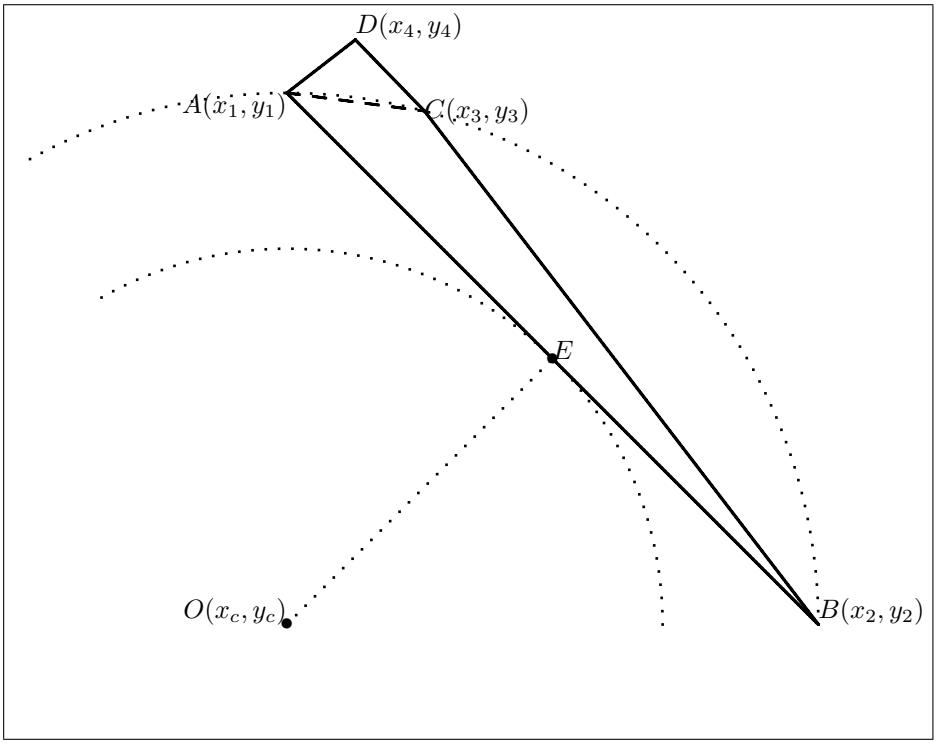


FIG. 2. *Maximum triangulation error attained on boundary edge.*

that three vertices form a triangle in DT iff no other vertex is interior to the circumcircle formed by these vertices. This is also commonly known as the "empty circle" property.

Case 1. The maximum error of the DT is attained at the midpoint (E) of a boundary edge (see Figure 2). In this case the error attained is due to linear interpolation along the edge AB, with value $|AB|^2/8$. Since the isoparametric bilinear interpolant over the quadrilateral also reduces to linear interpolation along the boundary edge, the maximum error for bilinear quadrilateral cannot be less than this value. Therefore the theorem holds.

Case 2. The maximum error of the DT is attained at the center of circumcircle, (x_c, y_c) (see Figure 3). For simplicity and without loss of generality, we perform a translation such that the isotropic quadratic data function is $\frac{1}{2}((x-x_c)^2 + (y-y_c)^2)$. The maximum error is $R^2/2$, where R is the radius of the circumcircle. The interpolation error given by the quadrilateral is (3.3),

$$E_Q(x_c, y_c) = \left(\sum_{i=1}^{i=4} f_i \phi_i(p, q)\right) - f(x_c, y_c), \quad f_i = f(x_i, y_i)$$

$$= \big((1 - \phi_3(p, q))R^2/2 + \phi_3(p, q) f_3\big) - 0 \tag{4.1}$$

since $f_1 = f_2 = f_3 = R^2/2$ and $f(x_c, y_c) = 0$. We have

$$f(x_3, y_3) = ((x_3 - x_c)^2 + (y_3 - y_c)^2)/2 > R^2/2\ , \tag{4.2}$$

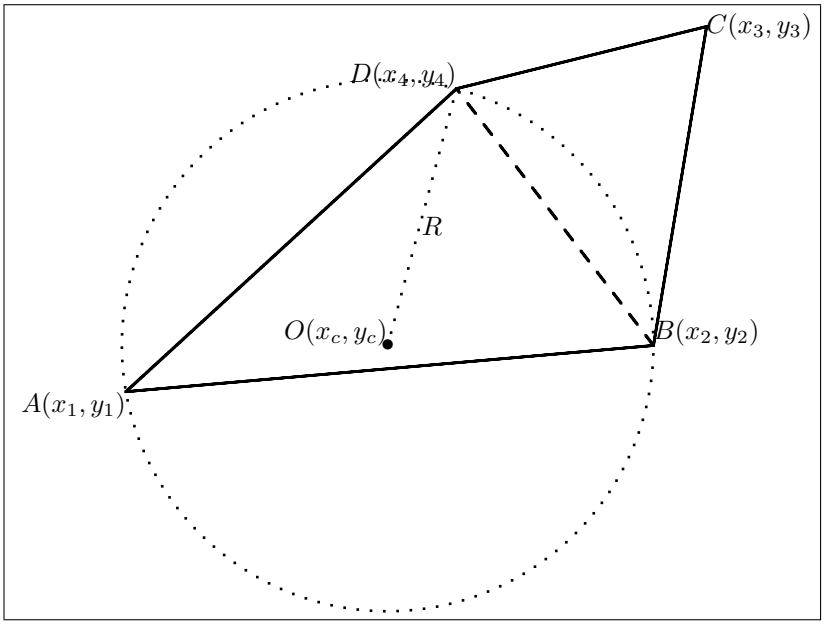


FIG. 3. *Maximum triangulation error attained at center of circum-circle.*

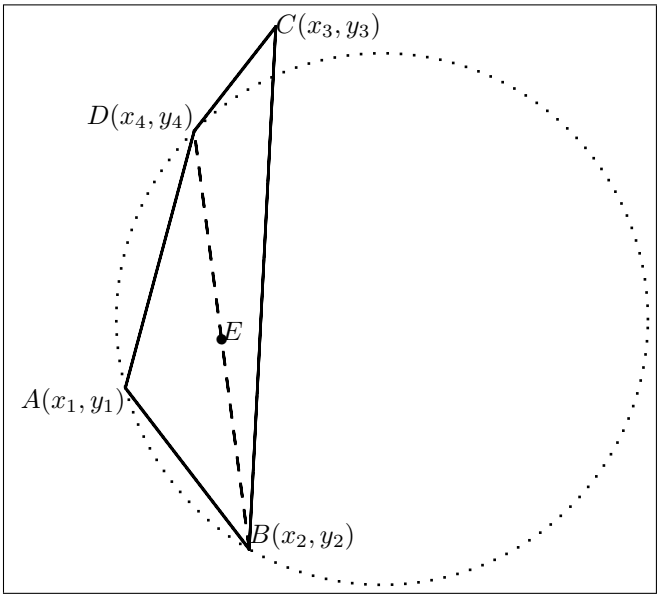


FIG. 4. *Maximum triangulation error cannot be on diagonal.*

and therefore the error attained by quadrilateral at (x_c, y_c) is higher than $R^2/2$, thus the theorem holds.

Cases 1 and 2 are exhaustive since the maximum error of the DT cannot be attained at the the midpoint of a diagonal, unless it also satisfies Case 1 or Case 2 as in a square (see Figure 4). We have $\angle BCD \leq \pi/2$ to satisfy the "empty circle"

property. If $\angle CDB > \pi/2$ (similar argument for $\angle CBD > \pi/2$), then by cosine rule for triangles,

$$|BC|^2 = |CD|^2 + |BD|^2 - 2|CD||BD|\cos(\angle CDB) > |BD|^2 \ , \tag{4.3}$$

thus the maximum error is attained in $\triangle BCD$ on edge BC (Case 1). The remaining alternative is where $\triangle BCD$ forms an acute triangle. Then $\triangle BCD$ will have a larger maximum error given in terms of radius of the circumcircle, which is covered in Case 2.

Therefore over the isotropic space, the DT refined linear triangulation is more accurate than the isoparametric bilinear quadrilateral. □

This theorem suggests if the data function is not saddle shaped, the refine DT triangulation (over the isotropic space) produced above will yield better approximation accuracy, even on arbitrary meshes of general convex quadrilaterals.

4.1. Comparison of efficiency ratio. For the optimal shape equilateral triangle, the area, $\mathcal{A}_T$, is $\sqrt{3}L^2/4$, from (2.4) we obtain an efficiency ratio of

$$\frac{\mathcal{E}_T}{\mathcal{A}_T} = \frac{L^2/6}{\sqrt{3}L^2/4} = 2\sqrt{3}/9 \approx 0.385.$$

Area of the optimal square configuration is L^2, thus the ratio is $1/4 = 0.25$. Hence for an *element-by-element* comparison, the quadrilateral is more efficient. In other words, if we were to approximate a function with either N quadrilaterals or N triangles, quadrilaterals are preferred.

On the other hand, triangles may have advantages over quadrilaterals for finite element computations. Matrix assembly and the solution of the sparse linear equations are commonly the most intensive calculations. If we decompose a quadrilateral mesh into triangles as done above, no extra nodes are introduced. There will be twice as many triangular elements but the resulting assembled matrix has a similar sparsity pattern and the same number of unknowns. Matrix assembly with a *general* convex quadrilateral usually requires costly evaluations of the Jacobian distortion in numerical quadrature over the isoparametric space, whereas assembly of linear triangle elements is simpler. Therefore if computation with N quadrilaterals is as costly as using $2N$ triangles, then triangles are preferred due to their better accuracy and simplicity. The actual computation costs may depend on the implementation of the finite element code.

Consider the approximation of a saddle-shaped function by a square (unrotated) over the isotropic space. The error formula gives

$$\begin{aligned} E_Q(p,q) &= \frac{1}{8}(\mu_1+\mu_2) - \frac{1}{2}(\mu_1(p-p_c)^2 + \mu_2(q-q_c)^2), \quad (p_c,q_c) = \left(\frac{1}{2},\frac{1}{2}\right) \\ &= -\frac{1}{2}((p-p_c)^2L^2 - (q-q_c)^2L^2), \quad \text{where } \mu_1 = L^2 = -\mu_2 \ . \end{aligned} \tag{4.4}$$

The maximum error is attained at the midpoint of each edge. Let $(p,q) = (1,1/2)$, then $\mathcal{E}_Q = L^2/8$. This gives an efficiency ratio of $1/8 = 0.125$. One optimal triangle shape for the saddle-shaped function is the triangle with vertices at $(0,0)$, $(L,0)$, $(1/2L, \sqrt{5}/2L)$ over the isotropic space [7], which has area $\sqrt{5}L^2/4$. The maximum error is $L^2/8$ and attained at the midpoint of each edge. This gives an efficiency ratio of $1/(2\sqrt{5}) \approx 0.224$. Thus over the isotropic space, a mesh with N (unrotated) squares should yield roughly the same accuracy as $2N$ triangles. This is verified in the numerical experiments.

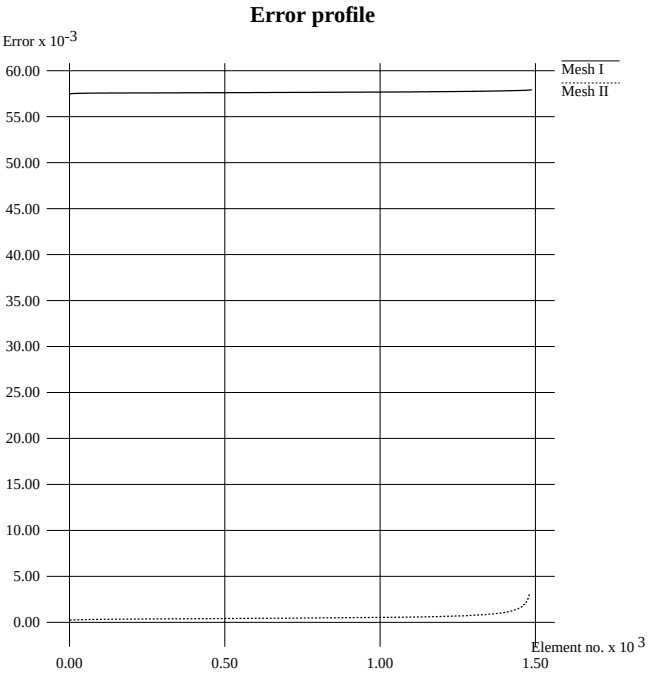


FIG. 5. *Error profiles for Example* 1.

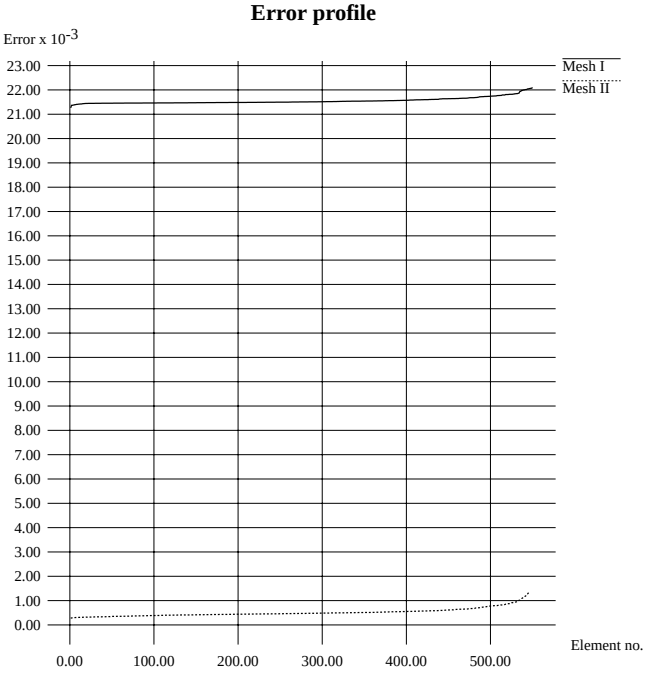


FIG. 6. *Error profiles for Example* 2.

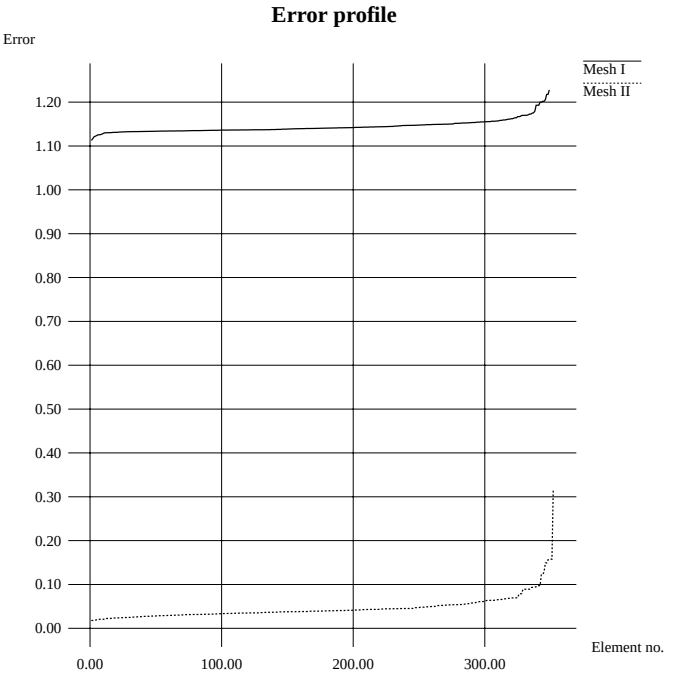


FIG. 7. *Error profiles for Example* 2.

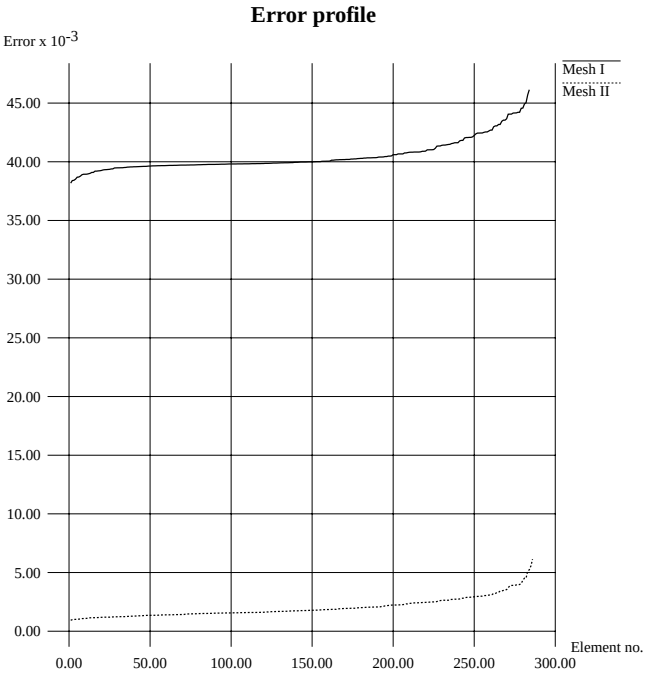


FIG. 8. *Error profiles for Example* 3.

TABLE 1
Summary of results for Example 1.

	Minimum error	Median error	90 percentile	Maximum error	Number of elements
Triangle	5.27E-2	5.39E-2	5.50E-2	5.74E-2	2923
Mesh I	5.75E-2	5.76E-2	5.78E-2	5.79E-2	1488
Mesh II	2.29E-4	4.62E-4	8.30E-4	3.04E-3	1480

5. Numerical experiments. In this section, we demonstrate that a well-designed mesh for bilinear interpolation of a saddle-shaped function may give substantial improvements over a triangular mesh. The examples are taken from [7]. The procedure in [7] for generating optimal triangular meshes is modified to generate optimal quadrilateral meshes. Only elements entirely interior to the unit square are generated to simplify the presentation.

Example 1. Exponential increase along x-axis,

$$f(x,y) = \exp(5x)\sin(5y).$$

Example 2. A near singularity at $(x_0, y_0) = (0.5, 0.2)$,

$$f(x,y) = \frac{(x-x_0)^2 - (y-y_0)^2}{((x-x_0)^2 + (y-y_0)^2)^2}.$$

Example 3. A more severe near singularity,

$$f(x,y) = \frac{((x-x_0)^2 + (y-y_0)^2)^2 - 8(x-x_0)^2(y-y_0)^2}{((x-x_0)^2 + (y-y_0)^2)^4}.$$

Example 4. Example 4 is Example 2 modified by a rescaling of y-axis,

$$f(x,y) = \frac{(x-x_0)^2 - (\sqrt{10}y - y_0)^2}{((x-x_0)^2 + (\sqrt{10}y - y_0)^2)^2}.$$

The results of the experiments are summarized in Figures 5–8 and Tables 1–4. Mesh I is generated by optimal squares over the isotropic space. Mesh II is generated by optimal squares with a $\pi/4$ rotation over the isotropic space to capture the "superconvergence" behavior. Both meshes have similar element size, element shape, and density and differ mainly in their orientation. The meshes are displayed in Figures 9–16. Results for optimal triangular meshes produced in [7] are included for comparison. Mesh I produces an almost level error profile. This indicates an equilibration of interpolation error evenly over all elements. Error profile for Mesh I is roughly comparable to an optimal triangular mesh with about twice as many triangles and in agreement with discussions in section 4. Mesh II displays the "superconvergence" behavior by consistently achieving an error 5–10 times smaller than Mesh I.

It can be shown [6] that the coordinate lines in the isotropic space are mapped to eigentrajectories of the Hessian matrix. Thus as the curved element boundaries are poorly approximated by straight edges, the resulting quadrilateral will no longer have parallel sides (Figures 13, 14). The simple analysis for superconvergence in section 3 for parallelograms may not be adequate and this leads to an anomalous increase in the error displayed in Example 3 of a severe singularity.

TABLE 2
Summary of results for Example 2.

	Minimum error	Median error	90 percentile	Maximum error	Number of elements
Triangle	1.87E-2	2.01E-2	2.16E-2	2.57E-2	1072
Mesh I	2.13E-2	2.15E-2	2.17E-2	2.21E-2	550
Mesh II	2.82E-4	4.69E-4	7.33E-4	1.38E-3	546

TABLE 3
Summary of results for Example 3.

	Minimum error	Median error	90 percentile	Maximum error	Number of elements
Triangle	1.02	1.16	1.32	1.70	650
Mesh I	1.11	1.14	1.16	1.23	349
Mesh II	1.80E-2	3.94E-2	6.75E-2	3.16E-1	352

TABLE 4
Summary of results for Example 4.

	Minimum error	Median error	90 percentile	Maximum error	Number of elements
Triangle	2.91E-2	3.68E-2	4.61E-2	6.46E-2	608
Mesh I	3.81E-2	4.00E-2	4.24E-2	4.61E-2	284
Mesh II	9.36E-4	1.76E-3	3.04E-3	6.13E-3	286

Table 5 shows the effect of generating finer meshes over the isotropic space. If we consider the median error, Mesh I shows the expected $O(h^2)$ convergence. From the efficiency ratio (error/area), we can also predict the decrease of error is proportional to the number of elements. Results for Mesh II clearly display the higher than $O(h^2)$ "superconvergence" behavior. From another perspective, about 5–10 times *more* elements are needed for Mesh I to match the accuracy of Mesh II.

6. Summary. We have used a simple locally quadratic model to develop a geometric interpretation of the interpolation error. We determine the optimal element shapes and their efficiency ratio (error/area) over the isotropic space. The analysis shows for approximating convex data functions, although bilinear quadrilaterals are more efficient, linear triangles are more accurate and may be preferred in finite element computations. For approximating saddle-shaped data functions, a well-designed quadrilateral mesh may show "superconvergence" improvements in approximation accuracy. Numerical experiments show good agreement with the analysis, and a surprising finding is different grid orientations may have an order of magnitude improvement in accuracy.

Appendix A. In this section, we show the error function for quadratic interpolation over a parallelogram is given by (3.4) by simple algebraic expansion. Let the data function be

$$f(x,y) = \frac{1}{2}[x,y]H[x,y]^t + [g_1,g_2][x,y]^t + c \tag{A.1}$$

and the affine isoparametric transformation be

$$\begin{bmatrix} x(p,q) \\ y(p,q) \end{bmatrix} = T \begin{bmatrix} p \\ q \end{bmatrix} + \begin{bmatrix} x_1 \\ y_1 \end{bmatrix},\ T = \begin{bmatrix} u_x & v_x \\ u_y & v_y \end{bmatrix} = \begin{bmatrix} x_2 - x_1 & x_4 - x_1 \\ y_2 - y_1 & y_4 - y_1 \end{bmatrix}. \tag{A.2}$$

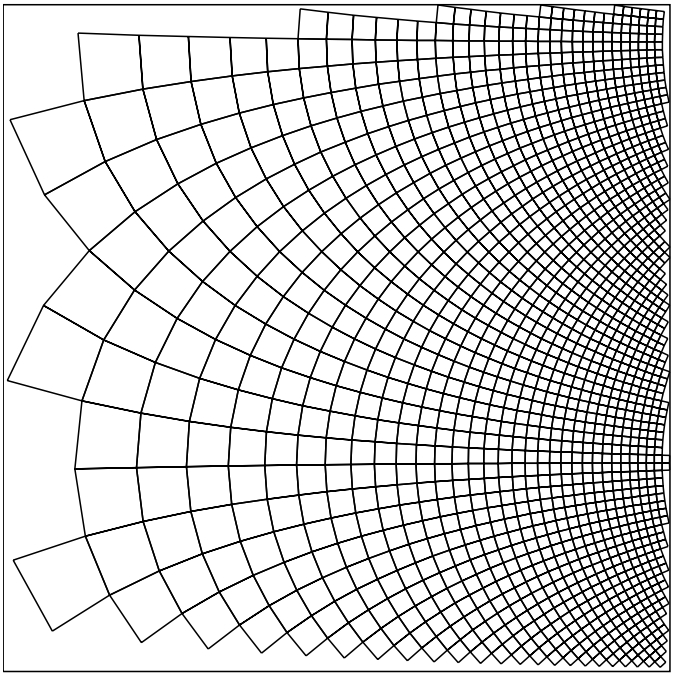


FIG. 9. *Mesh* I *for Example* 1.

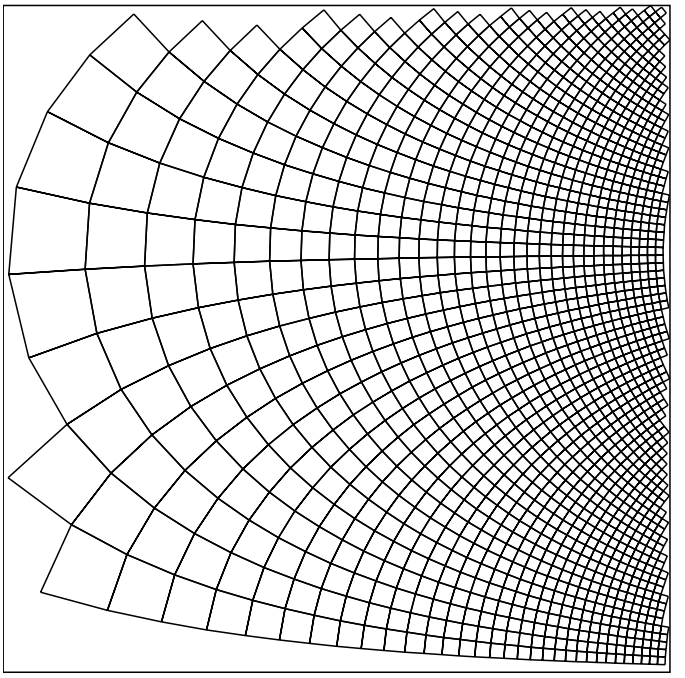


FIG. 10. *Mesh* II *for Example* 1.

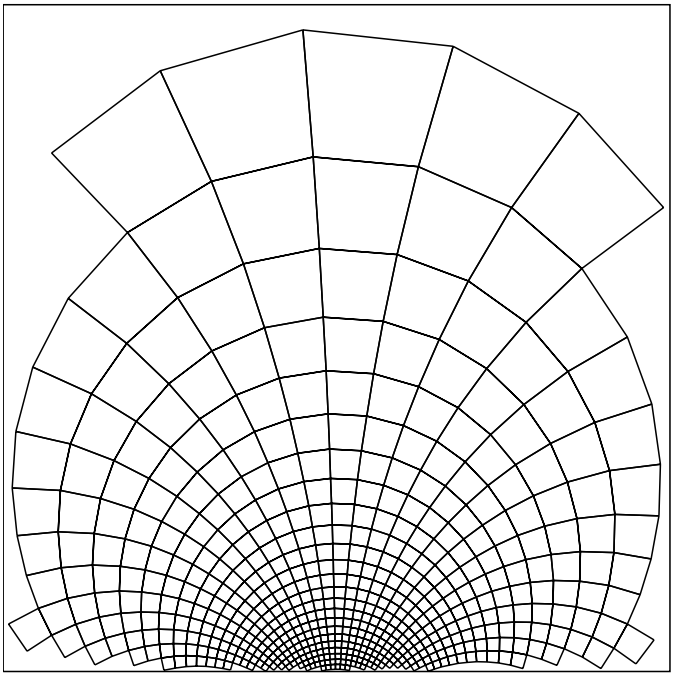


FIG. 11. *Mesh* I *for Example* 2.

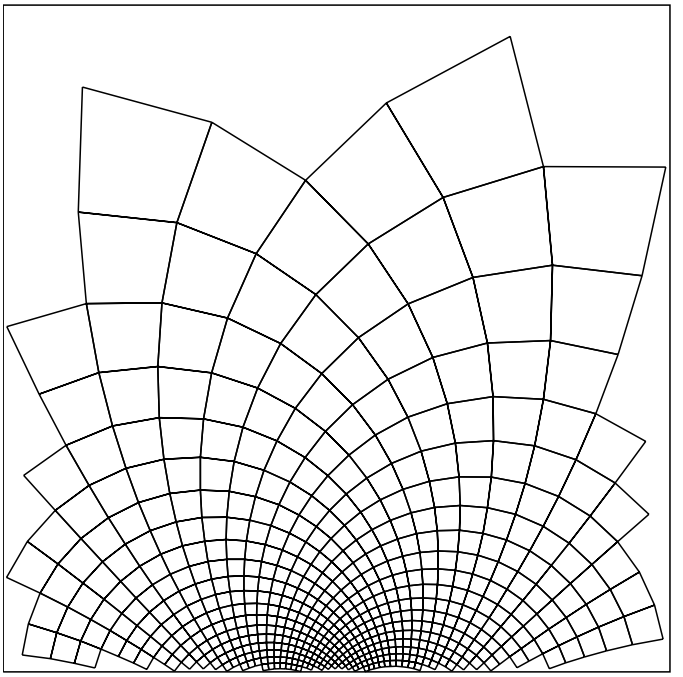


FIG. 12. *Mesh* II *for Example* 2.

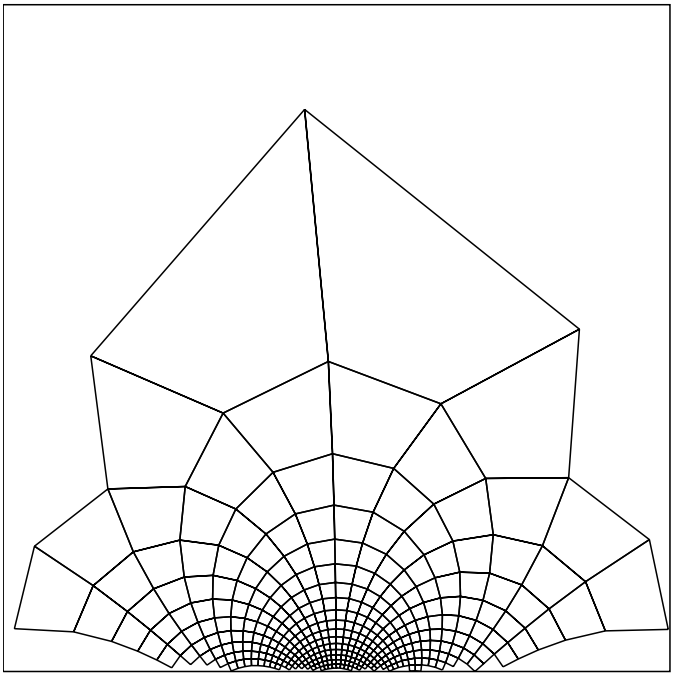


FIG. 13. *Mesh* I *for Example* 3.

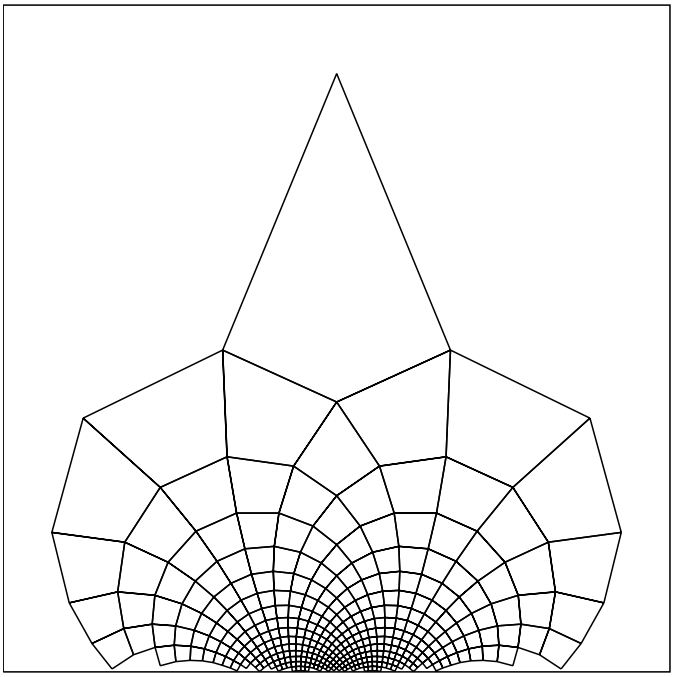


FIG. 14. *Mesh* II *for Example* 3.

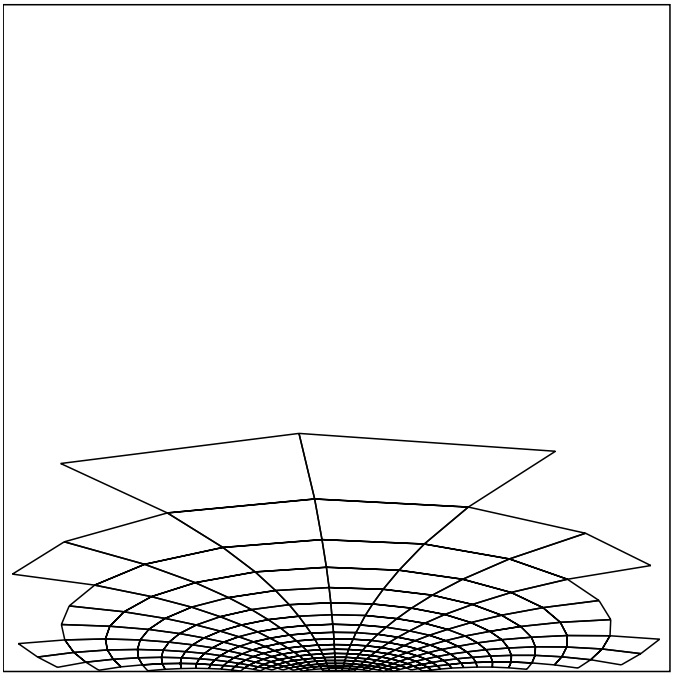


FIG. 15. *Mesh* I *for Example* 4.

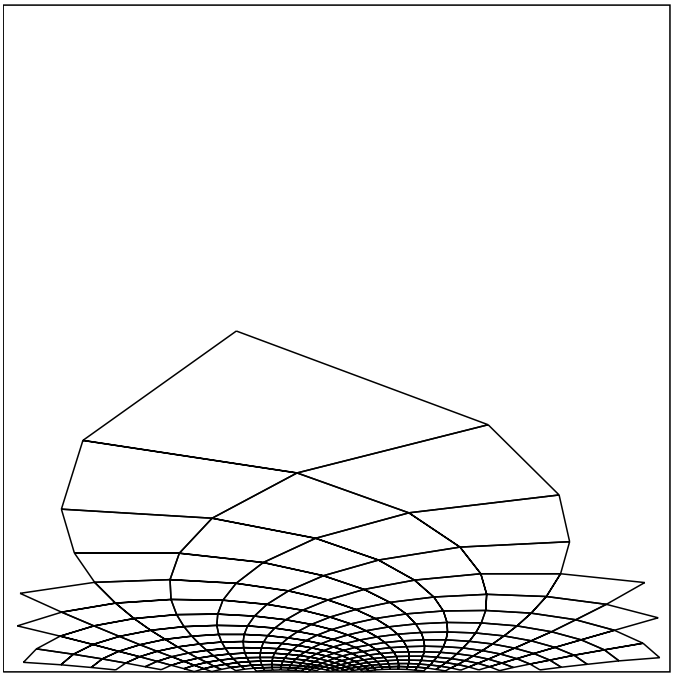


FIG. 16. *Mesh* II *for Example* 4.

TABLE 5
Convergence test on Example 3.

	Minimum error	Median error	90 percentile	Maximum error	Number of elements
Mesh I	11.1E-1	11.4E-1	11.6E-1	12.3E-1	349
Mesh I	3.22E-1	3.23E-1	3.24E-1	3.26E-1	1223
Mesh I	8.03E-2	8.07E-2	8.12E-2	8.23E-2	5063
Mesh I	1.99E-2	2.02E-2	2.04E-2	2.08E-2	20603
Mesh II	1.80E-2	3.94E-2	6.75E-2	3.16E-1	352
Mesh II	2.35E-3	4.22E-3	9.16E-3	6.35E-2	1260
Mesh II	3.10E-4	7.20E-4	1.29E-3	9.41E-3	5244
Mesh II	5.19E-5	1.79E-4	3.78E-4	1.24E-3	21389

Then the interpolation error can be shown to be

$$\begin{aligned} E_Q(p,q) &= p_b(x(p,q),y(p,q)) - f(x(p,q),y(p,q)) \\ &= \mathcal{E}_Q - \frac{1}{2}\left(\mu_1(p-p_c)^2 + \mu_2(q-q_c)^2\right), \end{aligned} \tag{A.3}$$

with centroid at $[p_c, q_c] = [\frac{1}{2}, \frac{1}{2}]$,

$$\begin{aligned} \mathcal{E}_Q &= E_Q(p_c,q_c) = \frac{1}{8}\left(\mu_1+\mu_2\right), \\ \mu_1 &= [u_x,u_y]H[u_x,u_y]^t, \quad \mu_2 = [v_x,v_y]H[v_x,v_y]^t. \end{aligned}$$

Let the data function over (p,q)-space be written as

$$\begin{aligned} \tilde{f}(p,q) &= f(x(p,q),y(p,q)) \\ &= \frac{1}{2}[p,q]\tilde{H}[p,q]^t + [\tilde{g}_1,\tilde{g}_2][p,q]^t + \tilde{c}, \end{aligned}$$

$$\text{where} \quad \tilde{H} = T^t H T = \begin{bmatrix} \tilde{h}_{11} & \tilde{h}_{12} \\ \tilde{h}_{12} & \tilde{h}_{22} \end{bmatrix} \quad \text{and} \tag{A.4}$$

$$\begin{aligned} [\tilde{g}_1,\tilde{g}_2] &= \left([g_1,g_2] + [x_1,y_1]H\right)T\,, \\ \tilde{c} &= c + [g_1,g_2][x_1,y_1]^t + \frac{1}{2}[x_1,y_1]H[x_1,y_1]^t\,. \end{aligned} \tag{A.5}$$

The function values at the four interpolating corners are

$$\begin{aligned} f_1 = \tilde{f}(0,0) = \tilde{c}\,, \quad f_3 = \tilde{f}(1,1) = \frac{1}{2}(\tilde{h}_{11}+\tilde{h}_{22}+2\tilde{h}_{12}) + \tilde{g}_1 + \tilde{g}_2 + \tilde{c}\,, \\ F_2 = \tilde{f}(1,0) = \frac{1}{2}\tilde{h}_{11} + \tilde{g}_1 + \tilde{c}\,, \quad f_4 = \tilde{f}(0,1) = \frac{1}{2}\tilde{h}_{22} + \tilde{g}_2 + \tilde{c}\,. \end{aligned} \tag{A.6}$$

By (3.3) and (A.3) (note the vanishing of linear and constant terms),

$$\begin{aligned} E_Q(p,q) &= \left(\sum_{i=1}^{i=4} f_i\phi_i(p,q)\right) - \tilde{f}(p,q) \\ &= \frac{1}{2}\Big(p(1-q)\tilde{h}_{11} + pq(\tilde{h}_{11}+\tilde{h}_{22}+2\tilde{h}_{12}) \end{aligned}$$

$$
\begin{aligned}
&\quad +(1-p)q\tilde{h}_{22} - (p^2\tilde{h}_{11} + q^2\tilde{h}_{22} + 2pq\tilde{h}_{12})\Big) \\
&= \frac{1}{2}\left(p\tilde{h}_{11} + q\tilde{h}_{22} + 2pq\tilde{h}_{12} - p^2\tilde{h}_{11} - q^2\tilde{h}_{22} - 2pq\tilde{h}_{12}\right) \\
&= \frac{1}{2}\left(p(1-p)\tilde{h}_{11} + q(1-q)\tilde{h}_{22}\right) \\
&= \frac{1}{8}(\tilde{h}_{11} + \tilde{h}_{22}) - \frac{1}{2}\left(\tilde{h}_{11}\left(p - \frac{1}{2}\right)^2 + \tilde{h}_{22}\left(q - \frac{1}{2}\right)^2\right). \qquad \text{(A.7)}
\end{aligned}
$$

From (A.2) and (A.4), we have $\tilde{h}_{11} = \mu_1$ and $\tilde{h}_{22} = \mu_2$; hence the error function has the form given in (A.3).

REFERENCES

[1] D. Ait-Ali-Yahia, W. G. Habashi, A. Tam, M.-G. Vallet, and M. Fortin, *A directionally adaptive methodology using an edge-based error estimate on quadrilateral grids*, Internat. J. Numer. Methods Fluids, 23 (1996), pp. 673–690.

[2] H. Borouchaki and P. J. Frey, *Adaptive triangular-quadrilateral mesh generation*, Internat. J. Numer. Methods Engrg., 41 (1998), pp. 915–934.

[3] H. Borouchaki, P. L. George, and B. Mohammadi, *Delaunay mesh generation governed by metric specifications. Part* I. *Algorithms*, Finite Elem. Anal. Des., 25 (1997), pp. 61–83.

[4] H. Borouchaki, P. L. George, and B. Mohammadi, *Delaunay mesh generation governed by metric specifications. Part* II. *Applications*, Finite Elem. Anal. Des., 25 (1997), pp. 85–109.

[5] F. Bossen, *Anisotropic Mesh Generation with Particles*, Tech. report CMU-CS-96-134, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, 1996; also available from http://ltswww.epfl.ch/ bossen/meshing.html.

[6] E. F. D'Azevedo, *On Optimal Triangulation for Piecewise Linear Approximation*, Ph.D. thesis, Department of Computer Science, University of Waterloo, Waterloo, ON, Canada, 1989.

[7] E. F. D'Azevedo, *Optimal triangular mesh generation by coordinate transformation*, SIAM J. Sci. Statist. Comput., 12 (1991), pp. 755–786.

[8] E. F. D'Azevedo and R. B. Simpson, *On optimal interpolation incidences*, SIAM J. Sci. Statist. Comput., 10 (1989), pp. 1063–1075.

[9] E. F. D'Azevedo and R. B. Simpson, *On optimal triangular meshes for minimizing the gradient error*, Numer. Math., 59 (1991), pp. 321–348.

[10] M. Fortin, M.-G. Vallet, D. Poirier, and W. G. Habashi, *Error Estimation and Directionally Adaptive Meshing*, AIAA paper A94-30728, 25th Fluid Dynamics Conference, Colorado Springs, CO, 1994.

[11] P. L. George, *Improvements on Delaunay-based three-dimensional automatic mesh generator*, Finite Elem. Anal. Des., 25 (1997), pp. 297–317.

[12] P. L. George and F. Hecht, *Nonisotropic grids*, in Handbook of Grid Generation, CRC Press, Boca Raton, FL, 1999.

[13] A. R. Mitchell and R. Wait, *The Finite Element Methods in Partial Differential Equations*, Wiley-Interscience, New York, 1977.

[14] E. Nadler, *Piecewise linear best l_2 approximation on triangulations*, in Approximation Theory V, C. K. Chui, L. L. Schumaker, and J. D. Ward, eds., Academic Press, Boston, 1986, pp. 499–502.

[15] J. Peraire, M. Vahdati, K. Morgan, and O. C. Zienkiewicz, *Adaptive remeshing for compressible flow computations*, J. Comput. Phys., 72 (1987), pp. 449–466.

[16] K. Shimada, *Anisotropic triangular meshing of parametric surfaces via close packing of ellipsoidal bubbles*, in Proceedings 6th International Meshing Roundtable 1997, Park City, UT, 1997; also available as Sandia Report SAND 97-2399 UC-405, Sandia National Laboratories, Albuquerque, NM.

[17] K. Shimada and D. C. Gossard, *Automatic triangular mesh generation of trimmed parametric surfaces for finite element analysis*, Comput. Aided Geom. Design, 15 (1998), pp. 199–222.

[18] R. B. Simpson, *Anisotropic mesh transformations and optimal error control*, Appl. Numer. Math., 14 (1994), pp. 183–198.

[19] I. S. Sokolnikoff, *Tensor Analysis, Theory and Applications to Geometry and Mechanics of Continua*, 2nd ed., John Wiley, New York, 1964.

SIAM J. SCI. COMPUT.
Vol. 22, No. 1, pp. 218–242
© 2000 Society for Industrial and Applied Mathematics

ALTERNATING PLANE SMOOTHERS FOR MULTIBLOCK GRIDS*

IGNACIO M. LLORENTE†, BORIS DISKIN‡, AND N. DUANE MELSON§

Abstract. Standard multigrid methods are not well suited for problems with anisotropic discrete operators, which can occur, for example, on grids that are stretched in order to resolve a boundary layer. One of the most efficient approaches to yield robust methods is the combination of standard coarsening with alternating-direction plane relaxation in the three dimensions. However, this approach may be difficult to implement in codes with multiblock structured grids because there may be no natural definition of global lines or planes. This inherent obstacle limits the range of an implicit smoother to only the portion of the computational domain in the current block. This report studies in detail, both numerically and analytically, the behavior of blockwise plane smoothers in order to provide guidance to engineers who use block-structured grids. The results obtained so far show alternating-direction plane smoothers to be very robust, even on multiblock grids. In common computational fluid dynamics multiblock simulations, where the number of subdomains crossed by the line of a strong anisotropy is low (up to four), textbook multigrid convergence rates can be obtained with a small overlap of cells between neighboring blocks.

Key words. robust multigrid methods, multiblock grids, anisotropic discrete operators

AMS subject classification. 65M55

PII. S106482759935736X

1. Introduction and previous work.

Standard multigrid techniques are efficient methods for solving many types of partial differential equations (PDEs), due to their optimal complexity (the required work is linearly proportional to the number of unknowns) [2], optimal memory requirements, and good efficiency and scalability in parallel implementations [12, 14]. Although highly efficient multigrid methods have been developed for a wide class of problems governed by PDEs, these methods are still underutilized in production and commercial codes [21]. One reason for this underutilization is that the textbook efficiency achieved in solving uniformly elliptic problems is not maintained in anisotropic problems; i.e., convergence rates of standard multigrid methods degrade on problems that have anisotropic discrete operators.

Several methods to deal with anisotropic operators have been studied in the multigrid literature. One popular approach is semicoarsening where the multigrid coarsening process is not applied uniformly to all of the coordinate directions [15, 16, 18, 23]. By selectively *not* coarsening the grid in certain directions, the anisotropy can be reduced on the coarser grids. This process makes the smoothing problem easier because only part of the high-frequency error (namely, the error components oscillating in the directions of coarsening) should be eliminated in relaxation.

Another approach for dealing with anisotropic problems is to develop robust smoothers that can eliminate all the high-frequency errors in the presence of strong

*Received by the editors June 17, 1999; accepted for publication (in revised form) October 22, 1999; published electronically June 13, 2000. This research was supported by the National Aeronautics and Space Administration under NASA contract NAS1-97046 while the first and second authors were in residence at the Institute for Computer Applications in Science and Engineering (ICASE), NASA Langley Research Center, Hampton, VA 23681-2199.

http://www.siam.org/journals/sisc/22-1/35736.html

†Departamento de Arquitectura de Computadores y Automática, Universidad Complutense, 28040 Madrid, Spain (llorente@dacya.ucm.es).

‡Institute for Computer Applications in Science and Engineering, Mail Stop 132C, NASA Langley Research Center, Hampton, VA 23681-2199 (bdiskin@icase.edu).

§Aerodynamic and Aeroacoustic Methods Branch, NASA Langley Research Center, Hampton, VA 23681-2199 (n.d.melson@larc.nasa.gov).

anisotropies [13, 17, 22]. These robust smoothers can be efficiently implemented in the framework of a multigrid method with full coarsening. Plane-implicit relaxation schemes belong to this family of robust smoothers. These schemes are natural for structured grids. (It is more difficult to apply them to unstructured grids because there is no natural definition of a plane or even a line.) Other intermediate alternatives that combine implicit relaxation with partial and full coarsening have been presented in the multigrid literature [6, 7, 9, 10, 19, 25].

Previously [13], we studied the behavior of plane-relaxation methods as multigrid smoothers on single-block grids in three dimensions. With numerical experiments and the local mode analysis, we compared different methods by considering the dependence of their smoothing factors on the anisotropy strength. It was observed that for strong anisotropies and practical grid sizes, there are significant differences in the smoothing factors, depending on whether periodic or Dirichlet boundary conditions are employed. For Dirichlet boundary conditions, increasing anisotropy turns the smoother into an exact solver. (The convergence factor is inversely proportional to the anisotropy strength.)

It has long been known (see, e.g., [2]) that the alternating-direction plane relaxation scheme is a simple, extremely efficient, and robust method. Its efficiency, in fact, is improved in the presence of strong anisotropies. This scheme was considered to be essentially more expensive (in work-count per relaxation sweep) than point relaxation schemes. However, it was found (see [13]) that the overall convergence rate of the three-dimensional (3-D) multigrid solver does not deteriorate if, instead of exactly solving two-dimensional (2-D) problems arising in plane-relaxation sweeps, just one 2-D V-cycle with alternating-line smoother is used in each point. A similar idea of performing plane relaxation with a 2-D multigrid method has been successfully implemented in [9]. With this improvement, the convergence factor per work unit of the 3-D V-cycle employing the alternating-direction plane relaxation scheme for an *isotropic* problem is just twice that of the most efficient cycle using a pointwise smoother. This cost does not look excessive, taking into account the excellent robustness of the alternating-direction plane relaxation smoother in treating anisotropic problems.

Such single-block algorithms are very efficient (especially for structured grids with stretching) due to their relatively easy implementation (on both sequential and parallel computers) and good architectural properties (parallelism and cache memory exploitation). However, multiblock grids are needed to deal with complex geometries or to facilitate parallel processing. This multiblock approach is already useful in serial computers: it improves the data locality properties and efficiently exploits the memory hierarchy of the underlying computer [8].

Plane smoothers have been successfully applied to solve the incompressible Navier–Stokes equations on *rectangular* multiblock grids (multiblock grids where lines and planes are globally defined) [17]. The methods combining semicoarsening with block smoothers (see, e.g., [7]) are also very efficient on such grids. On *general* multiblock grids (multiblock grids without global definitions of planes or lines), however, algorithms based on global semicoarsening are not viable. Moreover, the plane-implicit smoother can be applied only within the current block, and so this plane smoother becomes a blockwise plane smoother. The purpose of this paper is to study whether the optimal properties of plane-implicit smoothers deteriorate for general multiblock grids where these smoothers are not applied globally, but inside each block.

The idea of relaxing not whole lines but portions of lines first appeared in [1]. This issue was also discussed by Jones and Melson in [11], who suggested that multiblock

grids have a detrimental effect on the performance of implicit schemes. By using numerical experiments and rigorous analysis, they derived relations between the block size, the amount of overlap between blocks, and the strength of the anisotropy that must hold for the resulting multigrid algorithm to be efficient. By looking at the model problem, they showed that textbook multigrid efficiency is achieved only when the block sizes (and, therefore, the range of the implicit operators) are proportional to the strength of the anisotropy.

We have developed a flexible 3-D code to study the behavior of blockwise plane smoothers and determine whether the result relating the convergence rate with the block size, the overlap size, and the anisotropy strength obtained for the 2-D case holds for 3-D. The current version of the code solves the nonlinear diffusion-convection equation by using a full multigrid approach with the *full approximation scheme* [2, 24] for V-cycles. This report presents results for the linear anisotropic diffusion equation. This equation can be solved in a multiblock, cell-centered, stretched grid for complex geometries. Each block can overlap with neighboring blocks. The code is implemented in Fortran 77 and has been parallelized with the standard OpenMP directives for shared-memory parallel computing.

The research code is described in section 2. We have analyzed two cases of anisotropy. The first case is an anisotropic equation discretized on uniform grids (section 3) and the second case is an isotropic equation discretized on stretched grids (section 4). Different strategies for parallelizing multigrid solvers with blockwise relaxation schemes are outlined in section 5. Finally, section 6 presents some conclusions and future research directions.

2. The numerical problem. We will study the behavior of plane smoothers on multiblock grids when solving the anisotropic diffusion equation

$$a\frac{\partial^2 u(x,y,z)}{\partial x^2} + b\frac{\partial^2 u(x,y,z)}{\partial y^2} + c\frac{\partial^2 u(x,y,z)}{\partial z^2} = f(x,y,z) \tag{2.1}$$

on a 3-D rectangular open domain Ω, with suitable boundary conditions on $\delta\Omega$, where u is an unknown function and f is a specified source function. Coefficients a, b, and c in (2.1) can, in general, be functions of the spatial variables. A finite volume discretization of (2.1) is applied on a cell-centered computational grid Ω^1, where the explicit form of the function u at the boundary $\delta\Omega$ is used to implement the Dirichlet boundary conditions. The discretization is given by the system of equations $L^1 u^1 = f^1$.

This paper studies two sources of anisotropy: anisotropic equation coefficients and nonunitary cell aspect ratios due to grid stretching. These sources reflect different situations: the former represents problems with a uniform anisotropy throughout the domain, while the latter exhibits problems with an anisotropy that varies from cell to cell. Grid stretching is commonly used in computational fluid dynamics (CFD) grid generation to pack points in regions with large solution gradients while avoiding an excess of points in more benign regions. In the present work, the stretching of the grid in a given direction is characterized by the stretching ratio (quotient between two consecutive mesh sizes in the same direction). See Figure 2.1 for examples of stretched grids. Varying local grid aspect ratios cause the mesh sizes (and, therefore, the anisotropy strength) to be different in each cell. Note that for an exponential stretching in all the directions, every cell can have a different anisotropy.

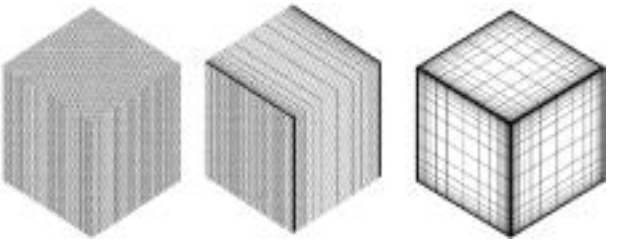


FIG. 2.1. $32 \times 32 \times 32$ *uniform grid,* $32 \times 32 \times 32$ *grid stretched along x-direction* (*stretching ratio* 1.5)*, and* $32 \times 32 \times 32$ *grid stretched along all directions* (*stretching ratio* 1.5).

2.1. The single-block algorithm. Our research code implements the *full approximation scheme* (FAS) to deal with nonlinearity. However, in the simplified (linear problem) case studied in this report the FAS performance is exactly the same as for the *correction scheme.*

A sequence of nonnested grids $\Omega^l (l = 1, \ldots, M)$ is used in the multigrid scheme where Ω^1 is the finest target grid and the rest of the grids are obtained by applying cell-centered coarsening. The coarse grids are uniform grids covering the target domain with one-cell-width margins. The finite volume discretization is defined on all the grids. Dirichlet boundary conditions are implemented by linear interpolation between two neighboring (marginal and inner) cell centers to the real boundary location. In nonstretched-grid solvers, $h_l = 2h_{l-1}$ and $h_1 = h$. $L^l u^l = f^l$ is the discretization of (2.1) on Ω^l. The following iterative algorithm represents a V(γ_1, γ_2)-cycle to solve the system $L^1 u^1 = f^1$.

ALGORITHM 1.

step 1: Apply γ_1 sweeps of the smoothing method to $L^1 u^1 = f^1$

RESTRICTION PART

for $l = 2$ to M

step 2: Compute the residual $r^{l-1} = f^{l-1} - L^{l-1} u^{l-1}$

step 3: Restrict the residual $r^l = I_l^{l-1} r^{l-1}$

step 4: Restrict the current approximation $v^l = I_l^{l-1} u^{l-1}$

step 5: Compute the right-hand side function $f^l = r^l + L^l v^l$

If $(l < M)$, then

step 6: Apply γ_1 sweeps of the smoothing method to $L^l u^l = f^l$

else

step 7: Solve the problem $L^M u^M = f^M$ on the coarsest grid

PROLONGATION PART

for $l = M - 1$ to 1

step 8: Correct the current approximation $u^l = u^l + I_{l+1}^l (u^{l+1} - v^{l+1})$

step 9: Apply γ_2 sweeps of the smoothing method to $L^l u^l = f^l$.

The intergrid operators that perform the restriction (I_l^{l-1} at Steps 3 and 4) and the prolongation (I_{l+1}^l at Step 8) connect the grid levels. The prolongation operator maps data from the coarser level to the current one while the restriction operators transfer values from the finer level to the current one. We used volume weighted summation for the restriction operator and trilinear interpolation in the computational

FIG. 2.2. *Alternating-direction plane-implicit smoother.*

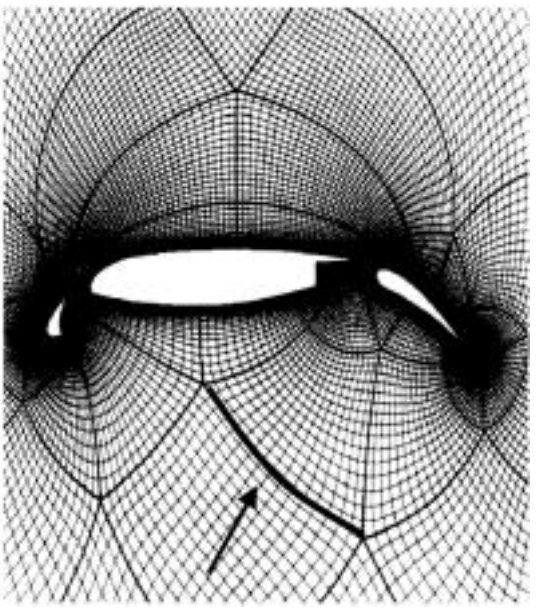


FIG. 2.3. *Multiblock grid for a multielement airfoil. Neighboring subgrids share a grid plane.*

space was used for the prolongation operator. (See [24] for detailed definitions of these intergrid operators.)

Alternating-direction plane smoothers (see Figure 2.2 for Steps 1, 6, and 9), in combination with full coarsening, have been found to be highly efficient and robust for anisotropic discrete operators on single-block grids [13].

- *Optimal work per cycle.* An exact solution of each plane is not necessary; an approximate solution obtained by a single, 2-D multigrid cycle gives the same convergence rate of the 3-D multigrid cycle as an exact solution of each plane, but in much less execution time.
- *Very low convergence factor.* The convergence rate improves as the anisotropy becomes stronger.

2.2. Multiblock grids. Multiblock grids divide domain Ω into P subdomains Ω_p $(p = 1, \ldots, P)$

$$\Omega = \cup_{p=1}^{P} \Omega_p, \tag{2.2}$$

where each subdomain Ω_p is covered with a structured grid Ω_p^1. The grid blocking is generated by the grid generation software to deal with geometric complexities (Fig-

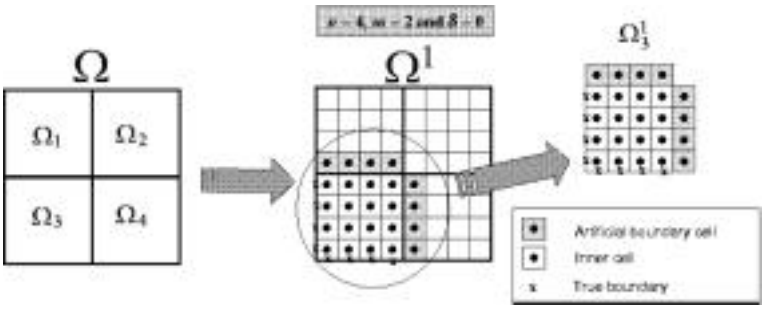


FIG. 2.4. *Data structure of a subgrid without overlap: n is the number of cells per side in every block, m is the number of blocks per side in the split, and δ is the number of overlapping cell planes.*

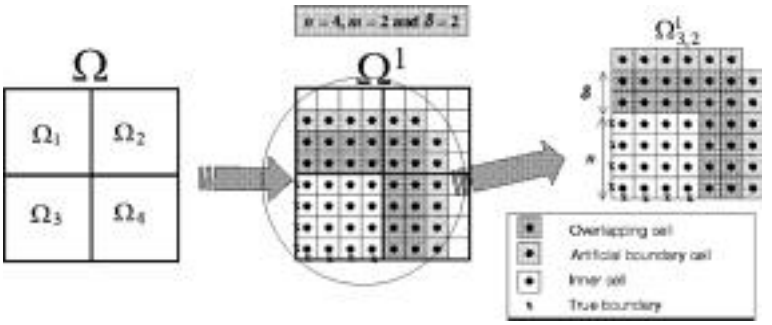


FIG. 2.5. *Data structure of a subgrid with overlap: n is the number of cells per side in every block, m is the number of blocks per side in the split, and δ is the number of overlapping cell planes.*

ure 2.3). A similar blocking is induced on the coarser grid levels. For multigrid, we construct a sequence of grids Ω^l $(l = 1, \ldots, M)$; each grid is defined as the union of P blocks:

$$\Omega^l = \cup_{p=1}^{P} \Omega_p^l; \ (l = 1, \ldots, M). \tag{2.3}$$

We assume that the grid lines are contiguous (possessing C^0 continuity) across the block interfaces (Figure 2.3).

This decomposition generates artificial boundaries within the original domain. It is necessary to apply some boundary conditions to the governing equations at these interfaces. In practice, a Dirichlet boundary condition is applied at artificial boundary cells (Figure 2.4). (Artificial boundary cells are cells adjacent to the boundary of the given block.) The values of the unknowns assigned to these boundary cells are derived from the values at the corresponding cells in the neighboring blocks. Therefore, subgrids are extended to include these artificial boundary cells.

2.3. Overlapping subdomains. Let us define the extended subgrid $\Omega_{p,\delta}^l$ which is the subgrid Ω_p^l including all the inner and artificial boundary cells, plus an external margin of δ-cells deep overlapping into the neighboring block (Figure 2.5). In the general case of a cubic (3-D) partitioning when the overlap can be different in each of the six directions, δ is a six-component vector $(\delta_x^p, \delta_x^m, \delta_y^p, \delta_y^m, \delta_z^p, \delta_z^m)$. The expressions $L_p^l u_p^l = f_p^l$ and $L_{p,\delta}^l u_{p,\delta}^l = f_{p,\delta}^l$ denote discretizations on Ω_p^l and $\Omega_{p,\delta}^l$, respectively.

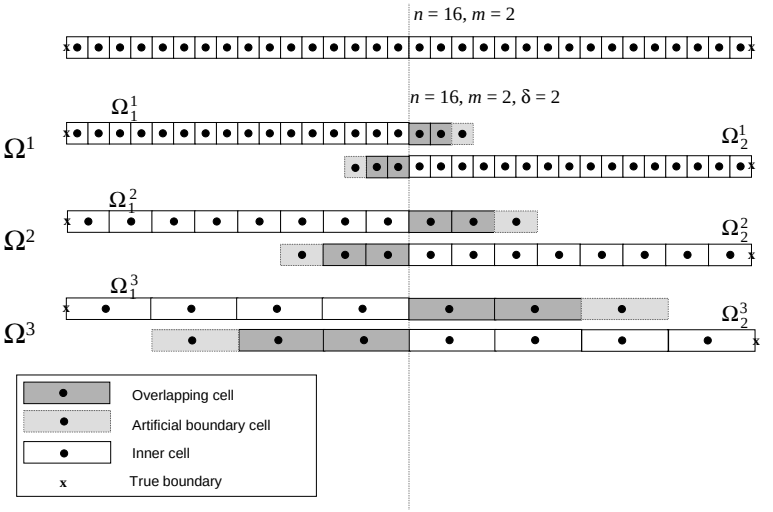


FIG. 2.6. *Grid hierarchy of overlapping subdomains with a fixed number of overlapping cells: n is the number of cells per side in every block, m is the number of blocks per side in the split, and δ is the number of overlapping cell planes.*

For simplicity, we consider a uniform overlapping (the same δ) throughout all the block interfaces. We must distinguish between the number of overlapping cells (2δ) and the physical size of the overlap ($(2\delta + 1)h$). The latter is defined as the distance between centers of the artificial boundary cells of the two extended subgrids sharing a block interface. There are two strategies for the coarse-level block overlapping:

- *Fixed overlap.* The overlap parameter δ is the same in all the levels of the multigrid hierarchy (Figure 2.6). Consequently, the overlapping space increases on the coarse levels. (It is doubled in each coarsening step.) This expansion happens because the number of the overlapping cells in the cross-interface direction remains fixed, while the size of each cell is doubled.
- *Decreasing overlap.* The number of overlapping cells in the cross-interface direction is reduced (δ is divided by 2 for each coarser level; see Figure 2.7). The overlapping space may also increase but much more slowly than in the previous case.

Again, in order to compare these strategies, we have to consider the performance (i.e., convergence rate per work unit) of the obtained algorithms. The work-count per cycle is a bit higher in the algorithm with a fixed overlap, due to the memory and computation overhead on the coarse levels. However, these two alternatives do not exhibit the same convergence rate (see discussion in section 3). Much of the efficiency of the multigrid algorithm is lost when using a decreasing overlap. Consequently, the fixed-overlap strategy was found to be globally more efficient.

The grid generation code generates a multiblock grid decomposed into subgrids in which grid lines are continuous through the block boundaries. The coarsening procedure is performed simultaneously on all the blocks. We assume that the block boundaries are visible on all the coarse grids (except possibly the coarsest grid covering the whole domain). This implies that the grid lines remain continuous on all the coarse levels. Extended subgrids are defined on all the levels at the beginning of the simulation code. The margin cells are constructed to coincide with the corresponding

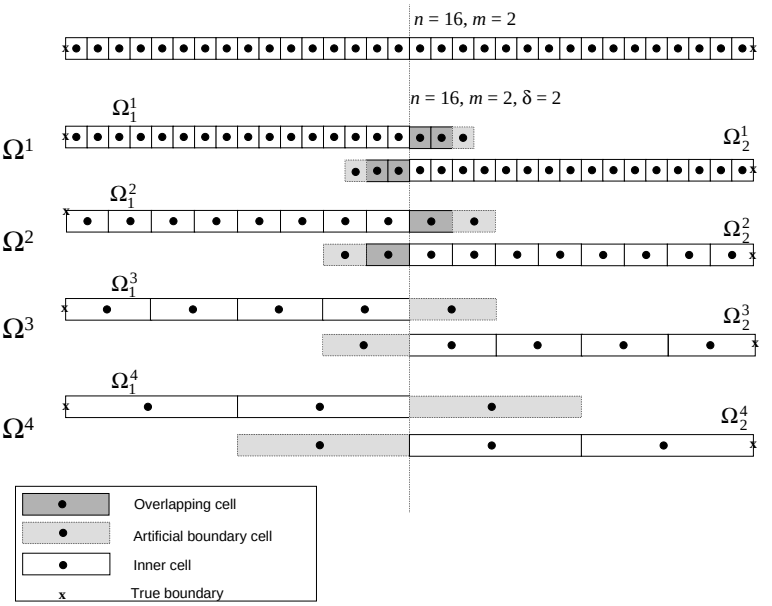


FIG. 2.7. *Grid hierarchy of overlapping subdomains with a decreasing number of overlapping cells: n is the number of cells per side in every block, m is the number of blocks per side in the split, and δ is the number of overlapping cell planes.*

inner cells in overlap regions. Note that overlapping cells may belong to several (more than 2) subgrids. This inherently implies a deterioration of the efficiency in a parallel setting.

In a general multiblock grid, there is no global definition of a line or a plane and so we have two alternatives to implement a V-cycle with an implicit smoother:

- *Domain decomposition with multigrid* applies the V-cycle inside each block.
- *Multigrid with blockwise plane smoother (multigrid with domain decomposition smoother)* applies the V-cycle on the entire multiblock domain, while the smoothers (Steps 1, 6, and 9) are performed inside each block.

2.4. Domain decomposition technique with multigrid. The following iterative algorithm represents a domain decomposition $\mathrm{V}_{\mathrm{DD}}(\gamma_1, \gamma_2)$-cycle to solve the system $L^1 u^1 = f^1$, where Ω^1 is a multiblock grid decomposed into P subgrids.

ALGORITHM 2.

for $p = 1$ to P

step 1: Apply $\mathrm{V}(\gamma_1, \gamma_2)$-cycle to solve the system $L^1_{p,\delta} u^1_{p,\delta} = f^1_{p,\delta}$ on $\Omega^1_{p,\delta}$

update 1: Update the overlap cells of neighboring blocks with values obtained in step 1.

This technique is widely known by the computational community; it is the one-level multiplicative approach to solve the problem in a domain decomposition manner [20]. This alternative does not present a high level of parallelism because blocks are updated in sequential order. Other alternatives (Jacobi or colored ordering) with a higher parallelism can be found in the domain decomposition literature. The main drawback of the one-level multiplicative approach is its slow convergence rate. The convergence of this technique cannot be better than the convergence of the Schwartz

method, which is known to be very poor. In fact, the convergence of a multigrid cycle with blockwise plane smoother will be bounded above by the convergence rate of the domain decomposition solver.

2.5. Multiblock with blockwise smoother. This approach is a priori more efficient because the multigrid algorithm is applied on the whole domain. Its efficiency and excellent parallelization potential already have been demonstrated for *isotropic* elliptic problems [4] where pointwise red-black smoothers were used. In the proposed multigrid method for multiblock grids, the plane smoothers are applied within each block because no global definition of plane or line is assumed. However, the rest of the operators (restriction and prolongation) are completely local and do not need any global information. This method is a simple modification of the one-block approach, with the smoother applied blockwise. The following iterative algorithm represents a V_{BW}-cycle with the blockwise smoother to solve the system $L^1 u^1 = f^1$, where Ω^1 is a multiblock grid.

ALGORITHM 3.

Apply γ_1 sweeps of the blockwise smoother:
  for $p = 1$ to P
    step 1: Apply the smoothing method to the system $L^1_{p,\delta} u^1_{p,\delta} = f^1_{p,\delta}$
    update 1: Update the overlap cells of neighboring blocks with $u^1_{p,\delta}$

RESTRICTION PART

for $l = 2$ to M
  for $p = 1$ to P
    step 2: Compute the residual $r^{l-1}_p = f^{l-1}_p - L^{l-1}_p u^{l-1}_p$
    step 3: Restrict the residual $r^l_p = (I^l_{l-1}) r^{l-1}_p$
    step 4: Restrict the current approximation $v^l_p = (I^l_{l-1}) u^{l-1}_p$
  for $p = 1$ to P
    update 2: Update $v^l_{p,\delta}$ and $r^l_{p,\delta}$ in the external margin cells by using v^l_p and r^l_p of neighboring blocks
  for $p = 1$ to P
    step 5: Compute the right-hand side $f^l_{p,\delta} = r^l_{p,\delta} + L^l_{p,\delta} v^l_{p,\delta}$
  If $(l < M)$, then
    Apply γ_1 sweeps of
      for $p = 1$ to P
        step 6: Apply the smoothing method to the system $L^l_{p,\delta} u^l_{p,\delta} = f^l_{p,\delta}$
        update 3: Update the overlap cells of neighboring blocks with $u^l_{p,\delta}$
  else
    step 7: Solve the coarsest-grid problem $L^l u^l = f^l$ on the whole domain
    for $p = 1$ to P
      update 4: Update $u^l_{p,\delta}$ in the external margin cells by using u^l_p of neighboring blocks

PROLONGATION PART

for $l = M - 1$ to 1
  for $p = 1$ to P
    step 8: Correct the current approximation $u^l_p = u^l_p - (I^l_{l+1})(u^{l+1}_p - v^{l+1}_p)$
  for $p = 1$ to P
    update 5: Update $u^l_{p,\delta}$ in the external margin cells by using u^l_p of neighboring blocks.
  Apply γ_2 sweeps of
    for $p = 1$ to P

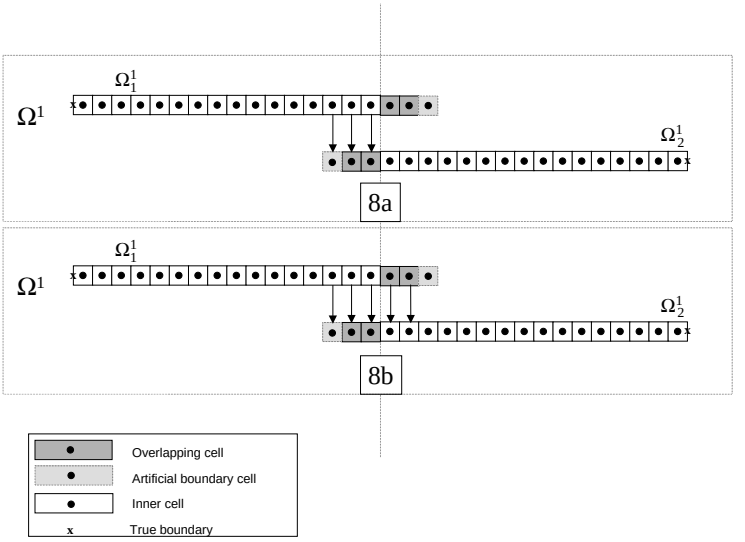


FIG. 2.8. *The algorithm presents two different update processes.*

step 9: Apply the smoothing method to the system $L^l_{p,\delta} u^l_{p,\delta} = f^l_{p,\delta}$
update 6: Update the overlap cells of neighboring blocks using $u^l_{p,\delta}$.

At Step 7, the problem is solved exactly on the coarsest grid common to all the blocks. The cost of solving the system on this grid is negligible compared with the total work.

At updates 2, 4, and 5 the external margin of overlapping cells of neighboring blocks are updated by using inner cells of the current block (Figure 2.8a). At updates 1, 3, and 6 the overlapping cells of neighboring blocks are updated by using inner and external overlapping cells of the current block (Figure 2.8b).

The algorithm includes two computation-update processes:

- Global computation of the residual, restriction, and prolongation (Steps 2, 3, 4, and 8) are performed on inner cells of Ω^l_p for all the blocks. Then the external margin of overlap cells of all the blocks are simultaneously updated (updates 2, 4, and 5).
- Within a smoothing step (Steps 1, 6, and 9), the update of the solution values at the internal and external overlap cells of neighboring blocks (updates 1, 3, and 6) is performed immediately after completing the smoothing process at the current block $\Omega^l_{p,\delta}$. The new, smoothed $u^l_{p,\delta}$ approximation is used. Thus, the order of these updates repeats the order in which the blocks are treated in the smoothing steps.

Next, we study the properties of the plane-smoother technique for multiblock grids in two situations:

- anisotropic equation discretized on uniform grids;
- isotropic equation discretized on stretched grids.

3. Anisotropic equation discretized on uniform grids. A cell-centered discretization of (2.1) on a uniform grid is given by

$$
\begin{aligned}
Lu_{i_x,i_y,i_z} \quad &\equiv \frac{\epsilon_1}{h_x^2}(u_{i_x-1,i_y,i_z} - 2u_{i_x,i_y,i_z} + u_{i_x+1,i_y,i_z}) \\
&+ \frac{\epsilon_2}{h_y^2}(u_{i_x,i_y-1,i_z} - 2u_{i_x,i_y,i_z} + u_{i_x,i_y+1,i_z}) \\
&+ \frac{1}{h_z^2}(u_{i_x,i_y,i_z-1} - 2u_{i_x,i_y,i_z} + u_{i_x,i_y,i_z+1}) = f'_{i_x,i_y,i_z},
\end{aligned} \tag{3.1}
$$

where (i_x, i_y, i_z) are coordinates of cell centers, $i_x = 1, \ldots, n_x, i_y = 1, \ldots, n_y, i_z = 1, \ldots, n_z$, $\epsilon_1 = a/c$, and $\epsilon_2 = b/c$ are the anisotropy coefficients; f' is a discretization of the continuous function $f(x, y, z)/c$; and h_x, h_y, and h_z are the mesh sizes in the x, y, and z directions, respectively. The equation is cell centered and the grid nodes are at the cell vertices. The grid may have different mesh sizes in each dimension.

A multigrid method separates the treatment of different error components: the high-frequency components of the error are reduced in relaxation (smoothing) steps and the low-frequency error components are eliminated in the coarse-grid correction. Thus, the efficiency of a multigrid solver depends strongly on the smoothing factor of the relaxation. Indeed, the derivation of a good smoother is probably the most important step when developing multigrid algorithms.

Consequently, the efficiency of our multigrid technique is determined by the smoothing properties of the blockwise plane smoother. Of course, the smoothing factor of the blockwise plane smoother approaches the factor of a pointwise smoother when the number of blocks increases. In the limit, each block consists of one cell only and the blockwise smoother reverts to a pointwise smoother. This is not a realistic case. We always assume that the number of cells per block is large. Our goal is to study the behavior of the blockwise smoothers when the number of subdomains crossed by the line of a strong anisotropy is low (up to four). This is common in CFD simulations. It is unlikely that regions of very strong anisotropies will traverse many blocks, especially for stretched grids. In such cases, textbook convergence rates could be obtained by allowing larger overlaps in the blocks with the strong anisotropy, while in other blocks the overlaps can be small. Even when an extremely strong anisotropy traverses four blocks in a line, we obtain good convergence rates when using a moderate overlap.

The smoothing properties of a blockwise plane smoother are analyzed for the model (3.1) for an anisotropy (in the equation coefficients) ranging from 1 to 10^6. The number of subdomains crossed by the strong anisotropy is either two or four. As a point of reference, we take the convergence rate obtained by a V(1,0)-cycle for the isotropic problem on a single-block grid with a Gauss–Seidel plane smoother. This cycle demonstrates a textbook multigrid convergence rate of about 0.36 (the plane-smoother convergence of reference) [13]. The convergence rate of the same V(1,0)-cycle with the pointwise Gauss–Seidel smoother is about 0.55 (the point-smoother convergence of reference).

3.1. Numerical results. Let us assume an N^3 global grid partitioned in m^3 blocks (cubic partitioning). Each block is covered with $\Omega^l_{p,\delta}$ extended subgrid. Our aim is to obtain the dependence of the convergence rate $\rho(m, n, \delta, \epsilon)$ on the following parameters:

- number of blocks per side, m;
- number of cells per block side, n $(n = \frac{N}{m})$;
- overlap parameter describing intersection between neighboring blocks, δ;
- strength of the anisotropy, ϵ.

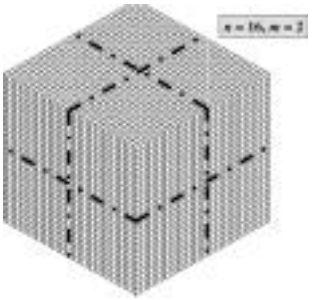


FIG. 3.1. *Rectangular uniform multiblock grid used in the numerical experiments: n is the number of cells per side in every block and m is the number of blocks per side in the split.*

3.1.1. Description of test problem. All the numerical experiments reported here deal with the numerical solution of (2.1) on the unit cube $\Omega = (0,1)\times(0,1)\times(0,1)$ with the right-hand side of

$$f(x,y,z) = -(a+b+c)\sin(x+y+z) \tag{3.2}$$

discretized on 64^3 and 128^3 multiblock grids (Figure 3.1) with different overlaps ($\delta = 0, 2, 4,$ and 8). Dirichlet boundary conditions are enforced on the boundary by evaluating

$$u(x,y,z) = \sin(x+y+z). \tag{3.3}$$

The asymptotic convergence rate monitored in the numerical tests is the asymptotic rate of reduction of the L_2 norm of the residual function in one V(1,1)-cycle.

The (x,y)-plane Gauss–Seidel smoother is applied inside each block and the blocks are updated in lexicographic order. For intergrid transfers, the same operators as defined in section 2.1 are used. The 2-D problems defined in each plane are solved by one 2-D V(1,1)-cycle with x-line Gauss–Seidel smoothing.

3.1.2. Discussion of results. Two sets of calculations were performed to determine the experimental asymptotic convergence rate as a function of the anisotropies. In the first set, two parameters (ϵ_1 and ϵ_2) are varied, and in the second, just one parameter (ϵ_1) is varied while the second parameter (ϵ_2) remains equal to 1.

- Both ϵ_1 and ϵ_2 are varied for the single block case ($m = 1$) and two multiblock cases ($m = 2$ and $m = 4$). Different overlaps between blocks ($\delta = 0$, 1, and 2) are examined. The upper graphs in Figures 3.2 and 3.3 show the results for 64^3 ($N = 64$) and 128^3 ($N = 128$) grids, respectively.
- Only the parameter ϵ_1 is varied for the single block case ($m = 1$), and two multiblock cases ($m = 2$ and $m = 4$). Different overlaps between blocks ($\delta = 0$, 1, and 2) are examined. The lower graphs in Figures 3.2 and 3.3 show these results for 64^3 ($N = 64$) and 128^3 ($N = 128$) grids, respectively.

All the graphs exhibit a similar behavior with respect to δ and ϵ. We can distinguish three different cases:

- In the single-block case ($m = 1$), the convergence rate decreases quickly for an anisotropy larger than 100, tending to (nearly) zero for very strong anisotropies. In fact, the convergence rate per V(1,1)-cycle decreases quadratically with increasing anisotropy strength, as was predicted in earlier work [13].

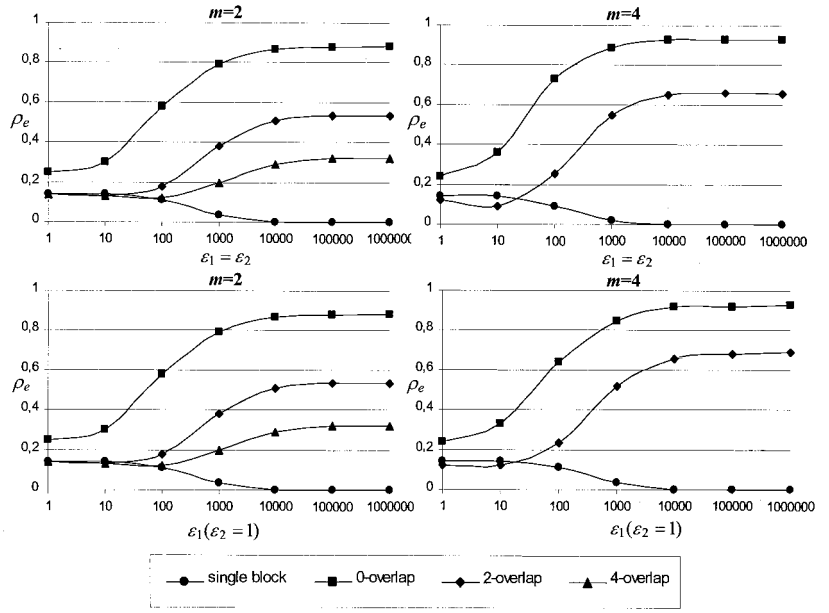


FIG. 3.2. *Experimental convergence factors, ρ_e, of one 3-D V (1, 1)-cycle with blockwise (x, y)-plane smoother in a 64^3 grid with respect to the anisotropy strength (ϵ_1 and ϵ_2) for different values of the overlap (δ) and the number of blocks per side (m).*

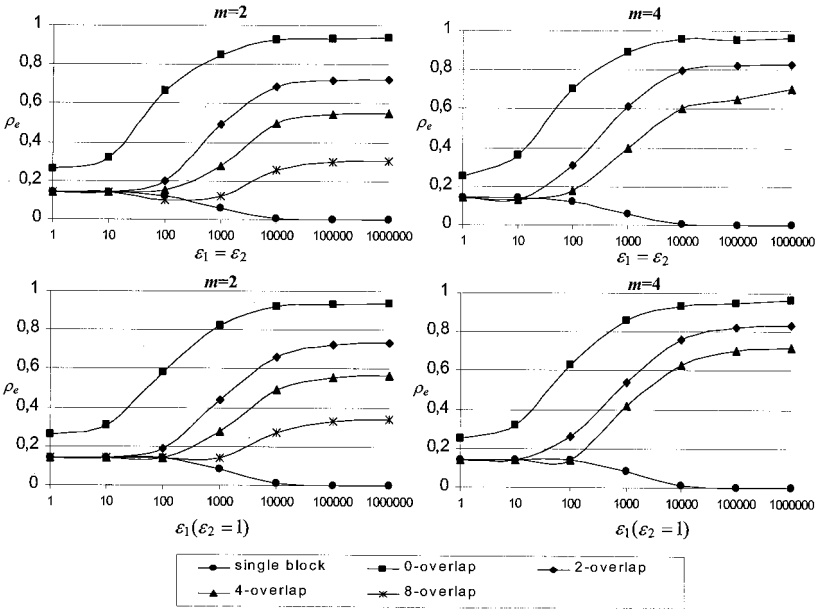


FIG. 3.3. *Experimental convergence factors, ρ_e, of one 3-D V(1, 1)-cycle with blockwise (x, y)-plane smoother in a 128^3 grid with respect to the anisotropy strength (ϵ_1 and ϵ_2) for different values of the overlap (δ) and the number of blocks per side (m).*

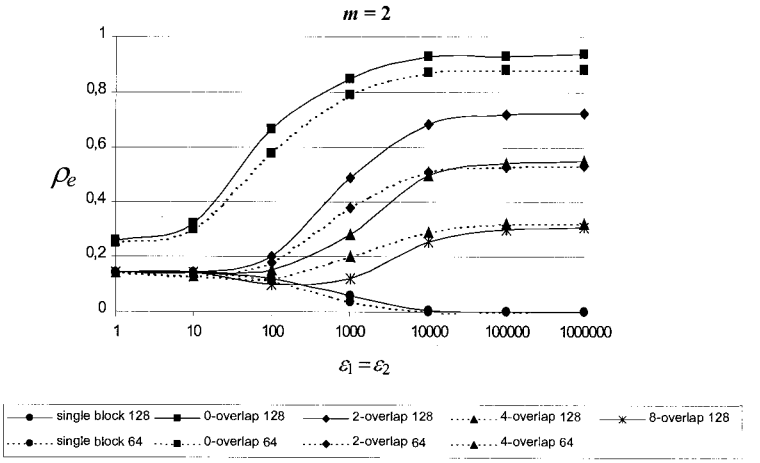


FIG. 3.4. *Experimental convergence factors, ρ_e, of one 3-D $V(1,1)$-cycle with blockwise (x,y)-plane smoother in 128^3 and 64^3 grids with respect to the anisotropy strength (ϵ_1 and ϵ_2) for different values of the overlap (δ) and 2 blocks per side ($m = 2$).*

- If the domain is blocked with the minimal overlapping ($m > 1$ and $\delta = 0$), the convergence rate for small anisotropies ($1 \leq \epsilon \leq 100$) is similar to that obtained with a single block with a pointwise smoother on the whole domain (i.e., point-smoother convergence of reference, which is about 0.55^2). It increases to a fixed value for larger anisotropies. This limit is defined by the convergence in the corresponding domain decomposition algorithm. The convergence rate for strong anisotropies gets closer to one for finer grids (compare Figures 3.2 and 3.3 for the same m and δ but different N).
- If the domain is blocked with larger overlapping ($m > 1$ and $\delta \geq 2$), the convergence rate for small anisotropies is similar to that obtained without blocking on the whole domain (i.e., plane-smoother convergence of reference which is about 0.36^2) and increases to the fixed domain decomposition limit for very strong anisotropies. The asymptotic value for strong anisotropies gets closer to one for smaller overlaps and finer grids (compare Figures 3.2 and 3.3 for the same m but different δ and N).

Numerical results show the following properties of the convergence behavior:

- Convergence is very poor with the minimal overlap ($\delta = 0$), but it improves rapidly with larger overlaps ($\delta \geq 2$).
- If we compare the results for 64^3 and 128^3 and $m = 2$ (Figures 3.2 and 3.3), we realize that the convergence rate for large anisotropies is bounded if the overlap is increased in proportion to the size of the block, i.e., $\rho(2, n, \delta, \epsilon) \approx \rho(2, 2n, 2\delta, \epsilon)$ (see Figure 3.4). Therefore, mesh-independent convergence rates can be obtained for large anisotropies by increasing the number of overlap cells in proportion to the number of cells per side in the subgrid. The amount of extra work (in relative terms) to maintain the convergence rate at a fixed level does not increase for finer grids.
- For 17-cell overlap ($\delta = 8$), 2 blocks per side ($m = 2$), and 32^3 cells per block ($n = 32$), we obtain for all anisotropies (nearly) the same convergence rate as for the isotropic case without blocking (plane-smoother convergence of reference). This overlap results in only a 25 percent computation and memory

penalty (these results are not shown in the figures).
- The smoothing factor for very strong anisotropies degenerates with an increasing number of blocks m because the blockwise smoother tends to be a pointwise smoother.

3.2. Analysis. The full-space Fourier analysis is known as a simple and very efficient tool to predict convergence rates of multigrid cycles in elliptic problems. However, this analysis is inherently incapable of accounting for boundary conditions. In isotropic elliptic problems where boundary conditions affect just a small neighborhood near the boundary, this shortcoming does not seem to be very serious. The convergence rate of a multigrid cycle is mostly defined by the relaxation smoothing factor in the *interior* of the domain, and therefore predictions of the Fourier analysis prove to be in amazing agreement with the results of numerical calculations. In the presence of a strong anisotropy, boundary conditions have a stronger effect on the overall convergence factor that a multigrid cycle can achieve. If the strong anisotropy direction aligns with the grid (as in (3.1)) and the domain is not blocked, then the smoothing factor calculated by the full-space Fourier analysis still predicts well the multigrid cycle convergence rate. In the case of nonalignment, however, the full-space analysis cannot reflect the poorness of the coarse-grid correction, which slows down the convergence (see [5, 3]) and so the half-space analysis must be used instead. The multiblock structure brings additional difficulties: in strongly anisotropic problems on decomposed domains, artificial boundaries have a large effect on the solution in the interior (e.g., on the relaxation stage), making even the half-space analysis inadequate. In the extreme case of a very large anisotropy, the behavior of a plane smoother is similar to that exhibited by a one-level overlapping Schwartz method [20]. The following analysis is an extension of the half-space Fourier analysis, taking into account the influence of a second boundary.

We will consider the discrete equation (3.1) on a layer $(i_x, i_y, i_z) : 0 \le i_x \le N$, $-\infty < i_y, i_z < \infty$. This domain is decomposed by overlapped subdomains in an x-line partitioning. The boundaries of all the subdomains are given by the set of cell planes orthogonal to the x-coordinate axis. Boundary conditions are represented by one 2-D Fourier component at a time. In this way, the original 3-D problem is translated into a one-dimensional (1-D) problem where frequencies of the Fourier component are considered parameters. When estimating the smoothing factor for the 3-D alternating-direction plane smoother, we analyze only the high-frequency Fourier components. A 2-D Fourier component $e^{i(\theta_y i_y + \theta_z i_z)}$, $(|\theta_y| \le \pi;\ |\theta_z| \le \pi)$ is a high-frequency component if the absolute value of either θ_y or θ_z is greater than or equal to $\pi/2$.

We are given values of $N, m, n, \delta, \epsilon_1$ and ϵ_2 (see definitions in section 3.1), and we let the integers $i_x = 0, \ldots, N$ numerate the cells. We also label the blocks with numbers from 1 to m. To predict the smoothing factors observed in our numerical experiments, we assume that the strongest anisotropy direction is aligned with the x axis, i.e., $\epsilon_1 > \epsilon_2 \ge 1$. The actual boundary separating two neighboring blocks is placed *between* the cell centers. Recall that the overlap parameter δ corresponds to the case where the width of the domain shared by two overlapped blocks is $(2\delta + 1)h$. The left boundary condition of the first block is defined at the center of the actual (not artificial) left boundary cell $l_1 = 0$; the right boundary condition is defined at $r_1 = n + \delta + 1$. Each block k $(k = 2, \ldots, m-1)$ has two (artificial) boundary planes settled in the interior of the domain: the center of the left boundary cell is located at $l_k = (k-1)n - \delta$ and the center of the right boundary cell is at $r_k = kn + \delta + 1$. The left boundary cell of the last mth block is centered at $l_m = (m-1)n - \delta$ and

the center of the right boundary cell is at $r_m = N$. One application of the 3-D alternating-direction plane smoother on the kth block includes three plane-relaxation sweeps. In the analysis, we assume that in the first sweep, the smoother proceeds in the x direction from $i_x = l_k$ to $i_x = r_k$, solving the corresponding y-z-plane problems exactly. In numerical tests, just one V(1,1)-cycle is applied to approximate the 2-D solutions. The second and third sweeps are performed in the y and z directions, respectively.

Let us consider (3.1) with the initial approximation $\tilde{u}_{i_x,i_y,i_z}$ and the source function f'_{i_x,i_y,i_z} given in the form

$$\begin{aligned}
&\tilde{u}_{i_x,i_y,i_z} = A(i_x)e^{i(\theta_y i_y + \theta_z i_z)},\\
&f'_{i_x,i_y,i_z} = F(i_x)e^{i(\theta_y i_y + \theta_z i_z)},\\
&i_x = 0,\dots,N,
\end{aligned}$$

where A and F are $(N+1)$-dimensional complex-valued vectors. $A(i_x)$ and $F(i_x)$ represent the corresponding amplitudes of the Fourier component at i_x.

Let $E(i_x)$ be the amplitude of the algebraic error in the kth block after updating the solution values at the overlap $[l_k,(k-1)n]$. We have $E(i_x) = U(i_x) - A(i_x)$, where $U(i_x)$ is the amplitude of the exact discrete solution

$$\begin{aligned}
&U(0) = U_0, \qquad U(N) = U_1,\\
&\epsilon_1 U(i_x - 1) + \mu_1 U(i_x) + \epsilon_1 U(i_x + 1) = F(i_x),\\
&i_x = 1,\dots,N-1,\\
&\mu_1 = -2\epsilon_1 + 2\epsilon_2\Big(\cos\theta_y - 1\Big) + 2\Big(\cos\theta_z - 1\Big),
\end{aligned}$$

with given boundary conditions U_0 and U_1.

In the first sweep, the new amplitude $E_1(i_x)$ is defined from

$$\begin{aligned}
&E_1(l_k) = E(l_k), \qquad E_1(r_k) = E(r_k),\\
&\epsilon_1 E_1(i_x - 1) + \mu_1 E_1(i_x) = -\epsilon_1 E(i_x + 1),\\
&i_x = l_k + 1,\dots,r_k - 1.
\end{aligned} \tag{3.4}$$

After the second sweep the new amplitude $E_2(i_x)$ is the solution of the following system of equations:

$$\begin{aligned}
&E_2(l_k) = E(l_k), \qquad E_2(r_k) = E(r_k),\\
&\epsilon_1 E_2(i_x - 1) + \mu_2 E_2(i_x) + \epsilon_1 E_2(i_x + 1) = -\epsilon_2 e^{i\theta_y} E_1(i_x),\\
&i_x = l_k + 1,\dots,r_k - 1,\\
&\mu_2 = -2\epsilon_1 + \epsilon_2\Big(e^{-i\theta_y} - 2\Big) + 2\Big(\cos\theta_z - 1\Big).
\end{aligned} \tag{3.5}$$

Finally, the amplitude $E_3(i_x)$ of the approximate solution obtained after the third sweep is found from the following system of equations:

$$\begin{aligned}
&E_3(l_k) = E(l_k), \qquad E_3(r_k) = E(r_k),\\
&\epsilon_1 E_3(i_x - 1) + \mu_3 E_3(i_x) + \epsilon_1 E_3(i_x + 1) = -e^{i\theta_z} E_2(i_x),\\
&i_x = l_k + 1,\dots,r_k - 1,\\
&\mu_3 = -2\epsilon_1 + 2\epsilon_2\Big(\cos\theta_y - 1\Big) + \Big(e^{-i\theta_z} - 2\Big).
\end{aligned} \tag{3.6}$$

The smoothing factor of the alternating-direction plane smoother step is the ratio between the L_2 norms of high-frequency errors before and after the step is performed. This analysis is very rigorous and especially useful for predicting the convergence in full multigrid (FMG) algorithms (see, e.g., [2]). In these algorithms, where only a few smoothing steps are performed on each grid and the initial error is basically the difference between solutions on successive grids, this analysis allows a direct estimation of the algebraic error at any stage of the algorithm.

To estimate asymptotic convergence rates, the following simplifications can be adopted.

3.2.1. Simplifications in different regimes. There are two processes affecting the amplitude of high-frequency error components. The first is the smoothing in the interior of the blocks. The effect of this smoothing is very similar to that on the single-block domain. It is well described by the smoothing factor, SM, derived from the usual full-space Fourier mode analysis (see [2, 13, 24]). If the problem is essentially isotropic ($\epsilon_1 = O(1)$), then each of the three sweeps (3.4)–(3.6) contributes to the alternating-direction plane-relaxation smoothing factor. This smoothing factor proves to be very small ($SM = 1/\sqrt{125} \approx 0.089$ in the pure isotropic problem), but the overall convergence rate in a V-cycle is worse because it is dictated by the coarse-grid correction. To predict the asymptotic convergence rate in a V-cycle, the two-level analysis should be performed. In problems with a moderate anisotropy in the x direction ($\epsilon_1 = O(h^{-1})$), the high-frequency error is reduced mainly in the third sweep (3.6) and the smoothing factor approaches the 1-D factor $SM = 1/\sqrt{5}$ (see [13]). In strongly anisotropic problems ($\epsilon_1 = O(h^{-2})$), the third sweep reduces the smooth error components as well, actually solving the problem rather than just smoothing the error.

The second process influencing the high-frequency error is the error propagation from incorrectly specified values at the block boundaries. The distance on which this high-frequency boundary error penetrates inside blocks strongly depends on the anisotropy. It can be estimated by considering a semi-infinite problem associated with the sweep (3.6). The left-infinite problem stated for the kth block assumes a zero right-hand side and omits the left boundary condition

$$E(r_k) = B, \qquad E_2(i_x) = 0, \qquad -\infty < i_x < r_k.$$

The solution of this problem is

$$E_3(i_x) = B\lambda_l\Big(\theta_y, \theta_z\Big)^{i_x - r_k},$$

where $\lambda_l(\theta_y, \theta_z)$ satisfies

$$\epsilon_1\lambda^{-1} + \mu_3 + \epsilon_1\lambda = 0, \qquad |\lambda| \geq 1. \tag{3.7}$$

For high-frequencies under the assumption $\epsilon_1 > \epsilon_2 \geq 1$,

$$\Big|\lambda_l\Big(\theta_y, \theta_z\Big)\Big| \geq \Lambda_l = \Big|\lambda_l\Big(0, \pi/2\Big)\Big|.$$

The right-infinite problem is similarly solved, providing

$$\Big|\lambda_r\Big(\theta_y, \theta_z\Big)\Big| \leq \Lambda_r = \Big|\lambda_r\Big(0, \pi/2\Big)\Big|,$$

where $\lambda_r(\theta_y, \theta_z)$ is a root of (3.7) satisfying $|\lambda| \leq 1$.

The roots of quadratic equation (3.7) for $\theta_y = 0$ and $\theta_z = \frac{\pi}{2}$ are given by

$$\Lambda_l = \left| -\frac{\hat{\mu}_3}{2} + \sqrt{\frac{\hat{\mu}_3^2}{4} - 1} \right| \quad \text{and}$$

$$\Lambda_r = \Lambda_l^{-1} = \left| -\frac{\hat{\mu}_3}{2} - \sqrt{\frac{\hat{\mu}_3^2}{4} - 1} \right|,$$

where

$$\hat{\mu}_3 = -2 - \frac{(i+2)}{\epsilon_1}.$$

Thus, if

$$\Lambda_r^{r_k - l_k - 2\delta - 1} \ll \Lambda_r^{2\delta+1},$$

i.e., the influence of the far boundary (e.g., r_{k+1} boundary on r_k), is negligible in comparison with the influence of the nearby boundary (l_{k+1} on r_k), then the high-frequency error amplitude reduction factor, RF, is estimated as the ratio between the amplitudes on the right boundary r_k before and after the plane-smoother step is performed. After relaxing the kth block, the amplitude at l_{k+1} is about $B\Lambda_l^{-2\delta-1}$ and after relaxing the $(k+1)$-th block, the estimated amplitude at r_k is $B(\Lambda_r/\Lambda_l)^{2\delta+1} = B(\Lambda_r)^{4\delta+2}$. Thus, the reduction factor could be approximated by

$$RF \approx \left(\Lambda_r\right)^{4\delta+2}. \tag{3.8}$$

If the anisotropy is strong ($\epsilon_1 = O(h^{-2})$), both boundaries (far and nearby) affect the error amplitude reduction factor. If the number of blocks m is not too large, then the corresponding problem includes m coupled problems (like 3.6) with zero right-hand sides. This multiblock problem can directly be solved. For the two-block partition it results in

$$RF = \left(\frac{\Lambda_l^{n-\delta-1} - \Lambda_r^{n-\delta-1}}{\Lambda_l^{n+\delta} - \Lambda_r^{n+\delta}} \right)^2. \tag{3.9}$$

In all the numerical tests described in sections 3.1–3.2, the V(1,1)-cycle convergence rate can be predicted by the (squared) smoothing factor, which is calculated as the maximum value of SM or RF.

3.2.2. Relation to domain decomposition convergence theory. When the anisotropy is very strong and the number of blocks is much greater than two, convergence rates observed in numerical tests are well described by the domain decomposition convergence theory [20]. In this case, we are not interested in the smoothing properties of the 3-D smoother but in its resolution properties because it becomes an exact method to solve N noncoupled 2-D problems (when ϵ_1 and ϵ_2 are very large) or N^2 1-D problems (when only ϵ_1 is large). In fact, asymptotic convergence rates for large anisotropies are in good agreement with convergence rates predicted for 1-D or 2-D domain-decomposition Schwartz methods.

TABLE 3.1

Experimental, ρ_e, and analytical, ρ_a, convergence factors of a single 3-D $V(1,1)$-cycle with blockwise (x,y)-plane Gauss–Seidel smoother ($2 \times 2 \times 2$ partition) versus anisotropy strength, width of the overlap, and the block size.

ϵ_1	δ	$n = 64$		$n = 128$	
		ρ_e	ρ_a	ρ_e	ρ_a
10^6	0	0.882	0.881	0.939	0.939
	2	0.534	0.529	0.731	0.729
	4	0.321	0.316	0.567	0.566
	8			0.341	0.340
10^4	0	0.87	0.87	0.92	0.92
	2	0.51	0.51	0.66	0.67
	4	0.29	0.29	0.49	0.49
	8			0.27	0.26
10^2	0	0.58	0.56	0.58	0.56
	2	0.18	0.14	0.19	0.14
	4	0.12	0.14	0.14	0.14
	8			0.14	0.14

3.2.3. Comparison with numerical tests. For simplicity, we consider the V(1,1)-cycle with only z-planes used in the smoothing step. The assumption that $\epsilon_1 > \epsilon_2 \geq 1$ validates this simplification. The predicted smoothing factor for this relaxation is $SM = 1/\sqrt{5}$ [13]. In numerical calculations for isotropic problems on single-block domains, the asymptotic convergence rate was 0.14 per V(1,1)-cycle, which is very close to the value predicted by the two-level Fourier analysis (0.134).

The next case we present is the comparison of the analytical predictions and the numerical results for the V(1,1)-cycle convergence rates for different anisotropies for the domain decomposed into two blocks in each direction. In this case, for predicting the reduction factor, RF, analytical expression (3.9) can be used. The formula for the asymptotic convergence rate ρ_a is

$$\rho_a = \max\Big(RF^2, 0.14\Big). \tag{3.10}$$

Table 3.1 presents a representative sample of experiments for the case $\epsilon_1 \gg \epsilon_2 = 1$. In this table, ρ_e corresponds to asymptotic convergence rates observed in the numerical experiments, while ρ_a is calculated by means of (3.10). The results demonstrate nearly perfect agreement.

The asymptotic convergence rate deteriorates as the number of blocks grows. For a multiblock partition, an accurate value of RF can still be calculated. (In the case when the domain is partitioned into m blocks, it will be the spectral radius of a $(m-1)$-by-$(m-1)$ matrix.) We decided, however, to present another, simplified, methodology. Here the convergence upper bound, ρ_{DD}, which is the asymptotic convergence rate in the multiblock domain decomposition solver is estimated numerically as $\rho_{DD} = \rho_e$ for $\epsilon_1 = \infty$. The reduction factor RF is calculated from expression (3.8), and the predicted convergence rate is

$$\rho_a = \max\Big(\min(RF^2, \rho_{DD}), 0.14\Big). \tag{3.11}$$

Practically, this formula implies that for small anisotropies ($\epsilon_1 \leq 10$) and for very large anisotropies ($\epsilon_1 \geq 10^4$) the asymptotic convergence rate is estimated by

TABLE 3.2
Moderate anisotropy: experimental, ρ_e, and analytical, ρ_a, convergence factors versus overlap parameter, block size, and number of blocks.

ϵ_1	δ	ρ_a	ρ_e			
			$n=16$	$n=32$		$n=64$
			$m=4$	$m=2$	$m=4$	$m=2$
10^2	0	0.56	0.64	0.58	0.63	0.58
	2	0.14	0.23	0.18	0.26	0.19
10^3	0	0.83	0.85	0.79	0.86	0.82
	2	0.40	0.52	0.38	0.54	0.44
	4	0.19		0.20	0.42	0.28

some constants calculated numerically. The analytical expression is correct in the range of moderate anisotropies. Notice that RF depends only on two parameters (ϵ_1 and δ) which means that estimate (3.11) provides a good prediction only if $n \gg \delta$. Table 3.2 compares this analytical prediction with the results of numerical tests for some moderate anisotropies and different values of the overlap parameter δ.

We can now discuss the different convergence behavior exhibited by fixed and decreasing overlaps. A decreasing overlap introduces errors on the coarse grids which become smooth errors on the fine grids and, therefore, cannot be eliminated by the smoother on the fine grids. This is especially true in cases of weak to moderate anisotropy when the reduction factor is defined by smoothing properties rather than by the properties of a domain decomposition solver. The convergence in this case is still bounded above by the convergence rate of the domain decomposition solver, but we lose a lot of efficiency, especially because the domain decomposition solver is sensitive to such issues as number of blocks and overlap size.

For a given m and δ/n ratio, if the domain decomposition solver is efficient enough, then it is possible to use the same δ/n ratio on coarser grids rather than using a fixed number of cells in the overlap regions. However, adopting a fixed δ/n ratio as a general rule for overlapping reduces the solver down to the level of a domain decomposition solver. In the case of fixed overlap, the domain decomposition solver efficiency is just an upper bound on the convergence rate. The bound is sharp only for huge anisotropies (huge in strength and relevant region size).

4. Isotropic equation discretized on stretched grids. In this section, we study numerically the behavior of plane-implicit smoothers for multiblock grids when the anisotropy is not uniform but exponential. For stretched grids, each cell can have a unique level of anisotropy (see Figure 4.1). The stretching is applied near the boundary to mimic grids used in CFD simulations.

Consider an N^3 global grid partitioned into m^3 blocks (cubic partitioning)(Figure 4.1). Again, each block is supplied with an extended $\Omega^l_{p,\delta}$ subgrid. Our goal is to analyze the dependence of the convergence rate $\rho(m, n, \delta, \alpha)$ on the following parameters:

- number of blocks per side, m;
- number of grid cells per block side, n;
- overlap parameter, δ;
- stretching ratio, $\alpha = \frac{h_{i+1}}{h_i}$.

4.1. Description of test problem. All the numerical experiments reported here deal with the numerical solution of (2.1) on the unit cube $\Omega = (0,1)\times(0,1)\times(0,1)$ with the right-hand side and boundary conditions given by expressions (3.2) and (3.3). The discretization is done on 64^3 and 128^3 rectangular stretched multiblock grids with

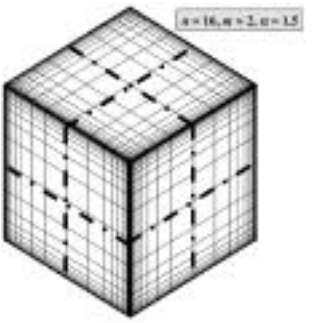


FIG. 4.1. *Rectangular stretched multiblock grid used in the numerical experiments: n is the number of cells per side in every block, m is the number of blocks per side in the split, and α is the stretching ratio.*

various overlaps ($\delta = 0, 2, 4, 8$).

As before, we tested a FAS version of V(1,1)-cycle with the blockwise alternating-direction plane smoother, unweighted solution averaging, volume-weighted residual averaging, and trilinear correction interpolation. The solutions of the 2-D problems in each plane are approximated by one 2-D V(1,1)-cycle employing alternating-direction line Gauss–Seidel smoothing. The asymptotic convergence rates observed in the numerical tests are presented in Figures 4.2 and 4.3.

4.2. Discussion of results. Numerical simulations were performed to obtain the experimental convergence rate with respect to the stretching ratio, α. The single-block and multiblock grids ($m = 1, 2$, and 4) with different overlaps ($\delta = 0, 2$, and 4) were tested. Figure 4.2 shows the results for a 64^3 grid, and Figure 4.3 for a 128^3 grid. The results can be summarized in the following two observations:

- With a 2^3 partitioning, even the minimum overlap ($\delta = 0$) is enough to exhibit good convergence rates. The results for a multiblock grid with overlap of $\delta = 2$ match with the results obtained for the single-block anisotropic case. That is, the convergence rate tends toward zero as the anisotropy increases.
- With a 4^3 partitioning, results are not as good. With the minimal overlap ($\delta = 0$), the convergence rate degrades for finer grids. However, with a larger overlap ($\delta = 2$), the convergence rate again tends towards the convergence rate demonstrated in single-block grid anisotropic cases.

5. Parallel implementation of multigrid technique. Blockwise smoothers may also be used to facilitate the parallel implementation of a multigrid method on a single-block (rectangular) grid. Notice that in this case there are global lines and planes and we are interested in blockwise smoothers only for purposes of parallel computing. To get a parallel implementation of a multigrid method, one can adopt one of the following strategies (see, e.g., [14]).

- *Domain decomposition*: A domain decomposition is applied first. Then, a multigrid method is used to solve problems inside each block.
- *Grid partitioning*: A multigrid method is used to solve the problem in the whole grid.

Domain decomposition is easier to implement and implies fewer communications (better parallel properties), but it has a negative impact on the convergence rate. On the other hand, grid partitioning implies more communication but it retains the convergence rate of the sequential algorithm (better numerical properties).

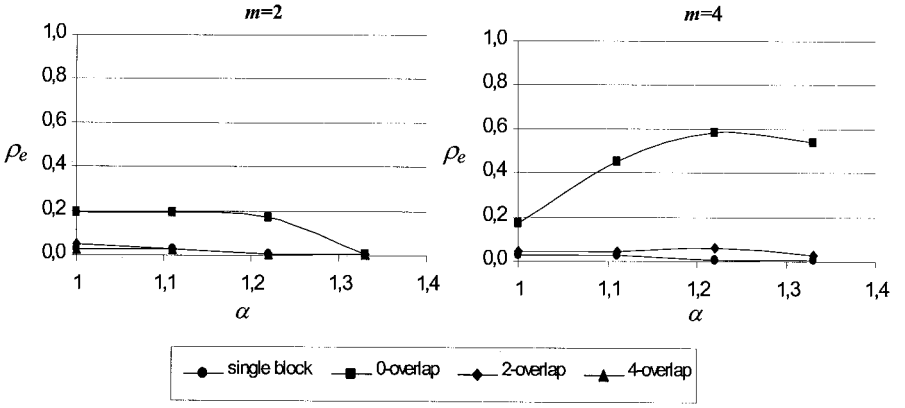


FIG. 4.2. *Experimental convergence factors, ρ_e, of one 3-D $V(1,1)$-cycle with blockwise alternating-direction plane smoother in a 64^3 grid with respect to the stretching ratio (α) for different values of the overlap (δ) and number of blocks per side (m).*

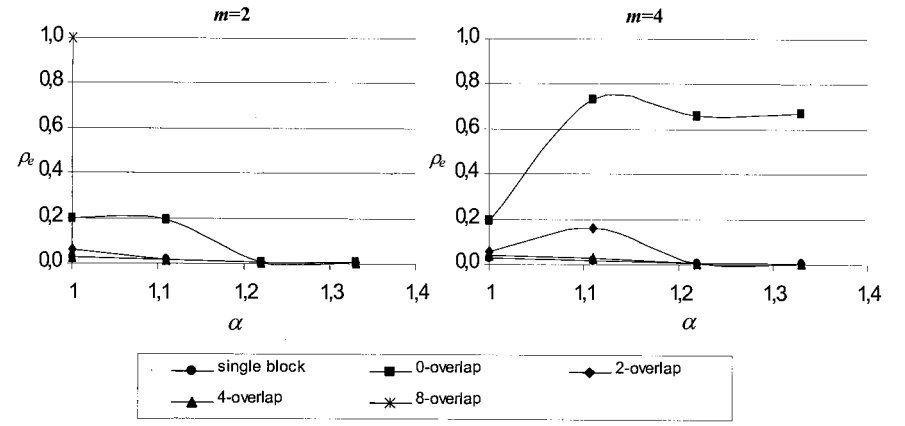


FIG. 4.3. *Experimental convergence factors, ρ_e, of one 3-D $V(1,1)$-cycle with blockwise alternating-direction plane smoother in a 128^3 grid with respect to the stretching ratio (α) for different values of the overlap (δ) and number of blocks per side (m).*

From a parallel code designer point of view, our approach is somewhere between domain decomposition and grid partitioning. Blockwise plane smoothers appear to be a tradeoff between architectural and numerical properties (domain decomposition and grid partitioning).

- *Convergence rate.* For the isotropic case the convergence rate is equal to that obtained with grid partitioning, and it approaches the convergence rate of a domain decomposition method as the anisotropy becomes stronger.
- *Communications.* Although higher than in domain decomposition, the number of communications is lower than in grid-partitioning algorithms.

However, it should be noted that due to the lack of definition of global planes and lines, grid partitioning is not viable in general multiblock grids.

Therefore, the use of blockwise smoothers is justified to facilitate parallel processing when the problem does not possess a strong anisotropy crossing the whole domain. In such a case, the expected convergence rate (when using moderate overlaps

at the block interfaces crossed by a strong anisotropy) is similar to the rate achieved with grid partitioning, but the number of communications is considerably lower.

6. Conclusions and future directions. We have analyzed the smoothing properties of blockwise plane smoothers for multiblock grids. For simplicity, our analysis and test problems have focused on rectangular grids, although the results can be extrapolated to more general multiblock grids. As a point of reference, we take the convergence rate obtained by the plane smoother in the isotropic problem on a single-block grid.

Typically, in true CFD simulations, there are no very large anisotropies crossing the whole domain. The regions of very strong anisotropies are narrow (usually where there are thin boundary layers). Therefore, it is unlikely that such a region will be divided into many blocks in the direction normal to the thin layer. Anisotropies crossing one, two, or up to four blocks are very practical, especially for stretched grids. For these cases, the overall convergence rate is still very good with a moderate overlap, as has been demonstrated in the present report.

The blockwise plane method works as a smoother for low anisotropies and becomes a domain decomposition solver for very large anisotropies. We have obtained an analytical expression that predicts the smoothing properties of the blockwise plane-implicit smoother with respect to the block size, the amount of overlap, and the strength of the anisotropy. The expression has been validated with numerical experiments.

As an example, for a line of strong anisotropy partitioned into two blocks, the multigrid convergence of reference can be obtained with an overlap of just 25 percent of the number of cells in the block, and this convergence rate can be maintained by increasing linearly the number of overlapping cells with the problem size. In actual problems, a much smaller overlap is likely to be sufficient.

We have also analyzed the case of an isotropic equation on grids which are stretched near to the boundaries, a common case in practical CFD problems. This case is even more favorable, as the convergence rate obtained for the single-block grid is maintained in multiblock grids with very small overlaps, tending to zero with increasing anisotropy strength.

Blockwise alternating-direction plane relaxation methods are found to be robust smoothers and their use should be considered for inclusion in the next generation of production CFD codes. A robust multiblock code should check whether there is an anisotropy crossing a block interface. If so, an extended subgrid overlapping with the neighboring block should be constructed. A multiblock strategy opens the possibility of using an adaptive smoother—that is, different smoothers for different portions of the domain, where choice of the smoother is based on a minimization of operation count while retaining optimum smoothing performance. An example of this would be using a plane-implicit smoother in the portions of the domain that have strong anisotropies while using a point smoother in the regions that are isotropic.

The present code updates the blocks following the lexicographic ordering. However, in order to run the code on a parallel computer, so that each processor solves a set of blocks, the update ordering must present a higher parallelism grade. The next step will be to analyze the convergence rate and architectural properties of red-black (or general-colored) ordering of the blocks. We intend to continue the work on the block-structured algorithms. In particular, we will study the applicability of blocked

alternating-direction plane methods as multigrid smoothers for convection-dominated problems and more complicated PDEs.

Acknowledgments. The results of the simulations were obtained on the Silicon Graphics Incorporated Origin 2000 operated by the Aerodynamic and Acoustic Methods Branch at the NASA Langley Research Center.

REFERENCES

[1] A. Brandt, *Multi-grid solvers on parallel computers*, in Elliptic Problem Solvers, M. Schultz, ed., Academic Press, New York, 1981, pp. 39–84.

[2] A. Brandt, *Multigrid Techniques:* 1984 *Guide with Applications to Fluid Dynamics*, Tech. Report, GMD-Studien 85, 1984.

[3] A. Brandt and B. Diskin, *Multigrid solvers for nonaligned sonic flows*, SIAM J. Sci. Comput., 21 (1999), pp. 473–501.

[4] A. Brandt and B. Diskin, *Multigrid solvers on decomposed domains*, in Domain Decomposition Methods in Science and Engineering, A. Quarteroni, J. Periaux, Y. A. Kuznetsov, and O.Widlund, eds., Contemp. Math. 157, AMS, Providence, RI, 1994, pp. 135–155.

[5] A. Brandt and B. Diskin, *Multigrid solvers for the non-aligned sonic flow: The constant coefficient case*, Computers and Fluids, 28 (1999), pp. 511–549.

[6] D. J. Mavriplis, *Multigrid strategies for viscous flow solvers on anisotropic unstructured meshes*, in Proceedings of the AIAA CFD Conference, AIAA paper 97-1952-CP, Snowmass, CO, 1997.

[7] J. E. Dendy, S. F. McCormick, J. Ruge, T. Russell, and S. Schaffer, *Multigrid methods for three-dimensional petroleum reservoir simulation*, in Tenth SPE Symposium on Reservoir Simulation, SPE paper 18049, Houston, TX, 1989.

[8] C. C. Douglas, *Caching in with multigrid algorithms: Problems in two dimensions*, Parallel Algorithms Appl., 9 (1996), pp. 195–204.

[9] J. E. Dendy Jr., *Two multigrid methods for three-dimensional problems with discontinuous and anisotropic coefficients*, SIAM J. Sci. Statist. Comput., 8 (1987), pp. 673–685.

[10] J. E. Dendy Jr., M. P. Ida, and J. M. Rutledge, *A semi-coarsening multigrid algorithm for SIMD machines*, SIAM J. Sci. Statist. Comput., 13 (1992), pp. 1460–1469.

[11] J. E. Jones and N. D. Melson, *A Note on Multi-Block Relaxation Schemes for Multigrid Solvers*, Tech. Report, ICASE 97-15, 1997.

[12] J. Linden, G. Lonsdale, H. Ritzdorf, and A. Schüller, *Scalability aspects of parallel multigrid*, Future Generation Computer Systems, 10 (1994), pp. 429–440.

[13] I. M. Llorente and N. D. Melson, *Behavior of plane relaxation methods as multigrid smoothers*, Electron. Trans. Numer. Anal., 10 (2000), pp. 92–114.

[14] I. M. Llorente and F. Tirado, *Relationships between efficiency and execution time of full multigrid methods on parallel computers*, IEEE Trans. Parallel and Distributed Systems, 8 (1997), pp. 562–573.

[15] W. Mulder, *A new multigrid approach to convection problems*, J. Comput. Phys., 83 (1989), pp. 303–323.

[16] N. H. Naik and J. V. Rosendale, *The improved robustness of multigrid elliptic solvers based on multiple semicoarsened grids*, SIAM J. Numer. Anal., 30 (1993), pp. 215–229.

[17] C. W. Oosterlee, *A GMRES-based plane smoother in multigrid to solve* 3*-D anisotropic fluid flow problems*, J. Comput. Phys., 130 (1997), pp. 41–53.

[18] A. Overman and J. V. Rosendale, *Mapping robust parallel multigrid algorithms to scalable memory architectures*, in Proceedings of the Sixth Copper Mountain Conference on Multigrid Methods, Copper Mountain, CO, 1993.

[19] S. Schaffer, *A semicoarsening multigrid method for elliptic partial differential equations with highly discontinuous and anisotropic coefficients*, SIAM J. Sci. Comput., 20 (1998), pp. 228–242.

[20] B. Smith, P. Bjorstad, and W. Gropp, *Domain Decomposition, Parallel Multilevel Methods for Elliptic Partial Differential Equations*, Cambridge University Press, Cambridge, UK, 1996.

[21] K. Stüben, *Parallel and robust multigrid with applications*, in Proceedings of the Euroconference on Supercomputation in Nonlinear and Disordered Systems: Algorithms, Applications and Architectures, L. Vázquez, F. Tirado, and I. M. Llorente, eds., Madrid, Spain, World Scientific, River Edge, NJ, 1997.

[22] C. Thole and U. Trottenberg, *Basic smoothing procedures for the multigrid treatment of elliptic* 3*d operators*, Appl. Math. Comput., 19 (1986), pp. 333–345.
[23] T. Washio and K. Oosterlee, *An evaluation of parallel multigrid as a solver and a preconditioner for singularly perturbed problems, Part* II*: Flexible* 2*D and* 3*D multiple semicoarsening*, SIAM J. Sci. Comput., 19 (1998), pp. 87–110.
[24] P. Wesseling, *An Introduction to Multigrid Methods*, John Wiley & Sons, New York, 1992.
[25] I. Yavneh, *Multigrid smoothing factors of red-black Gauss–Seidel applied to a class of elliptic operators*, SIAM J. Numer. Anal., 32 (1995), pp. 1126–1138.

SIAM J. SCI. COMPUT.
Vol. 22, No. 1, pp. 243–262
© 2000 Society for Industrial and Applied Mathematics

AN EFFICIENT PRIMAL-DUAL INTERIOR-POINT METHOD FOR MINIMIZING A SUM OF EUCLIDEAN NORMS*

KNUD D. ANDERSEN†, EDMUND CHRISTIANSEN‡, ANDREW R. CONN§, AND MICHAEL L. OVERTON¶

Abstract. The problem of minimizing a sum of Euclidean norms dates from the 17th century and may be the earliest example of duality in the mathematical programming literature. This nonsmooth optimization problem arises in many different kinds of modern scientific applications. We derive a primal-dual interior-point algorithm for the problem, by applying Newton's method directly to a system of nonlinear equations characterizing primal and dual feasibility and a perturbed complementarity condition. The main work at each step consists of solving a system of linear equations (the Schur complement equations). This Schur complement matrix is not symmetric, unlike in linear programming. We incorporate a Mehrotra-type predictor-corrector scheme and present some experimental results comparing several variations of the algorithm, including, as one option, explicit symmetrization of the Schur complement with a skew corrector term. We also present results obtained from a code implemented to solve large sparse problems, using a symmetrized Schur complement. This has been applied to problems arising in plastic collapse analysis, with hundreds of thousands of variables and millions of nonzeros in the constraint matrix. The algorithm typically finds accurate solutions in less than 50 iterations and determines physically meaningful solutions previously unobtainable.

Key words. sum of norms, nonsmooth optimization, duality, Newton method, primal-dual, interior-point method, Fermat problem

AMS subject classifications. 65K10, 73E20, 90C06, 90C25

PII. S1064827598343954

1. Introduction.

A problem which arises in many applications is to minimize a sum of Euclidean vector norms, i.e.,

$$D: \quad \min \left\{ \sum_{i=1}^{n} \|\boldsymbol{z}_i\| \ : \ \boldsymbol{y} \in \Re^m; \ \boldsymbol{z}_i \in \Re^d; \ \boldsymbol{A}_i^T \boldsymbol{y} + \boldsymbol{z}_i = \boldsymbol{c}_i, \ i = 1, \ldots, n \right\},$$

where $\boldsymbol{A}_i \in \Re^{m \times d}$, $\boldsymbol{c}_i \in \Re^d$, $i = 1, \ldots, n$, are given. In most applications $d = 2$ or $d = 3$ so that the terms in the sum are norms of vectors in a two or three-dimensional Euclidean space. If $d = 1$, the problem D is equivalent to a linear program (LP). The minimization objective is convex but not differentiable at any point where some $\boldsymbol{z}_i = 0$.

The sum of norms problem, D, has a long and interesting history. The special case $d = m = 2$, $n = 3$, $\boldsymbol{A}_i = \boldsymbol{I}$, was studied by Fermat in the 17th century. This amounts to finding the point in $\Re^2$ which minimizes the sum of distances from it

*Received by the editors August 26, 1998; accepted for publication (in revised form) March 15, 1999; published electronically June 13, 2000.
http://www.siam.org/journals/sisc/22-1/34395.html

†Lindo Systems 1415 N. Dayton St., Chicago, IL 60622 (na.kandersen@na-net.ornl.gov). The work of this author was conducted at Odense University with support from the Danish Science Council.

‡Department of Mathematics and Computer Science, Odense University, DK5230 Odense, Denmark (edc@imada. sdu.dk).

§IBM Thomas J. Watson Research Center, Yorktown Heights, NY 10598 (arconn@watson. ibm.com). Supported in part by the U.S. Department of Defense Advanced Research Projects Agency.

¶Courant Institute of Mathematical Sciences, New York University, New York, NY 10012 (overton@cs.nyu.edu). Supported in part by IBM Thomas J. Watson Research Center, the National Science Foundation, and the U.S. Department of Energy.

to three given points. In the early 19th century it was realized that this particular convex optimization problem has a natural dual maximization formulation. Kuhn [Kuh91] regards this as the first instance of duality in the mathematical programming literature. Further history is given in [Kuh67].

Duality theory for D is easily described using min-max theory. Let $\boldsymbol{x}_i \in \Re^d$, $i = 1, \ldots, n$. For consistency with standard notation for LP, we refer to $\boldsymbol{x}_i$ as the primal variables and $\boldsymbol{y}$, $\boldsymbol{z}_i$ as the dual variables, even though in our experience it is usually the dual problem D which explicitly arises in applications. We have

$$\begin{aligned}
\min_{A_i^T y + z_i = c_i} \sum_{i=1}^n \|\boldsymbol{z}_i\| &= \min_{A_i^T y + z_i = c_i} \max_{\|x_i\| \le 1} \sum_{i=1}^n \boldsymbol{x}_i^T \boldsymbol{z}_i \\
&= \max_{\|x_i\| \le 1} \min_{A_i^T y + z_i = c_i} \sum_{i=1}^n \boldsymbol{x}_i^T \boldsymbol{z}_i \\
&= \max_{\|x_i\| \le 1} \min_{y} \left(\sum_{i=1}^n \boldsymbol{c}_i^T \boldsymbol{x}_i - \boldsymbol{y}^T \sum_{i=1}^n \boldsymbol{A}_i \boldsymbol{x}_i \right) \\
&= \max \left\{ \sum_{i=1}^n \boldsymbol{c}_i^T \boldsymbol{x}_i \ : \ \|\boldsymbol{x}_i\| \le 1; \ \sum_{i=1}^n \boldsymbol{A}_i \boldsymbol{x}_i = \boldsymbol{0} \right\}.
\end{aligned}$$

The first equality follows from Cauchy–Schwartz, the second from min-max theory [Roc70, Cor. 37.3.2], the third trivially, and the fourth because if $\sum_{i=1}^n \boldsymbol{A}_i \boldsymbol{x}_i$ is not zero, the minimized value would be $-\infty$. Therefore, the dual of D is the primal problem

$$P: \quad \max \left\{ \sum_{i=1}^n \boldsymbol{c}_i^T \boldsymbol{x}_i \ : \ \boldsymbol{x}_i \in \Re^d, \ \|\boldsymbol{x}_i\| \le 1, \ i = 1, \ldots, n; \ \sum_{i=1}^n \boldsymbol{A}_i \boldsymbol{x}_i = \boldsymbol{0} \right\}.$$

This result is an easy generalization of the duality theory in [Kuh67] and is a special case of a much more general theory of convex optimization given in [NN94]. Nonetheless, we are not aware of its explicit appearance in the literature earlier than [And96b].

Although the duality theory has been known in its simplest form for nearly two centuries, it was not understood until relatively recently how to exploit duality in algorithms for minimizing D. Iteratively reweighted least squares (Weiszfeld's method [Wei37]) has long been used as a robust though slowly converging method to solve D. Another well-known approach is to replace the terms $\|\boldsymbol{z}_i\|$ in the objective by the differentiable quantity $\sqrt{\|\boldsymbol{z}_i\|^2 + \mu^2}$, where μ is a fixed positive number. This method is also robust but converges arbitrarily slowly as $\mu \to 0$. Neither of these algorithms use any aspect of duality. In both cases, the reason for the slow convergence is that, in most interesting applications, some of the norms in the objective D have zero as their optimal value.

Calamai and Conn [CC80,CC87] and Overton [Ove83] solved D using Newton methods combined with an active set approach to determine which norms $\|\boldsymbol{z}_i\|$ are zero at an optimal solution. These methods were the first to exploit the duality structure of the problem, as they explicitly compute both primal and dual solutions. However, Newton's method was derived in the $\boldsymbol{y}, \boldsymbol{z}$ space only, with the $\boldsymbol{x}$ variables computed by least-squares estimates. The methods of Calamai and Conn and Overton are quite efficient if not many norms $\|\boldsymbol{z}_i\|$ are zero. However, if this number is large, the number of iterations is typically also large because the active set of zero norms must be updated at every step.

During the past decade, it was realized that the class of interior-point methods, so successful for solving LP's, could be extended to solve many convex optimization problems; the primary reference for this important development is Nesterov and Nemirovskii [NN94]. Andersen [And96b] gave a specific method for solving D which is based on a primal interior-point method for LP. In this method the terms $\|\boldsymbol{z}_i\|$ are replaced by $\sqrt{\|\boldsymbol{z}_i\|^2 + \mu^2}$, but the quantity μ is treated as an extra variable, whose value is determined by duality estimates. Using this method, Andersen was the first to be able to solve D rapidly and accurately even when the number of variables is large and many norms $\|\boldsymbol{z}_i\|$ are zero at a solution point. In [AC98] it was demonstrated how the linearly constrained problem can be reduced to the unconstrained case using an exact l_1 penalty function, while still preserving the sparsity structure.

In this paper we present a primal-dual interior-point method for solving P and D. The basic algorithm is easy to motivate and implement. The number of iterations needed is substantially less than that required for the primal interior-point method used by Andersen [And96b]. This is consistent with general experience with primal-dual versus primal interior-point methods for LP [Wri97].

The sum of norms problem is a special case of quadratically constrained quadratic programming (QCQP), also known as optimization over the quadratic cone. Nesterov and Todd [NT98a,NT98b] gave a theoretical discussion of algorithms for optimization over homogeneous self-dual cones, including the quadratic cone. See also Adler and Alizadeh [AA95] for another primal-dual algorithmic approach to QCQP. Our view is that the sum of norms problem is sufficiently important that a specialized approach is justified. Also taking this view, Xue and Ye [XY97] gave a complexity analysis of the sum of norms problem, using an interior-point method and exploiting the general theory given in [NT98a,NT98b].

Our primal-dual algorithm is derived in the next section, applying Newton's method to three conditions: primal and dual feasibility, and complementarity. A key point is the derivation of the appropriate complementarity condition. The main work at each step consists of solving a system of linear equations (the Schur complement equations). This Schur complement matrix is not symmetric, unlike its counterpart in linear programming.

Section 3 discusses a Mehrotra predictor-corrector enhancement to the algorithm and considers symmetrizing the Schur complement equations, including a compensating skew corrector term. Section 4 presents experimental results for some small Steiner tree test problems.

Section 5 discusses a large-scale implementation using a symmetrized Schur complement. This has been used to solve applied problems arising in plastic collapse analysis with hundreds of thousands of variables and millions of nonzeros in the constraint matrix. The algorithm typically finds accurate solutions in less than 50 iterations and determines physically meaningful solutions that were considered unobtainable until now.

In fact, problem D arises in many applications. These include least-distance problems in two or three-dimensional Euclidean space, such as the classical Steiner tree problem. Xue and his coauthors [XLD99a,XLD99b] have investigated a wide variety of problems of this kind; see their papers for other references. However, problem D also arises in very different contexts. Alpert et al. [ACK$^+$98] have recently applied a variant of our algorithm presented in this paper to the placement of circuits in VLSI design. Chan, Golub, and Mulet [CGM96] applied a nonlinear version of the algorithm to some applications in image reconstruction. Ito and Kunisch [IK99] used

an active set method to solve related image reconstruction problems. Byrnes and Bright [BB] used iteratively reweighted least-squares to solve trajectory optimization problems in space exploration. In fact, this method (Weiszfeld's method) has long been used at the Jet Propulsion Laboratory as a basic workhorse to solve problems of the form D that arise in spacecraft missions such as the Galileo and Pioneer "fly-by's" of the outer planets [Mai87,Mic]. Strang [Str79] considered an isoparametric design problem to which Overton [Ove84] applied a version of his algorithm mentioned above. Parks [Par91] has applied related methods to solve minimal surface (soap bubble) problems. Alexander and Maddocks [AM93] used the method of [Ove83] to solve friction problems arising in robotics. Finally, plastic collapse analysis problems have been mentioned already and are discussed at greater length in the final section of the paper. A key similarity in all these applications is that some, and perhaps many, of the norms in the sum to be minimized can be expected to have the value zero at an optimal solution.

We believe there is great opportunity to apply the primal-dual method given in this paper to these and many other interesting applications.

Notation. Let $\boldsymbol{I}_d$ denote the $d \times d$ identity matrix. Let

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_n \end{bmatrix} \in \Re^{dn}, \quad \boldsymbol{z} = \begin{bmatrix} \boldsymbol{z}_1 \\ \vdots \\ \boldsymbol{z}_n \end{bmatrix} \in \Re^{dn}, \quad \boldsymbol{c} = \begin{bmatrix} \boldsymbol{c}_1 \\ \vdots \\ \boldsymbol{c}_n \end{bmatrix} \in \Re^{dn},$$

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{A}_1 & \cdots & \boldsymbol{A}_n \end{bmatrix} \in \Re^{m \times dn}.$$

The primal feasible region is given by

$$\mathcal{X} = \left\{ \boldsymbol{x} \in \Re^{dn} : \boldsymbol{A}\boldsymbol{x} = \boldsymbol{0}; \quad \|\boldsymbol{x}_i\| \leq 1, \ i = 1, \ldots, n \right\} \tag{1}$$

and the dual feasible region is

$$\mathcal{Y} = \left\{ (\boldsymbol{y}, \boldsymbol{z}) \in \Re^m \times \Re^{dn} : \boldsymbol{A}^T\boldsymbol{y} + \boldsymbol{z} = \boldsymbol{c} \right\}. \tag{2}$$

Consequently, we may rewrite D as

$$D: \quad \min \left\{ \sum_{i=1}^{n} \|\boldsymbol{z}_i\| \ : \ (\boldsymbol{y}, \boldsymbol{z}) \in \mathcal{Y} \right\}$$

and P as

$$P: \quad \max \ \left\{ \boldsymbol{c}^T\boldsymbol{x} \ : \ \boldsymbol{x} \in \mathcal{X} \right\}.$$

2. Complementarity and Newton's method. Suppose $\boldsymbol{x} \in \mathcal{X}$ and $(\boldsymbol{y}, \boldsymbol{z}) \in \mathcal{Y}$ are, respectively, primal and dual feasible. Then the duality gap, i.e., difference between the primal and dual objective functions, is

$$\sum_{i=1}^{n} \|\boldsymbol{z}_i\| - \sum_{i=1}^{n} \boldsymbol{c}_i^T\boldsymbol{x}_i = \sum_{i=1}^{n} \left(\|\boldsymbol{z}_i\| - \boldsymbol{x}_i^T\boldsymbol{z}_i \right) \geq 0. \tag{3}$$

The duality gap must be zero at an optimal solution. It is zero if and only if, for each $i = 1, \ldots, n$, either $\|\boldsymbol{z}_i\|$ is zero or $\boldsymbol{x}_i = \boldsymbol{z}_i / \|\boldsymbol{z}_i\|$. This complementarity condition can be conveniently expressed as

$$\boldsymbol{z}_i - \|\boldsymbol{z}_i\|\boldsymbol{x}_i = \boldsymbol{0}, \quad i = 1, \ldots, n. \tag{4}$$

It follows from the complementarity condition that for each i, either $\boldsymbol{z}_i = 0$ or $\|\boldsymbol{x}_i\| = 1$; we say that strict complementarity holds if, for each i, only one of these two conditions holds. It may happen that no strictly complementary solution exists, unlike in LP.

Primal-dual interior-point methods are based on Newton's method applied to three sets of equations: primal feasibility, dual feasibility, and an appropriate complementarity/centering condition. The feasibility equations are, respectively,

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{0} \tag{5}$$

and

$$\boldsymbol{A}^T\boldsymbol{y} + \boldsymbol{z} = \boldsymbol{c}. \tag{6}$$

We assume from now on that the $m \times dn$ matrix $\boldsymbol{A}$ has full rank. We also assume that $m < dn$, since otherwise P and D are solved by $\boldsymbol{x} = \boldsymbol{0}$, $\boldsymbol{z} = \boldsymbol{0}$.

The primal and dual feasibility equations consist of $m + dn$ equations in the $m + 2dn$ scalar variables represented by $\boldsymbol{y}$ and $\boldsymbol{x}$, $\boldsymbol{z}$. To make this a square system we need another dn equations, which are available in the form of the complementarity condition (4). This condition is not differentiable if $\|\boldsymbol{z}_i\|$ is zero, but it may be replaced by the centering condition

$$\boldsymbol{z}_i - \left(\|\boldsymbol{z}_i\|^2 + \mu^2\right)^{\frac{1}{2}} \boldsymbol{x}_i = \boldsymbol{0}, \quad i = 1, \ldots, n, \tag{7}$$

where $\mu > 0$.

The following theorem is from [And96b], showing that the centering condition (7) is in fact the complementarity condition for the following pair of smooth optimization problems:

$$D_\mu : \quad \min \left\{ \sum_{i=1}^{n} \left(\|\boldsymbol{z}_i\|^2 + \mu^2\right)^{\frac{1}{2}} : \ (\boldsymbol{y}, \boldsymbol{z}) \in \mathcal{Y} \right\},$$

$$P_\mu : \quad \max \left\{ \boldsymbol{c}^T\boldsymbol{x} + \mu \sum_{i=1}^{n} \left(1 - \|\boldsymbol{x}_i\|^2\right)^{\frac{1}{2}} : \ \boldsymbol{x} \in \mathcal{X} \right\}.$$

THEOREM 1. *The problems D_μ and P_μ are a primal-dual pair. Specifically, D_μ has the solution $(\boldsymbol{y}(\mu), \boldsymbol{z}(\mu))$ and P_μ has the solution $\boldsymbol{x}(\mu)$, all satisfying* (5), (6), *and* (7).

Proof. The proof is a simple modification of the proof (given in section 1) that P and D are a primal-dual pair. See [And96b] for details. □

This theorem shows that introducing the centering parameter μ in the complementarity conditions for the original pair of problems is equivalent to smoothing the norms in D and introducing a cost into P which moves the primal solution away from its boundary. The solutions $(\boldsymbol{x}(\mu), \boldsymbol{y}(\mu), \boldsymbol{z}(\mu))$ of P_μ, D_μ, for $\mu > 0$, define a sort of central path for P, D, though not one derived from a logarithmic barrier function and therefore not centered in the usual sense.

Let us write the centering condition (7) as

$$\Theta(\mu, \boldsymbol{z})\boldsymbol{x} - \boldsymbol{z} = \boldsymbol{0}, \quad \text{where} \quad \Theta(\mu, \boldsymbol{z}) = \operatorname{Diag}\left(\left(\|\boldsymbol{z}_i\|^2 + \mu^2\right)^{\frac{1}{2}} \boldsymbol{I}_d\right). \tag{8}$$

Collecting (6), (5), and (8) together, we have the nonlinear system of equations

$$\boldsymbol{G}_\mu(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}) = \begin{pmatrix} \boldsymbol{A}^T\boldsymbol{y} + \boldsymbol{z} - \boldsymbol{c} \\ \boldsymbol{A}\boldsymbol{x} \\ \Theta(\mu,\boldsymbol{z})\boldsymbol{x} - \boldsymbol{z} \end{pmatrix} = \boldsymbol{0}, \tag{9}$$

whose solution is $(\boldsymbol{x}(\mu),\boldsymbol{y}(\mu),\boldsymbol{z}(\mu))$. Newton's method applied to $\boldsymbol{G}_\mu$ at a given point $(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z})$ gives the following linear system defining updates to the variables:

$$\begin{bmatrix} \boldsymbol{0} & \boldsymbol{A}^T & \boldsymbol{I}_{dn} \\ \boldsymbol{A} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{E}_\mu & \boldsymbol{0} & -\boldsymbol{F}_\mu \end{bmatrix} \begin{bmatrix} \Delta\boldsymbol{x} \\ \Delta\boldsymbol{y} \\ \Delta\boldsymbol{z} \end{bmatrix} = \begin{bmatrix} \boldsymbol{r}_d \\ \boldsymbol{r}_p \\ \boldsymbol{r}_c \end{bmatrix}, \tag{10}$$

where

$$\boldsymbol{r}_d = \boldsymbol{c} - \boldsymbol{A}^T\boldsymbol{y} - \boldsymbol{z}, \quad \boldsymbol{r}_p = -\boldsymbol{A}\boldsymbol{x}, \quad \boldsymbol{r}_c = \boldsymbol{z} - \boldsymbol{E}_\mu\boldsymbol{x}, \tag{11}$$

$$\boldsymbol{E}_\mu = \operatorname{Diag}\left(\omega_i^\mu \boldsymbol{I}_d\right), \quad \boldsymbol{F}_\mu = \operatorname{Diag}\left(\boldsymbol{I}_d - \frac{1}{\omega_i^\mu}\boldsymbol{x}_i\boldsymbol{z}_i^T\right), \tag{12}$$

and

$$\omega_i^\mu = \left(\|\boldsymbol{z}_i\|^2 + \mu^2\right)^{\frac{1}{2}}. \tag{13}$$

The equation $\boldsymbol{r}_d = \boldsymbol{0}$ is maintained exactly at each step, by always defining $\boldsymbol{z} = \boldsymbol{c} - \boldsymbol{A}^T\boldsymbol{y}$. Although $\boldsymbol{r}_p$ can be set to zero initially by setting the first iterate $\boldsymbol{x} = \boldsymbol{0}$ or setting $\boldsymbol{x}$ to a null vector of $\boldsymbol{A}$, we do not assume this, since primal feasibility cannot be maintained exactly in the presence of rounding errors.

Eliminating $\Delta\boldsymbol{z}$ and using $\boldsymbol{r}_d = \boldsymbol{0}$, we have

$$\begin{bmatrix} \boldsymbol{F}_\mu^{-1}\boldsymbol{E}_\mu & \boldsymbol{A}^T \\ \boldsymbol{A} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \Delta\boldsymbol{x} \\ \Delta\boldsymbol{y} \end{bmatrix} = \begin{bmatrix} \boldsymbol{F}_\mu^{-1}\boldsymbol{r}_c \\ -\boldsymbol{A}\boldsymbol{x} \end{bmatrix}. \tag{14}$$

Defining $\boldsymbol{H}_\mu = \boldsymbol{E}_\mu^{-1}\boldsymbol{F}_\mu$, and using the definition of $\boldsymbol{r}_c$ in (11), we find after one more elimination step that

$$\boldsymbol{A}\boldsymbol{H}_\mu\boldsymbol{A}^T\Delta\boldsymbol{y} = \boldsymbol{A}\boldsymbol{E}_\mu^{-1}\boldsymbol{z} \tag{15}$$

and

$$\Delta\boldsymbol{x} = \boldsymbol{E}_\mu^{-1}\left(\boldsymbol{F}_\mu\Delta\boldsymbol{z} + \boldsymbol{r}_c\right), \tag{16}$$

where (immediately from dual feasibility)

$$\Delta\boldsymbol{z} = -\boldsymbol{A}^T\Delta\boldsymbol{y}. \tag{17}$$

The operations of multiplying vectors by $\boldsymbol{F}_\mu$ and $\boldsymbol{E}_\mu^{-1}$ are trivial since $\boldsymbol{F}_\mu$ is block diagonal and $\boldsymbol{E}_\mu$ is positive diagonal. Notice the explicit dependence of $\boldsymbol{E}_\mu$ and $\boldsymbol{F}_\mu$ on the centering parameter μ, in contrast to the situation in LP, where the corresponding diagonal matrices depend only on the current variables. This is a consequence of the more complicated nature of the complementarity condition (4).

The main cost of this process is forming and factoring the Schur complement $\boldsymbol{A}\boldsymbol{H}_\mu\boldsymbol{A}^T$. Except in the trivial case $d = 1$, the block-diagonal matrix $\boldsymbol{H}_\mu$ is not generally symmetric, since $\boldsymbol{F}_\mu$ is not. This presents difficulties for large sparse problems, where it is highly desirable to use sparse Cholesky techniques which apply only to symmetric, positive definite systems. Sparse LU factorization techniques for nonsymmetric linear systems are more costly since they require pivoting for stability and therefore are not able to exploit sparsity as effectively as Cholesky methods.

However, note that $\boldsymbol{F}_\mu$ is positive definite (in the sense that $v^T\boldsymbol{F}_\mu v > 0$ for all $v \neq 0$) as long as $\boldsymbol{x} \in \mathcal{X}$, and therefore so are $\boldsymbol{H}_\mu$ and $\boldsymbol{A}\boldsymbol{H}_\mu\boldsymbol{A}^T$ (since $\boldsymbol{E}_\mu$ and $\boldsymbol{F}_\mu$ commute). Furthermore, it follows from (7) that for $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) = (\boldsymbol{x}(\mu), \boldsymbol{y}(\mu), \boldsymbol{z}(\mu))$ (see Theorem 1), the matrix $\boldsymbol{F}_\mu$ and therefore also $\boldsymbol{H}_\mu$ and $\boldsymbol{A}\boldsymbol{H}_\mu\boldsymbol{A}^T$ are symmetric for all $\mu > 0$. Consequently, when defined sufficiently close to the "central path," $\boldsymbol{A}\boldsymbol{H}_\mu\boldsymbol{A}^T$ is nearly symmetric. One of the issues we shall discuss in the next section is the effect of symmetrizing $\boldsymbol{H}_\mu$ by defining it to be $\frac{1}{2}\boldsymbol{E}_\mu^{-1}(\boldsymbol{F}_\mu + \boldsymbol{F}_\mu^T)$ instead of $\boldsymbol{E}_\mu^{-1}\boldsymbol{F}_\mu$.

As $\mu \to 0$, for each i, either $\boldsymbol{z}_i(\mu)$ or $\boldsymbol{x}_i(\mu) - \boldsymbol{z}_i(\mu)/\|\boldsymbol{z}_i(\mu)\|$ converges to zero. In the latter case, the limit of the ith block of the corresponding $\boldsymbol{F}_\mu$ is singular, while in the former case the limit of the ith block of the corresponding $\boldsymbol{E}_\mu$ is zero.

We now discuss how to update the iterates $\boldsymbol{x}$, $\boldsymbol{y}$, and $\boldsymbol{z}$ after $\Delta\boldsymbol{x}$, $\Delta\boldsymbol{y}$, and $\Delta\boldsymbol{z}$ are computed. We start by observing that $\Delta\boldsymbol{z}$ is a descent direction for the smoothed dual objective function in D_μ: the gradient of this function with respect to $\boldsymbol{z}$ is easily seen to be $\boldsymbol{E}_\mu^{-1}\boldsymbol{z}$. Using (17) and (15), we have

$$\begin{aligned}(\Delta\boldsymbol{z})^T\boldsymbol{E}_\mu^{-1}\boldsymbol{z} &= -(\Delta\boldsymbol{y})^T\boldsymbol{A}\boldsymbol{E}_\mu^{-1}\boldsymbol{z} \\ &= -(\Delta\boldsymbol{y})^T\boldsymbol{A}\boldsymbol{H}_\mu\boldsymbol{A}^T\Delta\boldsymbol{y} \\ &< 0\end{aligned}$$

(unless $\boldsymbol{z} = \boldsymbol{0}$), since the symmetric part of $\boldsymbol{A}\boldsymbol{H}_\mu\boldsymbol{A}^T$ is positive definite. Consequently, it is natural to update $\boldsymbol{y}$ and $\boldsymbol{z}$ by using a line search on the smoothed dual function in D_μ, as follows.

Dual line search rule.

$$\tilde{\boldsymbol{y}} = \boldsymbol{y} + \hat{\beta}\Delta\boldsymbol{y}, \quad \tilde{\boldsymbol{z}} = \boldsymbol{z} + \hat{\beta}\Delta\boldsymbol{z}, \tag{18}$$

where

$$\hat{\beta} \approx \arg\min_{0\leq\beta\leq 1} \sum_{i=1}^{n} \left(\|\boldsymbol{z}_i + \beta\Delta\boldsymbol{z}_i\|^2 + \mu^2\right)^{\frac{1}{2}}. \tag{19}$$

The same steplength must be used for $\boldsymbol{y}$ and $\boldsymbol{z}$ to maintain dual feasibility. Of course, the univariate minimization problem need not be solved exactly.

The direction $\Delta\boldsymbol{x}$ is not necessarily an ascent direction for the penalized primal objective function in P_μ, so a line search is not appropriate to update the primal iterate $\boldsymbol{x}$. We consider two possibilities: the primal scaling rule and the primal steplength rule.

Primal scaling rule.

$$\tilde{\boldsymbol{x}} = \bar{\gamma}\,(\boldsymbol{x} + \Delta\boldsymbol{x}), \tag{20}$$

where

$$\bar{\gamma} = \max\{\gamma : \quad \gamma\|\boldsymbol{x}_i + \Delta\boldsymbol{x}_i\| \leq 1 \quad i = 1, \ldots, n\}. \tag{21}$$

The scaling rule is a trivial computation.

Primal steplength rule. The step to the boundary is given by

$$\alpha_{\max} = \max\{\alpha : \ \|\boldsymbol{x}_i + \alpha \Delta \boldsymbol{x}_i\| \leq 1, \ i = 1, \ldots, n\} \ = \ \min_i \alpha_i,$$

where α_i is the positive root of a quadratic equation:

$$\alpha_i = \frac{-\Delta \boldsymbol{x}_i^T \boldsymbol{x}_i + \sqrt{\|\Delta \boldsymbol{x}_i\|^2 \left(1 - \|\boldsymbol{x}_i\|^2\right) + \left(\Delta \boldsymbol{x}_i^T \boldsymbol{x}_i\right)^2}}{\|\Delta \boldsymbol{x}_i\|^2}, \quad i = 1, \ldots, n.$$

We choose $0 \ll \tau < 1$ and define

$$\bar{\alpha} = \tau \alpha_{\max}. \tag{22}$$

Then the steplength rule is defined by

$$\tilde{\boldsymbol{x}} = \boldsymbol{x} + \min(1, \bar{\alpha}) \Delta \boldsymbol{x}. \tag{23}$$

Both rules preserve primal feasibility in exact arithmetic. For the steplength rule, conventional experience with primal-dual interior-point methods dictates a choice of τ less than 1, but not much less, for example, $\tau = .99$ or $\tau = .999$. For sums of norms, however, we found that $\tau = 1$ also works quite well. This allows iterates $\boldsymbol{x}$ to actually reach the boundary of the feasible region, but the matrix $\boldsymbol{F}_\mu$ still cannot be singular as long as $\mu > 0$. Increased ill-conditioning of the linear systems which are solved as convergence takes place is a standard feature of interior-point methods and generally does not cause great difficulties except when the iterates are nearly optimal. The reason we do not allow $\tau = 1$ is that rounding errors may then cause the updated $\boldsymbol{x}$ to lie just outside the feasible region.

The scaling rule always places $\boldsymbol{x}$ exactly on the boundary of the feasible region. This is not appropriate if the solution $\boldsymbol{x}$ has all component norms $\|\boldsymbol{x}_i\| < 1$, but this is a trivial case since then the dual solution must be zero by complementarity. As far as we know, the scaling rule does not have an analogy in standard interior-point implementations: such a rule is possible only when the primal equality constraints are homogeneous as they are here.

Equations (15), (16), (17), and the updating rules just described define the basic ingredients of a primal-dual interior-point method for solving P and D. To complete the description of the algorithm we must define a rule for updating the parameter μ: for this we introduce a predictor-corrector method.

3. Mehrotra's predictor-corrector method and a symmetrized algorithm with a skew correction term. Mehrotra's predictor-corrector method is a standard tool in primal-dual interior-point software for LP. The basic idea is to first compute a predictor step defined by first-order approximations to the optimality conditions (i.e. Newton's method), and to follow this with a corrector step which also takes second-order terms into account. A key point is that both predictor and corrector use the same matrix factorization; only the right-hand sides of the linear equations defining the steps differ. Another key component is a technique for estimating the centering parameter μ. Mehrotra's method was originally given in [Meh92]; an excellent discussion may be found in [Wri97, Chapter 10].

We now discuss how to adapt Mehrotra's method to our problem. Let us replace $\boldsymbol{x}$, $\boldsymbol{y}$, and $\boldsymbol{z}$ in the centering condition (8) by $\boldsymbol{x}+\Delta\boldsymbol{x}$, $\boldsymbol{y}+\Delta\boldsymbol{y}$, and $\boldsymbol{z}+\Delta\boldsymbol{z}$, respectively, obtaining, for $i = 1, \ldots, n$,

$$\boldsymbol{z}_i + \Delta\boldsymbol{z}_i - \left(\|\boldsymbol{z}_i + \Delta\boldsymbol{z}_i\|^2 + \mu^2\right)^{\frac{1}{2}} (\boldsymbol{x}_i + \Delta\boldsymbol{x}_i) = \boldsymbol{0},$$

i.e.,

$$\boldsymbol{z}_i + \Delta\boldsymbol{z}_i - \omega_i^{\mu} \left(1 + 2\frac{\boldsymbol{z}_i^T \Delta\boldsymbol{z}_i}{(\omega_i^{\mu})^2} + \frac{\|\Delta\boldsymbol{z}_i\|^2}{(\omega_i^{\mu})^2}\right)^{\frac{1}{2}} (\boldsymbol{x}_i + \Delta\boldsymbol{x}_i) = \boldsymbol{0},$$

which gives

$$\boldsymbol{z}_i + \Delta\boldsymbol{z}_i - \omega_i^{\mu} \left(1 + \frac{\boldsymbol{z}_i^T \Delta\boldsymbol{z}_i}{(\omega_i^{\mu})^2} + \frac{\|\Delta\boldsymbol{z}_i\|^2}{2(\omega_i^{\mu})^2} - \frac{(\boldsymbol{z}_i^T \Delta\boldsymbol{z}_i)^2}{2(\omega_i^{\mu})^4} + \cdots\right) (\boldsymbol{x}_i + \Delta\boldsymbol{x}_i) = \boldsymbol{0}.$$

Moving first-order terms to the left-hand side, constant and second-order terms to the right-hand side, neglecting higher than second-order terms, and changing the sign of the equation we obtain

$$(\boldsymbol{E}_{\mu}\,\Delta\boldsymbol{x} - \boldsymbol{F}_{\mu}\,\Delta\boldsymbol{z})_i = (\boldsymbol{r}_c)_i - \frac{\boldsymbol{z}_i^T \Delta\boldsymbol{z}_i}{\omega_i^{\mu}} \Delta\boldsymbol{x}_i - \frac{\|\Delta\boldsymbol{z}_i\|^2}{2\omega_i^{\mu}} \boldsymbol{x}_i + \frac{(\boldsymbol{z}_i^T \Delta\boldsymbol{z}_i)^2}{2(\omega_i^{\mu})^3} \boldsymbol{x}_i \tag{24}$$

for $i = 1, \ldots, n$. The idea, then, is to compute the predictor steps $\Delta\boldsymbol{x}$, $\Delta\boldsymbol{y}$, and $\Delta\boldsymbol{z}$ from (15)–(17), and then use these to define the second-order terms which are included in the right-hand side of the linear system solved to obtain the corrector step, using the factorization of $\boldsymbol{A}\boldsymbol{H}_{\mu}\boldsymbol{A}^T$ a second time.

As noted above, a key component of Mehrotra's method is to exploit the result of the predictor step to define a heuristic value for the centering parameter μ to be used in the computation of the corrector step. This is provided by

$$\tilde{\mu} = \frac{(\mathrm{gap}(\boldsymbol{x} + \Delta\boldsymbol{x}, \boldsymbol{z} + \Delta\boldsymbol{z}))^3}{n\,(\mathrm{gap}(\boldsymbol{x}, \boldsymbol{z}))^2}, \tag{25}$$

where

$$\mathrm{gap}(\boldsymbol{x}, \boldsymbol{z}) = \sum_{i=1}^{n} \left(\|\boldsymbol{z}_i\| - \boldsymbol{x}_i^T \boldsymbol{z}_i\right). \tag{26}$$

This is a natural generalization of Mehrotra's formula for LP. The value $\tilde{\mu}$ may be substituted for μ in *all* the terms on the right-hand side of (24), including the second-order terms as well as the constant term, giving

$$\boldsymbol{h}_i^{(1)} = \boldsymbol{z}_i - \omega_i^{\tilde{\mu}} \boldsymbol{x}_i - \frac{\boldsymbol{z}_i^T \Delta\boldsymbol{z}_i}{\omega_i^{\tilde{\mu}}} \Delta\boldsymbol{x}_i - \frac{\|\Delta\boldsymbol{z}_i\|^2}{2\omega_i^{\tilde{\mu}}} \boldsymbol{x}_i + \frac{(\boldsymbol{z}_i^T \Delta\boldsymbol{z}_i)^2}{2(\omega_i^{\tilde{\mu}})^3} \boldsymbol{x}_i, \tag{27}$$

where

$$\omega_i^{\tilde{\mu}} = \left(\|\boldsymbol{z}_i\|^2 + \tilde{\mu}^2\right)^{\frac{1}{2}}. \tag{28}$$

It is not practical to substitute $\tilde{\mu}$ for μ on the left-hand side of (24), since the factorization of $\boldsymbol{H}_\mu$ has already been computed using the previous value for μ. Consequently, we also add to the right-hand side of (24) further corrector terms of the form

$$\boldsymbol{h}_i^{(2)} = \left(\omega_i^\mu - \omega_i^{\tilde{\mu}}\right) \Delta \boldsymbol{x}_i + \left(\frac{1}{\omega_i^\mu} - \frac{1}{\omega_i^{\tilde{\mu}}}\right) \left(\boldsymbol{z}_i^T \Delta \boldsymbol{z}_i\right) \boldsymbol{x}_i. \tag{29}$$

As noted in the previous section, the nonsymmetry of $\boldsymbol{H}_\mu$ is a major disadvantage for large sparse problems. We therefore consider here the idea of explicitly symmetrizing $\boldsymbol{H}_\mu$, defining it to be $\frac{1}{2}\boldsymbol{E}_\mu^{-1}(\boldsymbol{F}_\mu + \boldsymbol{F}_\mu^T)$ instead of $\boldsymbol{E}_\mu^{-1}\boldsymbol{F}_\mu$. This suggests subtracting a skew correction term $\frac{1}{2}\boldsymbol{E}_\mu^{-1}(\boldsymbol{F}_\mu - \boldsymbol{F}_\mu^T)\Delta\boldsymbol{z}$ from the right-hand side, i.e., adding terms

$$\boldsymbol{h}_i^{(3)} = \frac{1}{2\omega_i^\mu} \left((\Delta \boldsymbol{z}_i^T \boldsymbol{x}_i)\boldsymbol{z}_i - (\Delta \boldsymbol{z}_i^T \boldsymbol{z}_i)\boldsymbol{x}_i\right) \tag{30}$$

to (24). Note the use of ω_i^μ, not $\omega_i^{\tilde{\mu}}$, in the denominator. Putting all this together, the corrector step is defined by

$$\boldsymbol{E}_\mu \, \Delta \boldsymbol{x} - \boldsymbol{F}_\mu \, \Delta \boldsymbol{z} = \boldsymbol{r}_c^c, \tag{31}$$

where the ith block of the "corrected centering" residual is

$$(\boldsymbol{r}_c^c)_i = \begin{cases} \boldsymbol{h}_i^{(1)} + \boldsymbol{h}_i^{(2)} & \text{if} \quad \boldsymbol{H}_\mu = \boldsymbol{E}_\mu^{-1}\boldsymbol{F}_\mu, \\ \boldsymbol{h}_i^{(1)} + \boldsymbol{h}_i^{(2)} + \boldsymbol{h}_i^{(3)} & \text{if} \quad \boldsymbol{H}_\mu = \frac{1}{2}\boldsymbol{E}_\mu^{-1}\left(\boldsymbol{F}_\mu + \boldsymbol{F}_\mu^T\right), \end{cases} \tag{32}$$

using (27), (29), and (30).

Substituting $\boldsymbol{r}_c^c$ for $\boldsymbol{r}_c$ in (14), we therefore compute the corrector steps from

$$\boldsymbol{A}\boldsymbol{H}_\mu\boldsymbol{A}^T\Delta\boldsymbol{y} = \boldsymbol{A}\left(\boldsymbol{E}_\mu^{-1}\boldsymbol{r}_c^c + \boldsymbol{x}\right) \tag{33}$$

and

$$\Delta\boldsymbol{x} = \boldsymbol{E}_\mu^{-1}\left(\boldsymbol{F}_\mu\Delta\boldsymbol{z} + \boldsymbol{r}_c^c\right) \tag{34}$$

with $\Delta\boldsymbol{z}$ given by (17).

Note that by analogy with standard practice in LP, it might seem appropriate to modify the right-hand side $\boldsymbol{r}_c$ used by the predictor step by substituting 0 for μ in its definition.[1] In practice, whether or not this is done seems to have little effect, but one reason not to make this choice is that then the dual predictor step is no longer guaranteed to be a descent direction for the smoothed objective function in D_μ. There is no guarantee that the dual corrector step is a descent direction for either this function or the corresponding function defined using $\tilde{\mu}$ instead of μ, although it usually is. If it is a descent direction for the latter function, we update the iterates as before, using $\tilde{\mu}$ instead of μ in the objective function in the dual line search. Otherwise, we abandon both primal and dual corrector steps and use the predictor steps instead.

We now summarize the algorithm. We initialize it with $\boldsymbol{x} = \boldsymbol{0}$, $\boldsymbol{y}$ set to the minimizer of $\|\boldsymbol{c} - \boldsymbol{A}^T\boldsymbol{y}\|$, and $\boldsymbol{z} = \boldsymbol{c} - \boldsymbol{A}^T\boldsymbol{y}$. Assume that an initial value of $\mu > 0$ is given, as well as a termination tolerance ϵ.

[1] In fact, this was done in the experiments reported in section 5.

PREDICTOR-CORRECTOR ALGORITHM FOR MINIMIZING A SUM OF NORMS.

(1) Define ω_i^μ, $\boldsymbol{E}_\mu$, and $\boldsymbol{F}_\mu$ by (12)–(13).
(2) Define $\boldsymbol{H}_\mu$ by either $\boldsymbol{E}_\mu^{-1}\boldsymbol{F}_\mu$ or $\frac{1}{2}\boldsymbol{E}_\mu^{-1}(\boldsymbol{F}_\mu + \boldsymbol{F}_\mu^T)$ and find either the LU or Cholesky factorization, respectively, of $\boldsymbol{A}\boldsymbol{H}_\mu\boldsymbol{A}^T$. In the former case, quit if the LU factorization generates a zero pivot. In the latter case, either quit if the Cholesky factorization fails, or modify the factorization, effectively redefining $\boldsymbol{H}_\mu$ by a nearby positive definite approximation.
(3) Determine the predictor steps $\Delta\boldsymbol{y}$, $\Delta\boldsymbol{z}$, and $\Delta\boldsymbol{x}$ from (15), (17), and (16).
(4) Define $\tilde{\boldsymbol{x}}$ from the primal scaling rule (see (20) and (21)) or the primal steplength rule (see (22) and (23)), and $\tilde{\boldsymbol{y}}, \tilde{\boldsymbol{z}}$ by the dual line search rule (see (18) and (19)). Quit if the dual line search fails to achieve a reduction in the smoothed dual.
(5) Define $\tilde{\mu}$ by (25) and $\omega_i^{\tilde{\mu}}$ by (28).
(6) Determine the corrector steps $\Delta\boldsymbol{y}$, $\Delta\boldsymbol{z}$, and $\Delta\boldsymbol{x}$ from (33), (17), and (34).
(7) If $(\boldsymbol{E}_{\tilde{\mu}}^{-1}\boldsymbol{z})^T\Delta\boldsymbol{z} < 0$, redetermine $\tilde{\boldsymbol{x}}$, $\tilde{\boldsymbol{y}}$, and $\tilde{\boldsymbol{z}}$ using the primal scaling or steplength rule with $\tilde{\mu}$ instead of μ, and the dual line search rule with $\tilde{\mu}$ instead of μ. Quit if the dual line search rule fails to achieve a reduction in the smoothed dual objective.
(8) Replace $\boldsymbol{x}$, $\boldsymbol{y}$, $\boldsymbol{z}$, and μ by $\tilde{\boldsymbol{x}}$, $\tilde{\boldsymbol{y}}$, $\tilde{\boldsymbol{z}}$, and $\tilde{\mu}$, respectively.
(9) If $\|\boldsymbol{A}\boldsymbol{x}\| > \text{gap}(\boldsymbol{x}, \boldsymbol{z})$, quit. (This cannot happen in exact arithmetic and indicates that rounding errors will dominate any further computation.)
(10) If $\text{gap}(\boldsymbol{x}, \boldsymbol{z}) < \epsilon$ and $\|c - \boldsymbol{A}^T\boldsymbol{y} - \boldsymbol{z}\| + \|A\boldsymbol{x}\| < \epsilon$, quit; otherwise repeat.

There are several ways the algorithm might terminate when rounding errors prevent further progress: breakdown of the factorization, failure in the line search, or growth in the primal infeasibility $\|\boldsymbol{A}\boldsymbol{x}\|$ with respect to the duality gap measure $\text{gap}(\boldsymbol{x}, \boldsymbol{z})$. The occurrence of any of these conditions essentially indicates that the convergence tolerance ϵ is set too small; in any case, when they occur, the current or previous approximation is generally quite accurate.

4. Experiments on small problems. We now report some numerical results for this algorithm, comparing the symmetrized and nonsymmetrized versions and other algorithmic options described above. These results were obtained using a Matlab implementation run on a set of small topologically-constrained Steiner tree test problems (for more details, see [DO98]). The sparsity in the data is determined by the tree structure and its topological constraint, but subject to these qualifications, the data are generated randomly. Each table shows a summary of results from many runs with different random data on the same problem class. Sparsity was not exploited. In all cases $d = 2$. The dual line search was performed using the Matlab **fmin** function with its default tolerance. The machine used was a Sparc Ultra with IEEE double precision arithmetic.

The tables show, for various cases, the number of iterations, the final values of $\text{gap}(\boldsymbol{x}, \boldsymbol{z})$ (defined in (26)), and the infeasibility norm sum $\|\boldsymbol{A}\boldsymbol{x}\| + \|\boldsymbol{c} - \boldsymbol{A}^T\boldsymbol{y} - \boldsymbol{z}\|$, each as medians over a set of randomly generated problems in a given class. The termination tolerance ϵ was set to 10^{-10}.

In Tables 1 and 2, we consider a class of Steiner tree problems with $n = 50$, $m = 62$, for which strict complementarity holds at the solution, and for which the median number of indices for which $\|\boldsymbol{z}_i\| = 0$ at the optimal solution is 15. We compare the nonsymmetric and symmetrized variants of the algorithm ($\boldsymbol{H}_\mu = \boldsymbol{E}_\mu^{-1}\boldsymbol{F}_\mu$ and $\boldsymbol{H}_\mu = \frac{1}{2}\boldsymbol{E}_\mu^{-1}(\boldsymbol{F}_\mu + \boldsymbol{F}_\mu^T)$, respectively) with two choices for updating the primal variable: the scaling rule and the steplength rule with $\tau = 0.999$. All variants used the

TABLE 1
Summary of results for the scaling rule.

Version	iter (median)	gap (median)	infeas (median)
Not symmetrized	8	$1e-12$	$1e-12$
Symmetrized	8	$1e-06$	$6e-13$
Symmetrized, skew corr	7	$1e-08$	$7e-13$
Symmetrized, mod Chol	15	$5e-11$	$1e-12$
Symmetrized, skew corr, mod Chol	9	$2e-11$	$4e-12$

TABLE 2
Summary of results for the steplength rule.

Version	iter (median)	gap (median)	infeas (median)
Not symmetrized	9	$4e-12$	$6e-13$
Symmetrized	11	$8e-08$	$2e-13$
Symmetrized, skew corr	9	$4e-09$	$1e-14$
Symmetrized, mod Chol	15	$4e-11$	$8e-13$
Symmetrized, skew corr, mod Chol	10	$1e-11$	$4e-13$

dual line search rule. For the symmetrized algorithm, we tested both a version which quits if the Cholesky factorization of $\boldsymbol{H}_\mu$ fails and one that modifies the factorization and continues iterating: the latter is standard practice in LP [Wri97, p. 219]. We also tested a variant of the symmetrized version which omits the skew correction $\boldsymbol{h}_i^{(3)}$. Finally, we also tested the effect of omitting the correction $\boldsymbol{h}_i^{(2)}$, but this had essentially no effect in any case.

Table 1 shows the results for the primal scaling rule and Table 2 shows the results using the primal steplength rule. The notations "skew corr" and "mod Chol" refer to the use of the skew correction term and the modified Cholesky factorization, respectively.

The results clearly confirm three remarkable properties of primal-dual predictor-corrector algorithms now well known for linear programming:

- Robust convergence to an optimal solution in all cases tested.
- Rapid local convergence so a consistently small number of iterations is required despite the demand for high accuracy.
- Highly accurate solutions achieved despite the extremely ill-conditioned linear systems being solved toward the end of the solution process.

We now comment in more detail on the results in Tables 1 and 2. First, notice the high accuracy achieved by the nonsymmetric version of the algorithm; the symmetrized version without the modified Cholesky factorization cannot reach the same level of accuracy. With modified Cholesky, high accuracy is achievable, but more iterations are required. The inclusion of the skew correction term $\boldsymbol{h}_i^{(3)}$ substantially improves the performance of the symmetrized version of the algorithm whether or not the Cholesky factorization is modified.

For both the nonsymmetric and symmetrized versions the primal scaling rule has a slightly lower iteration count than the primal steplength rule, apparently because this version of the algorithm has a somewhat faster local convergence rate. However, we note that the scaling rule has the significant disadvantage that it is not applicable if nonhomogeneous linear constraints are added to the problem.

In Tables 3 through 6 we display results for a different class of topologically constrained Steiner tree examples based on the Chung–Graham ladder problem (see

TABLE 3
Results for the scaling rule on the 20 *Chung–Graham ladder problems with strictly complementary solutions.*

Version	iter (median)	gap (median)	infeas (median)
Not symmetrized	7	$2e-13$	$7e-15$
Symmetrized	16	$6e-11$	$7e-15$
Symmetrized, skew corr	10	$2e-11$	$8e-15$
Symmetrized, mod Chol	16	$6e-11$	$7e-15$
Symmetrized, skew corr, mod Chol	10	$2e-11$	$8e-15$

TABLE 4
Results for the steplength rule on the 20 *Chung–Graham ladder problems with strictly complementary solutions.*

Version	iter (median)	gap (median)	infeas (median)
Not symmetrized	9	$2e-12$	$7e-15$
Symmetrized	16	$6e-11$	$8e-15$
Symmetrized, skew corr	10	$7e-12$	$8e-15$
Symmetrized, mod Chol	16	$6e-11$	$8e-15$
Symmetrized, skew corr, mod Chol	10	$7e-12$	$8e-15$

[DO98]). For these examples, $n = 85$, $m = 84$, and the median number of indices for which $\|\boldsymbol{z}_i\|$ equals 0 at the optimal solution is 10. For most of these problems, no strictly complementary (SC) solution exists. (Recall from section 2 that a solution is said to be strictly complementary if, for each i, exactly one of the two conditions $\|\boldsymbol{z}_i\| = 0$ and $\|\boldsymbol{x}_i\| = 1$ holds.)

Tables 3 and 4 show results for the 20 cases out of 200 generated where an SC solution is found (using the primal scaling and primal steplength rules, respectively), while Tables 5 and 6 show results for the other 180 cases where no SC solution is found, presumably because such a solution does not exist. The algorithm achieves the same accuracy (by the duality gap and feasibility measures) on the SC and non-SC problems, but the iteration count is markedly higher in the non-SC case, and the rate of convergence of the algorithm was observed to be slower in the non-SC case. For the non-SC problems, the residuals $\|\boldsymbol{z}_i\|$ are not reduced nearly as close to zero for indices i for which SC does not hold. The reason for this is that the duality gap tolerance requires the products $\boldsymbol{z}_i^T(\boldsymbol{z}_i/\|\boldsymbol{z}_i\| - \boldsymbol{x}_i)$ to be small and *both* factors in the product for such an index i converge to zero as the solution is approached.

For these problems, the modified Cholesky factorization is not needed: the results are identical whether or not it is used.

On the basis of the experiments reported in this section, we recommend the symmetrized version of the algorithm with the skew correction term and the modified Cholesky factorization, using the dual line search and either the primal scaling or the primal steplength rule. The choice of the symmetrized version is based on the substantial advantage of being able to use the Cholesky factorization instead of the LU factorization.

5. Large sparse problems arising in plastic collapse analysis. A variant of the algorithm described above has been used to solve some challenging large sparse problems arising in plastic collapse analysis. We used a symmetrized version of the algorithm, with $\boldsymbol{H}_\mu = \frac{1}{2}\boldsymbol{E}_\mu^{-1}(\boldsymbol{F}_\mu + \boldsymbol{F}_\mu^T)$, so that the Schur complement $\boldsymbol{A}\boldsymbol{H}_\mu\boldsymbol{A}^T$ can be factored by Cholesky decomposition, modified to ensure positive definiteness as

TABLE 5
Results for the scaling rule on the 180 *Chung–Graham ladder problems with NO strictly complementary solution.*

Version	iter (median)	gap (median)	infeas (median)
Not symmetrized	14	$3e-11$	$8e-15$
Symmetrized	16	$5e-11$	$8e-15$
Symmetrized, skew corr	14	$4e-11$	$8e-15$
Symmetrized, mod Chol	16	$5e-11$	$8e-15$
Symmetrized, skew corr, mod Chol	14	$4e-11$	$8e-15$

TABLE 6
Results for the steplength rule on the 180 *Chung–Graham ladder problems with NO strictly complementary solution.*

Version	iter (median)	gap (median)	infeas (median)
Not symmetrized	16	$3e-11$	$8e-15$
Symmetrized	22	$5e-11$	$8e-15$
Symmetrized, skew corr	17	$3e-11$	$8e-15$
Symmetrized, mod Chol	22	$5e-11$	$8e-15$
Symmetrized, skew corr, mod Chol	17	$3e-11$	$8e-15$

discussed earlier. This is the primary cost of the algorithm. Details of this and other numerical linear algebra issues are available in [AA97,AY97,And96a]. The primal steplength rule was used, with $\tau = 0.99$ in (22).

This sparse implementation was developed over several years with large-scale applications in mind. There are two primary differences from the algorithm discussed in section 3. The first is that a different generalization of Mehrotra's method was used, based on differentiating a form of the centering condition which incorporates the symmetrization of $\boldsymbol{F}_\mu$ directly, and therefore does not require a skew correction term. The second is that individual centering parameters were used instead of one parameter, namely,

$$\tilde{\mu}_i = \begin{cases} \tilde{\mu} & \text{if} \quad 0.25 < \left(1 - \|\tilde{\boldsymbol{x}}_i\|^2\right), \\ \tilde{\mu}\left(1 - \|\tilde{\boldsymbol{x}}_i\|^2\right)^{-\frac{1}{2}} & \text{if} \quad \mu \le \left(1 - \|\tilde{\boldsymbol{x}}_i\|^2\right) \le 0.25, \\ \sqrt{\tilde{\mu}} & \text{if} \quad \left(1 - \|\tilde{\boldsymbol{x}}_i\|^2\right) < \mu \end{cases}$$

for $i = 1, \ldots, n$. This modification was found to give significant improvements in performance for the large-scale problems.

The first three classes of test problems are taken from [And96b], where a primal barrier method was used. We are unaware of any other published results for large sparse problems of the form (D). These test problems are finite-dimensional discretizations of collapse problems in rigid plasticity. The discretization step and the physical interpretation of the results can be found in [Chr96,ACO98]. The discrete optimization problems are the same as in [And96b]. The $m \times dn$ matrix $\boldsymbol{A}$ is a typical finite element matrix which in plastic analysis is not square since the equilibrium equation for the continuum is underdetermined. As earlier, $\boldsymbol{H}_\mu$ is block diagonal with block size $d \times d$. In the cases reported here d is either 2 or 3. The runs were made on the same Convex 3240 vector machine (using IEEE-compatible double precision) as in [And96b] so comparisons of accuracy and CPU time are meaningful.

In Tables 7–11, n and m specify the problem dimensions, while $|\boldsymbol{A}|$, $|\boldsymbol{A}\boldsymbol{H}_\mu\boldsymbol{A}^T|$, and $|\boldsymbol{L}|$, respectively, denote the number of nonzeros in $\boldsymbol{A}$, the upper triangle of $\boldsymbol{A}\boldsymbol{H}_\mu\boldsymbol{A}^T$, and the Cholesky factor $\boldsymbol{L}$ of $\boldsymbol{A}\boldsymbol{H}_\mu\boldsymbol{A}^T$. These numbers are the same as in [And96b], except for small variations in sparsity due to improvements in the implementation. The iteration count is denoted by "iter" and "cpu" is the CPU time in seconds. The heading "$\|\boldsymbol{z}_i\| = 0$" indicates the number of norms in the dual objective that are zero at the optimal solution. More precisely, $\|\boldsymbol{z}_i\|$ is interpreted as being zero if it is less than the tolerance 10^{-10}. The heading "relgap" denotes the relative duality gap

$$\frac{\left|\sum_{i=1}^n \|\boldsymbol{z}_i\| - \boldsymbol{c}^T\boldsymbol{x}\right|}{\sum_{i=1}^n \|\boldsymbol{z}_i\| + 1}.$$

In addition to being scaled this expression is the left-hand side of (3) rather than the right-hand side used in (26): if $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{0}$ exactly we have

$$\boldsymbol{c}^T\boldsymbol{x} = (\boldsymbol{A}^T\boldsymbol{y} + \boldsymbol{z})^T\boldsymbol{x} = \boldsymbol{x}^T\boldsymbol{z}.$$

Hence the difference is dominated by the primal infeasibility $\|\boldsymbol{A}\boldsymbol{x}\|$ indicated in the last column.

TABLE 7
Problem and solution characteristics for sspN.

N	n	m	$\|\boldsymbol{A}\|$	$\|\boldsymbol{A}\boldsymbol{H}_\mu\boldsymbol{A}^T\|$	$\|\boldsymbol{L}\|$
4	25	15	224	97	105
10	121	99	1760	1010	1785
50	2601	2499	48800	31010	171083
100	10201	9999	197600	127010	929515
300	90601	89999	1792800	1161010	13203975
400	160801	159999	3190400	2068010	31011299

N	iter	cpu	$\|\boldsymbol{z}_i\| = 0$	relgap	$\|\boldsymbol{A}\boldsymbol{x}\|$
4	7	0	0	$2e-10$	$2e-14$
10	8	1	0	$2e-10$	$7e-15$
50	9	65	0	$2e-09$	$4e-13$
100	10	354	0	$1e-09$	$9e-14$
300	11	6709	0	$1e-09$	$2e-13$
400	12	22139	0	$2e-10$	$2e-13$

Table 7 summarizes the results for the first set of problems, denoted ssp$N$ (simply supported plate with a point load solved on an $N \times N$ grid). They all have the same structure but vary in size, depending on the grid in the finite element analysis. In this problem $d = 3$. This set of problems is characterized by having no zero norms in the solution, i.e., they are, in fact, smooth optimization problems. In [And96b] the constraints $\|\boldsymbol{x}_i\| \leq 1$ are satisfied within a tolerance of order 10^{-9}. These constraints are satisfied exactly in the primal-dual method. Except for this small improvement in accuracy, the primal-dual method shows no significant difference, for these problems, compared to the primal barrier method in [And96b]. There is a small reduction in the iteration count but not in the CPU time. This is a consequence of the fact that these problems are smooth. We shall see below that the primal-dual method handles the presence of zero norms more efficiently than the primal method.

For this problem, as well as for the other results reported below, there is no significant difference in the final duality gap and primal infeasibility achieved by the two algorithms. They are, in all cases, about 10^{-8} or less.

The second set of problems, denoted by $lNa13$, arises in the plane strain model in plasticity. Again N indicates the grid size. In these problems $d = 2$. Characteristics and results are given in Table 8.

TABLE 8
Problem and solution characteristics for $lNa13$.

N	n	m	$\lvert\boldsymbol{A}\rvert$	$\lvert\boldsymbol{A}\boldsymbol{H}_\mu\boldsymbol{A}^T\rvert$	$\lvert\boldsymbol{L}\rvert$
3	49	52	1390	1142	1207
12	625	640	21331	26406	57421
21	1849	1876	64762	84384	278691
30	3721	3760	131683	175086	726046
60	14641	14720	524403	713766	4337857
99	39601	39732	1425134	1957631	14693151
120	58081	58240	2092843	2881926	24202413

N	iter	cpu	$\lVert\boldsymbol{z}_i\rVert = 0$	relgap	$\lVert\boldsymbol{A}\boldsymbol{x}\rVert$
3	14	1	12	$5e-10$	$2e-12$
12	19	34	179	$2e-09$	$4e-12$
21	24	200	958	$4e-10$	$4e-12$
30	24	461	2315	$7e-09$	$3e-11$
60	28	4710	11265	$4e-09$	$1e-10$
99	30	17937	33503	$7e-09$	$9e-10$
120	34	44144	50548	$8e-09$	$4e-10$

In this problem set, the number of zero norms varies from 25 percent for $N = 3$ to 87 percent for $N = 120$. Compared with the primal barrier method in [And96b] there is a significant reduction in the number of iterations and in CPU time. This is shown in Table 9. The primal-dual algorithm also obtains significantly more zero norms in the optimal solution. From our physical understanding of the solution we believe this is correct. It is one of several indications that the primal-dual method is more accurate than the primal barrier method.

TABLE 9
Comparison of the primal barrier method and the primal-dual method for $lNa13$.

		Primal barrier			Primal-dual		
N	n	$\lVert\boldsymbol{z}_i\rVert = 0$	iter	cpu	$\lVert\boldsymbol{z}_i\rVert = 0$	iter	cpu
3	49	0	26	2	12	14	1
12	625	179	33	68	179	19	34
30	3721	1756	52	1313	2315	24	461
60	14641	9843	80	15246	11265	28	4710
120	58081	47602	176	284378	50548	34	44144

There is an important physical interpretation of the complementarity condition (4) in the plasticity problems considered in this section: the vectors $\boldsymbol{z}_i$ represent the deformation (strain) tensor at discrete points in the continuum while the $\boldsymbol{x}_i$ represent the stresses. Thus, if there is any deformation at a point, then the stresses at that point are on their bounds (have norm one) and their directions are determined by the complementarity condition. With this interpretation the complementarity condition

is the so-called "flow rule" for the material.

In the third set of test problems, only a small part of the material undergoes deformation; therefore a very large number of the norms are expected to be zero in the optimal solution. As shown in Table 10, the number of zero norms varies from 62 to 96 percent of the total number of terms. Comparison with [And96b] confirms the observations from Table 9: for the primal-dual method the iteration count is significantly reduced and increases very slowly with the problem size. The CPU time is reduced by a factor 4 or more, and we are able to solve larger instances of the problem.

TABLE 10
Problem and solution characteristics for lNa20.

N	n	m	$\lvert\boldsymbol{A}\rvert$	$\lvert\boldsymbol{A}\boldsymbol{H}_\mu\boldsymbol{A}^T\rvert$	$\lvert\boldsymbol{L}\rvert$
20	1681	1718	59054	76945	246012
40	6561	6636	234145	315504	1511239
60	14641	14754	525236	715653	4433304
80	25921	26072	932327	1277402	8645485
120	58081	58308	2094509	2885699	24240011

N	iter	cpu	$\lVert\boldsymbol{z}_i\rVert = 0$	relgap	$\lVert\boldsymbol{A}\boldsymbol{x}\rVert$
20	20	134	1224	$9e-09$	$4e-11$
40	32	1416	6156	$6e-09$	$3e-11$
60	32	4937	14181	$7e-09$	$5e-11$
80	32	12133	25359	$8e-09$	$4e-11$
120	29	36516	57127	$7e-09$	$3e-11$

The last class of test problems is taken from [AC98]. These are problems of the form (D) with additional linear equality constraints:

$$\min\left\{\sum_{i=1}^{n}\lVert\boldsymbol{z}_i\rVert\,,\ \text{such that } \boldsymbol{A}_i^T\boldsymbol{y}+\boldsymbol{z}_i=\boldsymbol{c}_i, i=1,\ldots,n,\ \text{ and } \boldsymbol{E}^T\boldsymbol{y}=\boldsymbol{d}\right\}, \tag{35}$$

where $\boldsymbol{E}\in\Re^{m\times l}$ and $\boldsymbol{d}\in\Re^l$, i.e., l is the number of linear constraints. In [AC98], it is shown how the ℓ_1 penalty function approach makes it possible to transform the linearly constrained problem to the unconstrained form (D) in section 1, and the physical interpretation and setup of the test problems are described. This class of problems is denoted cl$N$13.

Characteristics and results for these constrained problems are seen in Table 11. In addition to the number l of linear constraints, there is a new column, "constr", indicating the relative infeasibility of these constraints measured by the expression

$$\frac{\left\lVert\boldsymbol{E}^T\boldsymbol{y}-\boldsymbol{d}\right\rVert}{\lVert\boldsymbol{d}\rVert+1}.$$

For the primal barrier method in [AC98], the number of iterations varies from 30 (for $N=3$ and $N=12$) to 201 (for $N=300$). For the primal-dual method the variation is from 11 (for $N=3$) to 24 (for $N=201$) and 35 (for $N=399$). For the case $N=201$ the CPU time is reduced from 36,371 seconds in [AC98] to 6,179 using the primal-dual method. However, we can do even better: in the cl$N$13 problems there is one column that is relatively dense, resulting in considerable fill-in during the

TABLE 11
Problem and solution characteristics for clN13.

N	n	m	l	$\lvert\boldsymbol{A}\rvert$	$\lvert\boldsymbol{A}\boldsymbol{H}_\mu\boldsymbol{A}^T\rvert$	$\lvert\boldsymbol{L}\rvert$
3	9	26	10	163	168	198
12	144	320	145	2818	2790	7402
30	900	1880	901	17848	17511	78128
60	3600	7360	3601	71698	70126	413267
99	9801	19866	9802	195523	191000	1859559
120	14400	29120	14401	287398	280656	2047512
201	40401	81338	40402	807013	787580	9919534
399	159201	319466	159202	3182023	3103949	30001436

N	iter	cpu	$\lVert\boldsymbol{z}_i\rVert = 0$	relgap	$\lVert\boldsymbol{A}\boldsymbol{x}\rVert$	constr
3	11	0	1	$1.6e-08$	$4.4e-09$	$1.0e-15$
12	13	4	95	$6.7e-09$	$2.1e-13$	$1.1e-13$
30	16	31	651	$4.0e-09$	$2.3e-13$	$3.4e-12$
60	20	180	2878	$7.4e-09$	$1.3e-13$	$7.8e-11$
99	24	890	8234	$1.2e-08$	$1.0e-13$	$7.0e-13$
120	25	1238	12311	$1.2e-08$	$1.0e-11$	$2.0e-13$
201	24	6179	35803	$3.1e-08$	$1.0e-13$	$5.1e-13$
399*	35	34776	146326	$7.8e-14$	$2.0e-13$	$6.3e-13$

factorization. Using the technique described in [And96a] for handling dense columns these problems can be solved more efficiently, making it possible to solve for larger values of N. The asterisk in the table indicates that the result for $N = 399$ was obtained by this method. Using the same technique, the case $N = 201$ required 4,293 CPU seconds, and there were 6,367,553 nonzero elements in the $\boldsymbol{L}$ factor.

We conclude that for nonsmooth problems the primal-dual method is significantly more efficient than the primal barrier method applied in [And96b,ACO98,AC98]. The number of iterations increases slowly with the size of the problem. Finally, the primal-dual method appears to be less vulnerable to ill-conditioning near the optimal solution.

Acknowledgments. During the early work on this paper the first author visited the Scientific Computing and Mathematics program at Stanford University. He wishes to thank in particular Michael Saunders for valuable discussions during this period.

REFERENCES

[AA95] I. ADLER AND F. ALIZADEH, *Primal-Dual Interior Point Algorithms for Convex Quadratically Constrained and Semidefinite Optimization Problems*, Technical Report 46-95, RUTCOR, Rutgers University, New Brunswick, NJ, 1995.

[AA97] E. D. ANDERSEN AND K. D. ANDERSEN, *The APOS Linear Programming Solver: An Implementation of the Homogeneous Algorithm*, Technical Report, CORE Discussion Paper 9730, CORE, Université Catholique de Louvain, Belgium, 1997.

[AC98] K. D. ANDERSEN AND E. CHRISTIANSEN, *Minimizing a sum of norms subject to linear equality constraints*, Comput. Optim. Appl., 11 (1998), pp. 65–79.

[ACK+98] J. ALPERT, T. F. CHAN, A. B. KAHNG, I. L. MARKOV, AND P. MULET, *Faster minimization of linear wirelength for global placement*, IEEE Trans. Computer Aided Design for Integrated Circuits and Systems, 17 (1998), pp. 3–13.

[ACO98] K. D. ANDERSEN, E. CHRISTIANSEN, AND M. L. OVERTON, *Computing limit loads by minimizing a sum of norms*, SIAM J. Sci. Comput., 19 (1998), pp. 1046–1062.

[AM93] J. C. ALEXANDER AND J. H. MADDOCKS, *Bounds on the friction-dominated motion of a pushed object*, Internat. J. Robotics Research, 12 (1993), pp. 231–248.

[And96a] K. D. Andersen, *A modified Schur complement method for handling dense columns in interior-point methods for linear programming*, ACM Trans. Math. Software, 22 (1996), pp. 348–356.

[And96b] K. D. Andersen, *An efficient Newton barrier method for minimizing a sum of Euclidean norms*, SIAM J. Optim., 6 (1996), pp. 74–95.

[AY97] E. D. Andersen and Y. Ye, *On a homogeneous algorithm for a monotone complementarity problem with nonlinear equality constraints*, in Complementarity and Variational Problems: State of the Art, M. C. Ferris and J.-S. Pang, eds., SIAM, Philadelphia, 1997, pp. 1–11.

[BB] D. V. Byrnes and L. E. Bright, *Design of High-Accuracy Multiple Flyby Trajectories Using Constrained Optimization*, AAS Paper 95-307, AAS/AIAA Astrodynamics Specialist Conference, Halifax, NS, Canada, 1995.

[CC80] P. H. Calamai and A. R. Conn, *A stable algorithm for solving the multifacility location problem involving Euclidean distances*, SIAM J. Sci. Statist. Comput., 1 (1980), pp. 512–526.

[CC87] P. H. Calamai and A. R. Conn, *A projected Newton method for l_p norm location problems*, Math. Programming, 38 (1987), pp. 75–109.

[CGM96] T. F. Chan, G. H. Golub, and P. Mulet, *A nonlinear primal dual method for TV-based image restoration*, SIAM J. Sci. Comput., 20 (1999), pp. 1964–1977.

[Chr96] E. Christiansen, *Limit analysis of collapse states*, in Handb. Numer. Anal. IV, P. G. Ciarlet and J. L. Lions, eds., North–Holland, Amsterdam, 1996, pp. 193–312.

[DO98] D. R. Dreyer and M. L. Overton, *Two heuristics for the Euclidean Steiner tree problem*, J. Global Optim., 13 (1998), pp. 95–106.

[IK99] K. Ito and K. Kunisch, *An active set strategy based on the augmented Lagrangian formulation for image restoration*, M^2AN Math. Model. Numer. Anal., 33 (1999), pp. 1–21.

[Kuh67] H. W. Kuhn, *On a pair of dual nonlinear programs*, in Nonlinear Programming, J. Abadie, ed., North–Holland, Amsterdam, 1967, pp. 38–54.

[Kuh91] H. W. Kuhn, *Nonlinear programming: A historical note*, in History of Mathematical Programming, J. K. Lenstra, A. H. G. Rinnooy Kan, and A. Schrijver, eds., North–Holland, Amsterdam, 1991, pp. 82–96.

[Mai87] E. H. Maize, *Linear statistical analysis of maneuver optimization strategies*, in Proceedings of the American Astronautical Society and American Insititute of Aeronautics and Astronautics Astrodynamics Specialist Conference, Kalispell, MT, 1987.

[Meh92] S. Mehrotra, *On the implementation of a primal-dual interior point method*, SIAM J. Optim., 2 (1992), pp. 575–601.

[Mic] J. Michel, *Private Communication*, Marietta College, Marietta, OH, and Jet Propulsion Laboratory, Pasadena, CA.

[NN94] Y. Nesterov and A. Nemirovskii, *Interior Point Polynomial Algorithms in Convex Programming*, SIAM, Philadelphia, 1994.

[NT98a] Y. Nesterov and M. Todd, *Primal-dual interior-point methods for self-scaled cones*, SIAM J. Optim., 8 (1998), pp. 321–364.

[NT98b] Y. Nesterov and M. Todd, *Self-scaled barriers and interior-point methods for convex programming*, Math. Oper. Res., 22 (1998), pp. 1–42.

[Ove83] M. L. Overton, *A quadratically convergent method for minimizing a sum of Euclidean norms*, Math. Programming, 27 (1983), pp. 34–63.

[Ove84] M. L. Overton, *Numerical solution of a model problem from collapse load analysis*, in Computing Methods in Applied Sciences and Engineering VI, J. L. Lions and R. Glowinski, eds., North–Holland, Amsterdam, 1984, pp. 421–437.

[Par91] H. R. Parks, *Numerical approximation of parametric oriented area-minimizing hypersurfaces*, SIAM J. Sci. Statist. Comput., 13 (1992), pp. 499–511.

[Roc70] R. T. Rockafellar, *Convex Analysis*, Princeton University Press, Princeton, NJ, 1970.

[Str79] G. Strang, *A minimax problem in plasticity theory*, in Functional Analysis Methods in Numerical Analysis, Lecture Notes in Math. 701, Springer-Verlag, New York, 1979.

[Wei37] E. Weiszfeld, *Sur le point par lequel la somme des distances de n points donnes est minimum*, Tohoku Math. J., 43 (1937), pp. 355–386.

[Wri97] S. J. Wright, *Primal-Dual Interior-Point Methods*, SIAM, Philadelphia, 1997.

[XLD99a] G. Xue, T. P. Lillys, and D. E. Dougherty, *Computing the minimum cost pipe network interconnecting one sink and many sources*, SIAM J. Optim., 10 (1999), pp. 22–42.

[XLD99b] G. Xue, G.-H. Lin, and D.-Z. Du, *Grade of service minimum Steiner Euclidean trees*, in Proceedings IEEE International Symposium on Circuits and Systems, Orlando, FL, 1999, IEEE Press, Piscataway, NJ.

[XY97] G. Xue and Y. Ye, *An efficient algorithm for minimizing a sum of Euclidean norms with applications*, SIAM J. Optim., 7 (1997), pp. 1017–1036.

SIAM J. SCI. COMPUT.
Vol. 22, No. 1, pp. 263–284
© 2000 Society for Industrial and Applied Mathematics

THE NUMERICAL SOLUTION OF THE STEADY STATE SOLID FUEL IGNITION MODEL AND ITS OPTIMAL CONTROL*

ALFIO BORZÌ† AND KARL KUNISCH†

Abstract. We present a finite difference multigrid technique that efficiently solves optimal control problems associated with the steady state solid fuel ignition model. This is a nonlinear indefinite problem which gives rise to singular optimal control systems. Regarding the direct problem, necessary and sufficient conditions for existence of solutions are given and second-order convergence of numerical solution errors with respect to the mesh size is proven. A robust nonlinear multigrid method is implemented to validate this convergence estimate.

Two optimal control strategies for solid fuel ignition phenomena are considered and compared. The corresponding optimality systems are solved to second-order accuracy by a multigrid method whose convergence properties are independent of the values of the weights in the cost functionals and of the number of grid points.

Key words. steady state solid fuel ignition model, finite difference method, convergence estimates, optimal control problem, nonlinear multigrid method

AMS subject classifications. 49J20, 65N06, 65N12, 65N55

PII. S1064827599360194

1. Introduction.

Modeling of combustion phenomena is an important task in various fields of application. Combustible systems give rise to reaction-diffusion problems where heat production occurs and must be controlled. Control can be required, for example, to avoid blow-up phenomena or to obtain an optimal combustion process.

Our goal is to develop a finite difference multigrid method that accurately and efficiently solves singular optimal control problems for such systems. We further compare two different control formulations with respect to their ability and cost to influence the temperature distribution. The steady state solid fuel ignition model, which can give rise to blow-up phenomena [4, 9, 10, 11, 14, 15, 16, 18, 21, 22, 23], is given by

$$\begin{aligned} \Delta u + \delta \exp(u) &= f \text{ in } \Omega, \\ u &= 0 \text{ on } \partial\Omega. \end{aligned} \tag{1.1}$$

Due to the nonmonotonic sign in front of the exponential term, this nonlinear indefinite problem may admit multiple solutions or no solution at all depending on Ω and on the value of the positive coefficient δ. Actually, the absence of coerciveness represents a major difficulty in the analysis and the solution of this problem.

We present new results concerning existence of solutions of (1.1) and of its discretized counterpart in the presence of control, $f \neq 0$. Our analysis is based on the upper and lower solutions method [10, 16, 22, 23]. We extend a result of Bramble [5] to prove second-order accuracy of the numerical solution with respect to the mesh size.

*Received by the editors August 17, 1999; accepted for publication (in revised form) January 4, 2000; published electronically June 20, 2000. This work was supported in part by the SFB 03 "Optimization and Control."

http://www.siam.org/journals/sisc/22-1/36019.html

†Institut für Mathematik, Karl-Franzens-Universität Graz, Heinrichstraße 36, A-8010 Graz, Austria (alfio.borzi@kfunigraz.ac.at, karl.kunisch@kfunigraz.ac.at).

The nonlinear algebraic problem arising from the discretization of (1.1) is solved by a nonlinear multigrid method [1, 6, 7, 8, 12, 24]. For the smoother, the Gauss–Seidel variant of the Picard iteration considered in the theory of upper and lower solutions method is used. We prove that the sequence of approximations obtained by this iterative scheme is monotone and converges under certain conditions to the solution of the discrete problem.

The optimal control problems we consider are defined by

$$\begin{aligned} &\min_{f\in L^2(\Omega)} J(u(f), f), \\ &\Delta u + \delta \exp(u) = f \text{ in } \Omega, \\ &u = 0 \text{ on } \partial\Omega, \end{aligned} \tag{1.2}$$

where two tracking-type cost functionals are considered. Notice that the optimal control problems associated to the steady state solid fuel ignition model are "singular"; see [17]. Proper functional settings within which the existence of solutions to (1.2) is guaranteed are proposed and analyzed in [14, 15]. However, it is not obvious whether with either of the two formulations a significant degree of control of the state variable can be achieved and which of the two cost functions performs preferably. For this reason, we implement reliable multigrid algorithms for both formulations and give a comparison from the point of view of open loop control.

More precisely, we apply multigrid techniques to the optimality system associated with (1.2). This will allow to solve the optimal control problem in a few multigrid cycles. These multigrid algorithms have contraction factors independent of the number of grid points and of the value of the weights in the cost functionals. For the smoothing process, we show that the Gauss–Seidel–Newton (GSN) method is superior to the Gauss–Seidel–Picard (GSP) approach.

In the next section we derive and analyze the steady state solid fuel ignition model. In section 3 the discretization of this model and its accuracy are considered. In the following, section 4, the nonlinear multigrid method is discussed and numerical experiments with the steady state solid fuel ignition model are presented. In section 5 optimal control problems for this model are formulated. The numerical solution process for the optimal control problem is presented and the results of numerical experiments are discussed.

2. Derivation and analysis of the model. In the following two subsections we derive the steady state solid fuel ignition model and prove necessary and sufficient conditions for the existence of solutions to this model.

2.1. Derivation of the model. Consider a solid explosive material and assume that this material is contained in a bounded two-dimensional domain Ω. Unless otherwise specified it is assumed that $\partial\Omega$ is smooth or that Ω is a quadrilateral. The ignition process is characterized by the appearance of a localized warm region in which heat is produced due to a chemical reaction process. By considering a solid fuel we actually study the pointwise balance between chemically generated heat and heat transfer due to conduction; see, e.g., [4].

We consider a combustion system whose equilibrium state is characterized by a temperature T_0, pressure p_0, density ρ_0, and the oxidant mass fraction y_0. In terms of these reference values the combustion model can be written in a nondimensional form by the transformation $T \to T/T_0$, $p \to p/p_0$, $\rho \to \rho/\rho_0$, and $y \to y/y_0$. The following system of equations describing the combustion of the solid fuel is obtained [4]:

$$\begin{aligned} \partial_t T - \Delta T &= \epsilon\delta y^m \exp\left(\frac{T-1}{\epsilon T}\right), \\ \partial_t y - \beta\Delta y &= -\epsilon\delta\Gamma y^m \exp\left(\frac{T-1}{\epsilon T}\right), \end{aligned} \tag{2.1}$$

with initial and boundary conditions given by

$$\begin{aligned} &T(x,0) = 1, \quad y(x,0) = 1, \quad x \in \Omega, \\ &T(x,t) = 1, \quad \frac{\partial y(x,t)}{\partial \eta} = 0, \quad x \in \partial\Omega, \quad t \in (0,\infty), \end{aligned} \tag{2.2}$$

where β, Γ, and δ are nonnegative constants and $\frac{\partial}{\partial \eta}$ denotes the normal derivative. Further $m = \nu_F + \nu_O$, with ν_F and ν_O stoichiometric constants. The parameter ϵ is given by $\epsilon = \frac{RT_0}{E}$, where R is the gas constant and E is the total energy of the system.

If the following conditions are met,

(I) the heat loss is sufficiently large compared to the energy release (energy equilibrium is established);

(II) the maximum system temperature is never significantly different from the initial value due to the relatively cold boundary,

then a first-order approximation with respect to the changes in T and y of (2.1)–(2.2) is justified. We set

$$T = 1 + \epsilon u, \quad y = 1 - \epsilon c,$$

where u and c are two auxiliary quantities to express temperature- and oxidant mass-fraction changes. If further $\epsilon \ll 1$, the equations in u and c decouple and the following model for u is obtained:

$$\begin{aligned} \partial_t u - \Delta u &= \delta \exp(u) - f \text{ in } \Omega, \quad t \in (0,\infty), \\ u &= 0 \text{ in } \Omega, \qquad t = 0, \\ u &= 0 \text{ on } \partial\Omega, \quad t \in (0,\infty), \end{aligned} \tag{2.3}$$

which is called the *solid fuel ignition model* [4]. Here we introduced a continuous source term function f that will be used to describe the control mechanism.

The equation for c reads $\partial_t c - \beta\Delta c = \delta\Gamma \exp(u)$. It is linear in c and its solution depends on u. We shall not consider this problem any further.

The stationary version of (2.3) is called the *steady state solid fuel ignition model* [4] and it is given by

$$\begin{aligned} \Delta u + \delta \exp(u) &= f \text{ in } \Omega, \\ u &= 0 \text{ on } \partial\Omega. \end{aligned} \tag{2.4}$$

Although this model is obtained from assumptions simplifying the full solid fuel problem it provides many features of this model; see [4] and references therein. The investigation of the steady state model is important for the analysis from the point of view of dynamical systems as well as numerical analysis of the time-dependent problem, which will be considered in a forthcoming paper.

TABLE 1
Numerical estimates for δ^* for a unit square domain.

Reference	δ^*	$u(0.5, 0.5)$
from Jensen's ineq.	7.2616	1.00
[9]	6.8054	1.39043
[9]	6.8081	1.39166
[18]	6.808124	1.39166
[18]	6.8082	1.392

2.2. Analytical results. In this section we determine necessary and sufficient conditions for existence of solutions to the steady state solid fuel ignition problem with control. For the convenience of the reader we recall some analytical results for the case without control,

$$\Delta u + \delta e^u = 0 \text{ in } \Omega, \tag{2.5}$$

with homogeneous Dirichlet boundary conditions in two spatial dimensions. In [10] it is shown that (2.5) can have at most two solutions. If solutions of (2.5) exist, then there exists a unique minimal (global) solution defined by $u \leq v$, where v is any other solution of (2.5) [10]. A necessary condition for the existence of at least one solution is expressed by the following theorem [10].

THEOREM 2.1. *Let λ_0 be the smallest eigenvalue of $-\Delta$ under homogeneous Dirichlet boundary conditions. If $\lambda_0 < \delta e$, then the set of solutions is empty.*

Thus, in order to have at least one solution we must have $\lambda_0 \geq \delta e$. For the proof of Theorem 2.1 one should notice that a solution u of (2.5) must satisfy Jensen's inequality [10]:

$$-\lambda_0 J + \delta e^J \leq 0, \quad J = (\Phi_0, u)_{L^2},$$

where Φ_0 is the eigenfunction of $-\Delta$ corresponding to λ_0, with the normalization $(\Phi_0, 1) = 1$. Further we notice that if $\lambda_0 < \delta e$ Jensen's inequality is not satisfied, henceforth there exists no solution. Moreover, recalling that the eigenvalues of $-\Delta$ on $\Omega = [0, L] \times [0, L]$ are given by $\lambda_{mn} = \pi^2(m^2 + n^2)/L^2$, $m, n = 1, 2, \ldots$, we conclude that the set of solutions becomes empty for Ω sufficiently large.

If Ω is a unit square, then there exists a critical value $\delta^* > 0$ such that [4, 11, 10]:

- If $\delta > \delta^*$, there is no solution.
- If $\delta \in]0, \delta^*]$ ($\delta \in]0, \delta^*[$), then there exists at least one and at most two solution which belong to $W^{2,p}(\Omega)$, $p \geq 1$.
- If $\delta = \delta^*$, there exists one solution $u^* \in H_0^1(\Omega) \cap W^{2,p}(\Omega)$, $p \geq 1$, and δ^* is referred to as the turning point.

For the case of $\Omega = [0, 1] \times [0, 1]$, the turning point has been estimated, based on continuation techniques [9, 11, 18]; see Table 1.

We next consider the inhomogeneous case (2.4) where the existence of solutions depends also on the inhomogeneity f. We can generalize the Jensen inequality to this case:

$$-\lambda_0 J + \delta e^J \leq (f, \Phi_0)_{L^2}, \quad J = (\Phi_0, u)_{L^2}, \tag{2.6}$$

which gives a necessary condition such that the steady state solid fuel ignition model with forcing term can have a solution.

Sufficient conditions for existence of solutions to (2.4) can be obtained by means of the upper and lower solutions method [10, 16, 22, 23]. In order to present our results obtained by this method, let us consider a general nonlinear elliptic problem:

$$\begin{aligned} -\Delta u &= F(u) \text{ in } \Omega, \\ u &= 0 \quad \text{ on } \partial\Omega, \end{aligned} \tag{2.7}$$

where Ω is of class $C^{2+\alpha}$, $\alpha > 0$, and F is Hölder continuous in (x, u). For the case of the solid fuel ignition problem we have $F(x, u) = \delta e^u - f(x)$, where f is a continuous function on $\overline{\Omega}$. We need the following definition.

DEFINITION 2.2. *A function $\tilde{u} \in C^\alpha(\overline{\Omega}) \cap C^2(\Omega)$, $\alpha > 0$, is called upper solution of* (2.7) *if*

$$-\Delta\tilde{u} \geq F(\tilde{u}) \text{ in } \Omega; \quad \tilde{u} = 0 \text{ on } \partial\Omega. \tag{2.8}$$

Similarly, $\hat{u} \in C^\alpha(\overline{\Omega}) \cap C^2(\Omega)$, $\alpha > 0$, is called lower solution if it satisfies the reversed inequality.

A pair of upper and lower solutions is called *ordered* if $\tilde{u} \geq \hat{u}$ in $\overline{\Omega}$. For any pair of ordered upper and lower solutions, we denote by $\langle \tilde{u}, \hat{u} \rangle$ the sector of all functions $u \in C(\overline{\Omega})$ such that $\hat{u} \leq u \leq \tilde{u}$ in $\overline{\Omega}$.

Assume that $F(x, u)$ satisfies a one-sided local Lipschitz condition, i.e., for every $\tilde{u} \geq \hat{u}$ there exists a continuous nonnegative function c in $\overline{\Omega}$ such that

$$F(x, u_1) - F(x, u_2) \geq -c(x)(u_1 - u_2) \text{ for } \hat{u} \leq u_2 \leq u_1 \leq \tilde{u}. \tag{2.9}$$

If $F(u) = \delta \exp(u) - f$ this condition is satisfied with $c(x)$ identically zero. We define a sequence $\{u^{(k)}\}$ by the following iteration process:

$$-\Delta u^{(k)} = F(u^{(k-1)}) \text{ in } \Omega; \quad u^{(k)} = 0 \text{ on } \partial\Omega. \tag{2.10}$$

Upper and lower sequences $\{\overline{u}^{(k)}\}$ and $\{\underline{u}^{(k)}\}$, corresponding to initial conditions $u^{(0)} = \tilde{u}$ and $u^{(0)} = \hat{u}$, with $\tilde{u}$ and $\hat{u}$ upper and lower solutions, respectively, are of special interest. The following properties hold. (See, e.g., [22].)

LEMMA 2.3. *Upper and lower sequences are well defined and possess the monotone property,*

$$\hat{u} \leq \underline{u}^{(k)} \leq \underline{u}^{(k+1)} \leq \overline{u}^{(k+1)} \leq \overline{u}^{(k)} \leq \tilde{u} \text{ in } \overline{\Omega}$$

for every k.

The next theorem follows from results in [22].

THEOREM 2.4. *Let $\tilde{u}$ and $\hat{u}$ be ordered upper and lower solutions of* (2.7)*, and let F be Hölder continuous in (x, u) and satisfy the Lipschitz condition* (2.9)*. Then $\{\overline{u}^{(k)}\}$ converges monotonically from above to a solution $\bar{u}$, and $\{\underline{u}^{(k)}\}$ converges monotonically from below to a solution $\underline{u}$, and both solutions of* (2.7) *belong to $C^{2+\alpha}(\overline{\Omega})$, $\alpha > 0$. Moreover, $\underline{u} \leq \bar{u}$.*

To assert the existence of one solution to (2.4) it remains to construct an ordered pair of upper and lower solutions. To construct a lower solution choose $\hat{u} = \psi$, with ψ the solution of the Poisson problem, $\Delta\psi = f$, with homogeneous Dirichlet boundary conditions and $f \in C(\overline{\Omega})$. In fact,

$$-\Delta\psi = -f < \delta e^{\hat{u}} - f = F(\hat{u}),$$

as desired.

We next construct an upper solution to (2.4) for δ sufficiently small. For this purpose we express $f = f^+ - f^-$ with f^+ and f^- nonnegative functions defined by $f^+ = \max_\Omega(f, 0)$ and $f^- = \max_\Omega(-f, 0)$. Consider the linear problem

$$-\Delta\omega = c\omega + 1 + f^- \text{ in } \Omega; \quad \omega = 0 \text{ on } \partial\Omega, \tag{2.11}$$

where $0 < c < \lambda_0$ with λ_0 the smallest positive eigenvalue of $-\Delta$ with homogeneous Dirichlet boundary conditions. From the Fredholm alternative and the maximum principle it follows that (2.11) has a unique solution $\omega^* > 0$ in Ω. Then $\tilde{u} = \omega^*$ is an upper solution to (2.4), provided that δ is sufficiently small. In fact we have

$$-\Delta\omega^* = c\omega^* + 1 + f^- \geq \delta e^{\omega^*} + f^- \geq \delta e^{\tilde{u}} - (f^+ - f^-) \geq \delta e^{\tilde{u}} - f = F(\tilde{u})$$

for all sufficiently small $\delta > 0$.

Finally notice that $\tilde{u} \geq \hat{u}$. In fact, since $\tilde{u}$ and f^+ are positive functions in Ω it follows from $-\Delta(\tilde{u} - \hat{u}) = c\tilde{u} + 1 + f^+$ that $\tilde{u} - \hat{u} \geq 0$. That is, $\tilde{u}$ and $\hat{u}$ are ordered upper and lower solutions of (2.4) and the existence of solutions to (2.4) follows from Theorem 2.4.

3. The numerical solution of the steady state solid fuel ignition model. In the next two sections we define a finite difference multigrid algorithm for the steady state solid fuel ignition model. We prove that the solution of the nonlinear indefinite discrete problem provides second-order accurate solutions. In order to validate this estimate numerically we solve the problem by using the multigrid scheme. This solver is optimal in terms of number of computer operations, and it is robust in the sense that it reliably computes solutions very close to the turning point.

3.1. The discretization of the steady state solid fuel ignition problem. We commence by describing the finite difference approximation [19] of the steady state solid fuel ignition problem on $\Omega = [0, 1] \times [0, 1]$. Let Ω_{h_k}, $k = 1, 2, \ldots, M$, denote a sequence of uniform grids. The integer k is called the level number. The coarsest grid corresponds to $k = 1$ with mesh size $h_1 = 1/4$, and the finest corresponds to $k = M$ with mesh size $h_M = h_1/2^{(M-1)}$. The set of mesh points is defined by (x_i, y_j), $x_i = (i-1)h_k$ and $y_j = (j-1)h_k$, $i, j = 1, \ldots, N_k$, where $N_k = 2^{(k+1)}+1$. A numerical function on Ω_{h_k} will be denoted by u^{h_k}. To every continuous function u on Ω there exists an associated numerical function defined by its pointwise restriction to the grid points of Ω_{h_k} denoted by $u(x_i, y_j)$, or u_{ij} for short.

The five-point stencil finite difference discretization of (2.4) on Ω_h is denoted by

$$\begin{aligned} \Delta^h u^h + \delta \exp(u^h) &= f^h \text{ in } \Omega_h, \\ u^h &= 0 \text{ on } \partial\Omega_h. \end{aligned} \tag{3.1}$$

The methods used in the previous section to prove necessary and sufficient conditions for existence of solutions to the steady state solid fuel ignition model can be extended to the present discrete case. In particular, a necessary condition for existence of solutions is given by the inequality (2.6) with λ_0 the smallest eigenvalue of $-\Delta^h$ with homogeneous Dirichlet boundary conditions.

A sufficient condition for existence of solutions to (3.1) can be obtained by adaptation of the upper and lower solutions method to the discrete system of finite difference equations which correspond to (3.1) [21]. A discrete function $\tilde{u}^h$ ($\hat{u}^h$) is called an upper (lower) solution of (3.1) if $-\Delta^h\tilde{u}^h \geq F^h(\tilde{u}^h)$ ($-\Delta^h\hat{u}^h \leq F^h(\hat{u}^h)$), with

$F^h(u^h) = \delta \exp(u^h) - f^h$. Utilizing the upper and lower solutions method of the previous section we can construct an ordered pair of upper and lower solutions for the discretized problem (3.1). The existence of a solution to (3.1) follows from a modification of Theorem 2.4 to the discrete case. (Compare Theorem 3.3 of [21].)

3.2. Accuracy of the numerical approximation. The new result of this section is the second-order accuracy of the solution to (3.1). The linear indefinite case was considered in [5]. We extend those results to the present nonlinear indefinite case.

Let u be the minimal solution to the steady state solid fuel ignition model. We have the following lemma.

LEMMA 3.1. *Let $u \in C^4(\overline{\Omega})$ be one bounded solution to the steady state solid fuel ignition model* (2.4) *and let u^h satisfy* (3.1), *with $\lim_{h\to 0} ||u - u^h||_0 = 0$, where $||\cdot||_0$ denotes the maximum norm on Ω_h. Then for sufficiently small mesh sizes h,*

$$||u - u^h||_0 = O(h^2).$$

To prove this lemma notice that $u \in C^4(\overline{\Omega})$ satisfies

$$\begin{aligned} \Delta^h u + \delta \exp(u) &= f^h + T^h(u) \text{ in } \Omega_h, \\ u &= 0 \text{ on } \partial\Omega_h, \end{aligned} \tag{3.2}$$

where $T^h(u)$ is the local truncation error given by

$$T^h(u_{ij}) = \frac{1}{12}h^2 \left(\frac{\partial^4 u}{\partial x^4} + \frac{\partial^4 u}{\partial y^4} \right) + O(h^4).$$

For every mesh size h define a function $v \in C^4(\overline{\Omega})$, with $v = 0$ on $\partial\Omega$, such that its restriction v^h to Ω_h satisfies $v^h = u - u^h$. For these functions one can use the following estimate due to Bramble [5]:

$$||v^h||_0 \le C||\Delta^h v^h + \kappa v^h||_0, \tag{3.3}$$

where C is independent of h and κ is a positive function satisfying

$$|\kappa|_{\overline{\Omega}} \equiv \max_{\overline{\Omega}} |\kappa(x)| \le \bar{\kappa},$$

where $\bar{\kappa}$ must not coincide with an eigenvalue λ of the problem: $\Delta W + (\kappa - \bar{\kappa})W + \lambda W = 0$ in Ω, $W = 0$ on $\partial\Omega$. We apply Bramble's estimate with $\kappa = \delta e^u$. Using Taylor's expansion for e^{u^h} around u, and the fact that $\Delta^h u^h + \delta e^{u^h} = f^h$, we obtain

$$||v^h||_0 \le C||\Delta^h v^h + \delta e^u v^h||_0 = C||\Delta^h u - \Delta^h u^h + \delta e^u (u - u^h)||_0$$

$$= C||\Delta^h u - \Delta^h u^h + \delta e^u - \delta e^{u^h} + \delta \sum_{n=0}^{\infty} (-1)^n \frac{(u - u^h)^{n+2}}{(n+2)!} e^u||_0$$

$$\le C||\Delta^h u + \delta e^u - f^h||_0 + \delta||u - u^h||_0^2 \sum_{n=0} \frac{||u - u^h||_0^n}{(n+2)!}||e^u||_0.$$

Utilizing (3.2) we find

$$||u - u^h||_0 \le C||T^h(u)||_0 + B_h\delta||e^u||_0||u - u^h||_0^2 \le C\frac{1}{6}h^2 M_4 + B_h\bar{\kappa}||u - u^h||_0^2,$$

where

$$B_h = \sum_{n=0} \frac{||u-u^h||_0^n}{(n+2)!} \text{ and } M_4 = \max\left(\max_{\Omega}\left|\frac{\partial^4 u}{\partial x^4}\right|, \max_{\Omega}\left|\frac{\partial^4 u}{\partial x^4}\right|\right).$$

Since by assumption $\lim_{h\to 0}||u-u^h||_0 = 0$ there exists an h_0 sufficiently small such that $1 - B_h\bar{\kappa}||u-u^h||_0 > 0$ for any $h \le h_0$. This implies the estimate

$$0 \le (1 - B_h\bar{\kappa}||u-u^h||_0)||u-u^h||_0 \le C\frac{1}{6}h^2 M_4, \quad h \le h_0.$$

Consequently, $||u-u^h||_0 = O(h^2)$ for h sufficiently small as desired.

The estimate of Lemma 3.1 will be demonstrated by numerical experiments. We shall show that as the mesh is refined according to $h \to h' = h/2$, the norm of the solution error reduces with a factor of 4.

4. A multigrid method for the steady state solid fuel ignition problem. After proving existence of the numerical approximation and its degree of accuracy we now construct a multigrid algorithm that computes these discrete solutions in an efficient way, i.e., with a contraction number which is much smaller than one and independent of the mesh size. By this method we are able to validate the second-order accuracy estimate given above.

The approach we use is the full approximation scheme of Brandt [6] which we describe in the following sections. As a smoother for problems of the type considered here the GSP iteration is choosen; see, e.g., [18]. Convergence for the GSP scheme will be proved. This relaxation process is then accelerated by multigrid techniques.

4.1. The GSP iterative method. A monotone sequence $\{u^{(h,m)}\}$ of approximations to (3.1) can be obtained by repeated application of the Picard iteration:

$$-\Delta^h u^{(h,m)} = F^h(u^{(h,m-1)}) \text{ in } \Omega_h, \quad u^{(h,m)} = 0 \text{ on } \partial\Omega_h. \tag{4.1}$$

This iteration has the drawback of requiring the solution to a Poisson problem at each step. We shall therefore consider a Gauss–Seidel version that does not require the exact solution of the discrete Poisson problem, but nevertheless provides a monotone sequence.

Consider the standard regular splitting $A^h = D^h - (L+U)^h$, where A^h represents the matrix of coefficients for $-\Delta^h$ with Dirichlet boundary conditions.

A Gauss–Seidel iterative method for $A^h u^h = F^h$ can be expressed as $u^{(h,m)} = [I - (D^h - L^h)^{-1}A^h]u^{(h,m-1)} + (D^h - L^h)^{-1}F^h$. That is,

$$u^{(h,m)} = u^{(h,m-1)} - (D^h - L^h)^{-1}[-\Delta^h u^{(h,m-1)} - F(u^{(h,m-1)})]. \tag{4.2}$$

This iteration has the following property.

LEMMA 4.1. *The repeated application of the GSP iteration* (4.2) *with lower (upper) solution as initial value defines a monotonically increasing (decreasing) sequence* $\{u^{(h,m)}\}$ *of lower (upper) solutions.*

To prove this lemma let us consider the case of a lower solution as initial approximation. (The case of an upper solution can be proven the same way.) Observe that $(D^h - L^h)^{-1}$ is positive definite. (See, e.g., [25].) This implies that if $u^{(h,m-1)}$ is a lower solution, that is, $-\Delta^h u^{(h,m-1)} - F(u^{(h,m-1)}) \le 0$, then $u^{(h,m)} \ge u^{(h,m-1)}$ and the sequence is monotonic increasing.

To prove that $u^{(h,m)}$ is also a lower solution let us apply $(D^h - L^h)$ from the left to (4.2) to obtain

$$(D^h - L^h)u^{(h,m)} = (D^h - L^h)u^{(h,m-1)} - [A^h u^{(h,m-1)} - F(u^{(h,m-1)})].$$

Notice that all elements of U^h are positive. Then add and subtract $-U^h u^{(h,m)}$ on the left-hand side and $-U^h u^{(h,m-1)}$ on the right-hand side to obtain

$$\begin{aligned}(D^h - L^h - U^h)u^{(h,m)} + U^h u^{(h,m)} &= (D^h - L^h - U^h)u^{(h,m-1)} + U^h u^{(h,m-1)} \\ &\quad - [A^h u^{(h,m-1)} - F(u^{(h,m-1)})],\end{aligned}$$

$$A^h u^{(h,m)} = -U^h(u^{(h,m)} - u^{(h,m-1)}) + F(u^{(h,m-1)}),$$

$$A^h u^{(h,m)} \le F(u^{(h,m-1)}) \le F(u^{(h,m)}),$$

and therefore $-\Delta^h u^{(h,m)} - F(u^{(h,m)}) \le 0$, which proves Lemma 4.1.

Since the GSP iteration is slow it cannot be recommended for the solution of (3.1). An efficient solver is obtained by implementing a multigrid method where the GSP iteration acts as a smoother. By using the GSP scheme we avoid the difficulties reported in [9, 18] to define a robust multigrid smoother to solve the present problem.

4.2. The multigrid method for nonlinear problems. We construct a multigrid algorithm by combining the GSP iteration used as the smoother and the coarse grid correction. The resulting algorithm solves (3.1) to the level of the truncation error in a number of operations proportional to the number of variables. To sketch this method, called *full approximation scheme* (FAS), consider the discrete problem (3.1) expressed as

$$\mathcal{A}^h(u^h) = f^h \text{ on } \Omega_h. \tag{4.3}$$

On the grid of level k, with mesh size h, the smoothing procedure is denoted by $u^h = S^\nu(u^h, f^h)$, where S^ν is the smoothing operator applied ν times. This corresponds, for example, to ν steps of the GSP scheme. To correct for the smooth component of the error, a coarse grid correction (CGC) is defined. First, a coarse grid problem is constructed on the grid with mesh size $H = h_{k-1}$,

$$\mathcal{A}^H(u^H) = I_h^H f^h + \tau_h^H, \tag{4.4}$$

where $I_h^H : \Omega_h \to \Omega_H$ denotes a restriction operator, and τ_h^H is the fine-to-coarse defect correction defined by

$$\tau_h^H = \mathcal{A}^H(\hat{I}_h^H u^h) - I_h^H \mathcal{A}^h(u^h), \tag{4.5}$$

with $\hat{I}_h^H : \Omega_h \to \Omega_H$ being a restriction operator not necessarily equal to I_h^H. For example, $\hat{I}_h^H$ can be the straight injection. Once the coarse grid problem is solved, the coarse grid correction follows,

$$u_{new}^h = u^h + I_H^h(u^H - I_h^H u^h), \tag{4.6}$$

where $I_H^h : \Omega_H \to \Omega_h$ represents an interpolation operator. If the high frequency components of the error on the finer grid are indeed well damped, then the grid

Ω_H should provide enough resolution for u^h, and hence $I_H^h u^H$ should be a good approximation for u^h. This idea of transferring to a coarser grid can be applied along the set of nested meshes. One starts at level M with a zero approximation and applies the smoothing iteration ν_1 times. Then the problem is transferred to a coarser grid and so on. Once the coarsest grid is reached, one solves the coarsest problem to convergence by applying, as we do, a few steps of the smoothing iteration. The solution obtained on each grid is then used to correct the approximation on the next finer grid. The coarse grid correction followed by ν_2 postsmoothing steps is applied from one grid to the next, down to the finest grid with level M. This entire process represents one multigrid cycle.

The application of N FAS cycles is denoted by N-FAS. One can choose a starting grid with a level number $K < M$ which is coarser than the finest grid where the solution is desired. In this case one applies N-FAS on level K and then the solution is interpolated on the next finer grid. The interpolation provides a first approximation for the N-FAS on this finer level and so on until the finest grid is reached. The combination of the nested iteration technique and the N-FAS scheme is called the N-FMG scheme.

4.2.1. Prolongation and restriction operators. It is useful to recall the algebraic interpretation of the multigrid method in order to choose optimal prolongation and restriction operators for the present nonlinear problem. See [1, 8, 24] for a general discussion on this aspect of multigrid methods.

In one spatial dimension, it is known that a standard choice of multigrid components can lead to an effective direct solver (cyclic reduction). For example, combining the algebraic equations of the discrete system, which corresponds to (3.1) on Ω_h, one obtains

$$u_{i-2} - 2u_i + u_{i+2} + H^2\delta\left[\frac{e^{u_{i-1}} + 2e^{u_i} + e^{u_{i+1}}}{4}\right] = H^2\left[\frac{f_{i-1} + 2f_i + f_{i+1}}{4}\right],$$

where $H = 2h$. Whereas, if we represent our problem on the grid Ω_H we have,

$$u_{i-2} - 2u_i + u_{i+2} + H^2\delta e^{u_i} = H^2\hat{f}_i.$$

Hence, one can derive an expression for $\hat{f}_i$ so that the solution of the two algebraic problems coincide, that is,

$$\hat{f}_i = \frac{f_{i-1} + 2f_i + f_{i+1}}{4} - \delta\left[\frac{e^{u_{i-1}} + 2e^{u_i} + e^{u_{i+1}}}{4}\right] + \delta e^{u_i}.$$

Comparing this formula with the definition of the coarse problem (4.4) in the FAS scheme, where

$$f^H = I_h^H f^h + \mathcal{A}^H(\hat{I}_h^H u^h) - I_h^H \mathcal{A}^h(u^h),$$

an explicit expression for I_h^H is obtained. Namely, in stencil notation,

$$I_h^H = [1 \quad 2 \quad 1]/4,$$

and $\hat{I}_h^H$ is the straight injection operator.

In two dimensions these operators correspond to the half-weighted restriction and the simple injection, respectively, [12, 24]. By using I_h^H and $\hat{I}_h^H$ optimal convergence properties are observed as can be seen in Table 2 in the next section. For prolongation I_H^h we use the bilinear interpolation operator.

TABLE 2
Contraction factors ρ^*. (Use $\delta = 6.0$ and $f = 0$.)

	Injection	Full-	Half-weighted
M	ρ^*	ρ^*	ρ^*
3	0.75	0.129	0.150
4	0.106	0.164	0.158
5	0.179	0.164	0.156
6	0.180	0.165	0.155
7	0.175	0.166	0.155

4.3. Experiments with the steady state solid fuel ignition model. We conducted numerical experiments to measure efficiency, accuracy, and robustness of the solution process defined by finite difference discretization and multigrid methods.

Before we consider specific examples let us define some quantities which will be used to describe the numerical features of our algorithm. We shall report the values of the maximum norm of the solution error, $e^h = u^h - u$, on the grid Ω_h, that is, $||e^h||_0 = \max_{\Omega_h} |u^h - u|$. If an exact solution of the continuous problem is not known we take the solution on Ω_h as the reference solution, compute the intergrid error $E^H = u^H - \hat{I}_h^H u^h$ on Ω_H, and the norm $||E^H||_0 = \max_{\Omega_H} |u^H - I_h^H u^h|$ [7].

We define the contraction factor as the "asymptotic" value of the ratio between the discrete L^2-norm of the residuals $r(u)^h = f^h - \mathcal{A}^h(u^h)$ resulting from two successive multigrid cycles on a given mesh [12],

$$\rho^* = \lim_N \frac{||r(u)^h_{(N)}||_{L^2}}{||r(u)^h_{(N-1)}||_{L^2}}. \tag{4.7}$$

In order to measure the computational cost of a multigrid cycle let us use the "work unit" (WU) [7]. A work unit is the computational work done by one smoothing step on the finest grid.

For the experiments reported in the following sections some parameters of the multigrid algorithm are held fixed: Two pre- and one postsmoothing step are used on each level. Further, $K = 3$ if an N-FMG algorithm is applied. We use the half-weighted restriction operator discussed in the previous section. This choice provides the best contraction number for the multigrid cycle as can be seen in Table 2. (In the tables, the notation (-k) means 10^{-k}.)

We consider two examples. For the first one we choose $u(x, y) = \frac{1}{\pi^2} \sin(\pi x) \sin(\pi y)$ and $\delta = 6.0$. By substituting this function into the steady state solid fuel ignition model (3.1) we obtain an expression for f so that (3.1) admits u as a solution. In the second example, $f = 0$ is taken, and because $\delta < \delta^* \sim 6.80$ we know that the model admits at least one solution. In both cases, the Jensen inequality is satisfied.

4.3.1. Example 1. In this case the function f is given such that the solution u of (3.1) is known. Results of numerical experiments are reported in Table 3. The quantity $||e^h||_0$ computed by the N-FMG scheme is reported for each level. ($M = \{3, 4, 5, 6\}$.) Observe that already for 4-FMG, the norm of the error reduces almost exactly by a factor $H^2/h^2 = 4$ as stated in Lemma 3.1. Similar results can be shown relative to the norm of the residual of (3.1), in agreement with the fact that the truncation error behaves like $O(h^2)$. This means that the problem has been solved to the level of this error. Therefore no improvement in the solution can be expected by further cycling of the multigrid method. This is clearly shown by comparison of the results obtained by 4-FMG and 6-FMG.

TABLE 3
Convergence behavior: Example 1.

	1-FMG	2-FMG	4-FMG	6-FMG	6-FMG
k	$\|\|e^h\|\|_0$	$\|\|e^h\|\|_0$	$\|\|e^h\|\|_0$	$\|\|e^h\|\|_0$	$\|\|e^H\|\|_0/\|\|e^h\|\|_0$
3	0.14(-2)	0.87(-3)	0.36(-3)	0.36(-3)	-
4	0.45(-3)	0.91(-4)	0.90(-4)	0.90(-4)	4.0
5	0.24(-4)	0.22(-4)	0.22(-4)	0.22(-4)	4.09
6	0.64(-5)	0.56(-5)	0.56(-5)	0.56(-5)	3.93

TABLE 4
Contraction factor ρ^* : *Example* 1.

M	$N=3$	$N=4$	$N=5$
3	0.159	0.159	0.129
4	0.109	0.119	0.146
5	0.103	0.104	0.138
6	0.109	0.109	0.130
7	0.113	0.114	0.126

TABLE 5
Convergence behavior: Example 2.

	1-FMG	2-FMG	4-FMG	6-FMG	6-FMG
k	$\|\|E^h\|\|_0$	$\|\|E^h\|\|_0$	$\|\|E^h\|\|_0$	$\|\|E^h\|\|_0$	$\|\|E^H\|\|_0/\|\|E^h\|\|_0$
3	0.15(-1)	0.43(-2)	0.57(-3)	0.51(-3)	-
4	0.37(-2)	0.16(-3)	0.13(-3)	0.13(-3)	3.92
5	0.47(-3)	0.34(-4)	0.32(-4)	0.32(-4)	4.06
6	0.30(-4)	0.86(-5)	0.85(-5)	0.82(-5)	3.90

TABLE 6
Contraction factor ρ^*: *Example* 2. ($\delta = 6.8$.)

M	$N=3$	$N=4$	$N=5$
3	0.14	0.14	0.27
4	0.53	0.49	0.48
5	0.46	0.45	0.45
6	0.43	0.44	0.45
7	0.37	0.42	0.44

In Table 4 we report the values of the quotient $\frac{||r(u)^h_{(N)}||_{L^2}}{||r(u)^h_{(N-1)}||_{L^2}}$ appearing in (4.7) to show that the present algorithm has a contraction factor independent of the number of grid points.

4.3.2. Example 2. We repeat the experiment presented above for the case $f = 0$ and $\delta = 6.0$, for which (3.1) admits at least one solution. In Table 5 the quantity $||E^H||_0$ is reported for each level ($M = \{3, 4, 5, 6\}$) after the application of the N-FMG scheme as in the Example 1. As in the previous example a second-order accurate solution is obtained by N-FMG for $N \geq 4$. Contraction factors similar to those found with the example above can be reported for the present case as well. Let us now choose $\delta = 6.8$ for which the solution is known to be very close to the turning point. In Table 6 we can see that the multigrid algorithm preserves its property of a mesh-size independent contraction factor.

5. Optimal control of the steady state solid fuel ignition model. In this section we show that multigrid methods can be used to efficiently solve the optimality systems arising in open loop optimal control for steady state problems. Control will be exerted by adding or subtracting thermal energy to the fuel through a source term. Due to the lack of coercivity of the state equation it is at first not obvious how to properly formulate optimal control problems related to the solid fuel ignition model. In previous work two formalisms were considered:

$$\begin{aligned} &\min_{f\in L^2(\Omega)} J(u(f),f),\\ &\Delta u+\delta\exp(u)=f \text{ in } \Omega,\\ &u=0 \text{ on } \partial\Omega, \end{aligned} \tag{5.1}$$

where the cost functional $J(u,f)$ is chosen either as

$$J_1(u,f)=\frac{1}{2}||u-z||^2_{L^2(\Omega)}+\frac{\beta}{2}||e^u-e^z||^2_{L^2(\Omega)}+\frac{\nu}{2}||f||^2_{L^2(\Omega)} \tag{5.2}$$

(see [15]), or as

$$J_2(u,f)=\frac{1}{2}||u-z||^2_{L^2(\Omega)}+\frac{1}{2}||\nabla u-\nabla z||^2_{L^2(\Omega)}+\frac{\nu}{2}||f||^2_{L^2(\Omega)} \tag{5.3}$$

(see [14]). In (5.2) and (5.3) the next to last terms guarantee the existence of solutions to (5.1). In these cost functionals ν is the weight of the cost of the control, β is a positive scaling factor, and $z\in H^2(\Omega)$ is the desired state. In both cases radial unboundedness of J in (5.1) is guaranteed by the choice of the cost functional. Thus, minimizing sequences are a priori bounded and existence of a solution $(f^*,u^*)=(f^*,u(f^*))$ to the optimal control problem (5.1) with $J=J_1$ or $J=J_2$ can be established. (For $J=J_2$, only the time-dependent case is considered in [14], but the same techniques can be used to treat the stationary case.)

To derive necessary optimality conditions of first order, we introduce the Lagrangian,

$$L(u,f,\lambda)=J(u,f)+<\Delta u+\delta\exp(u)-f,\lambda>_{H^{-1},H^1_0}. \tag{5.4}$$

Let $G:H^1_0(\Omega)\to H^{-1}$ denote the linearization of the nonlinear operator in (2.4) at an optimal solution u^*,

$$Gv=\Delta v+\delta e^{u^*}v.$$

Note that G, considered as operator in $L^2(\Omega)$, has compact resolvent so its spectrum consists only of eigenvalues. We assume that zero is not an eigenvalue of G. Under this condition it can be shown that there exists $\lambda^*\in H^1_0(\Omega)$ such that the first-order optimality conditions hold:

$$L_{(u,f)}(u^*,f^*,\lambda^*)=0 \quad \text{and} \quad \Delta u^*+\delta e^{u^*}=f^*,$$

where $L_{(u,f)}$ denotes the derivative of L with respect to (u,f). For $J=J_1$ this results in the optimality system

$$\begin{aligned} \Delta u+\delta\exp(u)-f&=0,\\ \Delta\lambda+\delta\exp(u)\lambda+(u-z)+\beta e^u(e^u-e^z)&=0,\\ \nu f-\lambda&=0, \end{aligned} \tag{OPC1}$$

where for convenience we omitted the superscript *. Similarly, the optimality condition for $J = J_2$ is found to be

$$\begin{aligned} \Delta u + \delta \exp(u) - f &= 0, \\ \Delta\lambda + \delta \exp(u)\lambda + (u - z) - \Delta(u - z) &= 0, \\ \nu f - \lambda &= 0. \end{aligned} \tag{OPC2}$$

In the following sections we shall concentrate on the numerical solution of (OPC1) and (OPC2) and their comparison from the point of view of open loop control. Before doing so, we first look at the regularity properties of (u, f, λ). Due to the last equation in (OPC1) and (OPC2) we have $f \in H^1(\Omega) \subset L^p(\Omega)$ for every $p \in [1, \infty)$. It follows that $u \in W^{2,p}(\Omega)$ and $\lambda \in W^{2,p}(\Omega)$ for both (OPC1) and (OPC2). In particular, u and λ as well as f are elements of $C(\Omega)$, so the numerical framework developed above is applicable.

6. Numerical solution of the optimal control problem. To solve the systems (OPC1) and (OPC2) we first discretize them using finite differences and then solve the resulting nonlinear algebraic systems using a multigrid method similar to that used for the steady state solid fuel ignition model.

The discrete system corresponding to (OPC1) is denoted by

$$\begin{aligned} \Delta^h u^h + \delta \exp(u^h) - f^h &= 0, \\ \Delta^h \lambda^h + \delta \exp(u^h)\lambda^h + (u^h - z^h) + \beta e^{u^h}(e^{u^h} - e^{z^h}) &= 0, \\ \nu f^h - \lambda^h &= 0, \end{aligned} \tag{6.1}$$

and for (OPC2) we have

$$\begin{aligned} \Delta^h u^h + \delta \exp(u^h) - f^h &= 0, \\ \Delta^h \lambda^h + \delta \exp(u^h)\lambda^h + (u^h - z^h) - \Delta(u^h - z^h) &= 0, \\ \nu f^h - \lambda^h &= 0. \end{aligned} \tag{6.2}$$

While we shall not analyze the convergence of the coupled systems (6.1) and (6.2) we observe that for given f the state equations in (6.1) and (6.2) have second-order convergence property by the results of section 3. Further the costate equations are linear in λ^h so that the results of Bramble [5] apply and we expect to obtain second-order accurate solutions of the optimality systems.

To solve (6.1) and (6.2) two smoothing schemes are considered in the construction of the multigrid algorithm. In the first case, as in [2, 3], we make the following definition: *An iteration step, $u^h = S^\nu(u^h, f^h)$, consists of one step of an iterative solver applied to the state equation, then one step of an iterative solver applied to the costate equation and finally update of the control function by means of $f = \lambda/\nu$.* As iterative solver for each equation we take one iteration of the GSP iteration.

In the second case, one iteration of the GSN iteration is used. It is a Newton step applied at each grid point to the set of variables $\phi = (u, \lambda, f)$. It is defined by

$$\phi_{ij}^{(\text{new})} = \phi_{ij}^{(\text{old})} - [\mathbf{G}'(\phi_{ij})]^{-1}\mathbf{G}(\phi_{ij}). \tag{6.3}$$

The vector equation $\mathbf{G}(u^h, \lambda^h, f^h) = \mathbf{0}$ represents (6.1) or (6.2) and $\mathbf{G}'(u^h, \lambda^h, f^h)$ denotes the Jacobian of $\mathbf{G}$.

We shall use either the GSP-multigrid method with the GSP scheme used as the smoother or the GSN-multigrid method with the GSN iteration scheme used as the smoother. Both methods are summarized as follows:

GSP- or GSN-multigrid method for solving $A^h(\phi^h) = f^h$.
(1) Apply ν_1 smoothing steps: $\phi^h = S^{\nu_1}(\phi^h, f^h)$.
(2) Transfer the approximate solution: $\phi^H = \hat{I}_h^H \phi^h$.
(3) Compute the right-hand side of the FAS equation: $f^H = I_h^H f^h + \tau_h^H$.
(4) Apply γ-FAS scheme to $A^H(\hat{\phi}^H) = f^H$ (iterative solution if $k = 1$).
(5) Use coarse level correction: $\phi^h = \phi^h - I_H^h(\phi^H - \hat{\phi}^H)$.
(6) Apply ν_2 smoothing steps: $\phi^h = S^{\nu_2}(\phi^h, f^h)$.

Notice that this algorithm does not involve any global linearization of the optimal control problem. It is applied directly to the optimal control system whereas other multigrid approaches [13] set up an integral equation characterizing the optimal control. Both the GSP-multigrid and GSN-multigrid solve (OPC1) and (OPC2) in a few multigrid cycles to second-order accuracy.

6.1. Experiments with the optimal control of the steady state solid fuel ignition model. In this section we give examples demonstrating the efficiency of the proposed methods to solve optimal control problems. We report the values of the tracking functionals $||u^h - z^h||_0$ as well as of the control costs $||f||_{L^2} = ||\lambda||_{L^2}/\nu$. The numerical accuracy of the solution can be expressed by $||E^H||_0$ and the norm of the error of the adjoint variable λ is defined by $||V^H||_0 = \max_{\Omega_H} |\lambda^H - \hat{I}_h^H \lambda^h|$. The residual of the adjoint equation is denoted by $r(\lambda)^h$. We recall that the critical value for δ is $\delta^* = 6.808$.

We first consider the GSP-multigrid method. For the desired state we take $z(x, y) = \frac{1}{\pi^2} \sin(\pi x) \sin(\pi y)$ and we set the parameters $\delta = \beta = 6.0$ and $\nu = 5 \cdot 10^{-3}$. For this value of ν good results are obtained. It turns out, however, that for smaller values of ν the performance of the GSP-multigrid method quickly degenerates. Thus the method is not robust with respect to this parameter. Some results with (OPC1) are reported in Table 7 where we can note that the values of $||u^h - z^h||_0$ quickly reach an optimal value and that second-order convergence of E^H holds. In Table 8 analogous results are presented for (OPC2). In both cases the N-FMG scheme with $N > 2$ guarantees an efficient optimal control and second-order convergence. The advantage of using multigrid acceleration of the GSP scheme can be seen in Figure 1 where the convergence history of the GSP-multigrid and the GSP iteration are compared for solving (OPC1). A similar behavior is obtained for (OPC2). In this figure one can notice that for the GSP iteration the error of the costate equation may initially increase, but nevertheless the GSP-multigrid method shows the desired convergence properties.

We have obtained a significant improvement in robustness upon the GSP-multigrid method by implementing the GSN-multigrid scheme. This method computes the solution of (OPC1) and (OPC2) in a few work units with values of ν as small as 10^{-9}. See the convergence history of the GSN-multigrid method illustrated in Figures 2 and 3. We were not able to obtain similar results with the GSP scheme.

The results reported in Tables 9, 10, and 11 confirm the expected optimal control behavior: As ν decreases, $||f^h||_{L^2}$ increases and $||u^h - z^h||_0$ decreases, both for (OPC1) and (OPC2). In both cases the values of $||u^h - z^h||_0$ are almost independent of the discretization level, showing that the numerical solution quickly attains its optimal value. As $||u^h - z^h||_0$ measures tracking ability, we observe the difference of values between (OPC1) and (OPC2) and note that, relative to control force, the J_2-based formulation allows better tracking than the J_1-based formulation.

A direct comparison among these tables shows that the GSN-multigrid method is very robust with respect to changes of ν. By inspecting the values of $||E^H||_0$ and

TABLE 7
Tracking and accuracy results with (OPC1) *obtained by the GSP-multigrid method.*

	2-FMG		5-FMG	
k	$\|\|u^h - z^h\|\|_0$	$\|\|E^H\|\|_0$	$\|\|u^h - z^h\|\|_0$	$\|\|E^H\|\|_0$
3	0.63(-1)	0.144(-1)	0.54(-1)	0.838(-3)
4	0.56(-1)	0.311(-2)	0.54(-1)	0.138(-3)
5	0.55(-1)	0.456(-3)	0.54(-1)	0.347(-4)
6	0.55(-1)	0.326(-4)	0.54(-1)	0.869(-5)
7	0.54(-1)	-	0.54(-1)	-

TABLE 8
Tracking and accuracy results with (OPC2) *obtained by the GSP-multigrid method.*

	2-FMG		5-FMG	
k	$\|\|u^h - z^h\|\|_0$	$\|\|E^H\|\|_0$	$\|\|u^h - z^h\|\|_0$	$\|\|E^H\|\|_0$
3	0.13	0.144(-2)	0.138	0.613(-3)
4	0.13	0.154(-3)	0.138	0.154(-3)
5	0.13	0.389(-4)	0.138	0.387(-4)
6	0.13	0.978(-5)	0.138	0.969(-5)
7	0.13	–	0.138	–

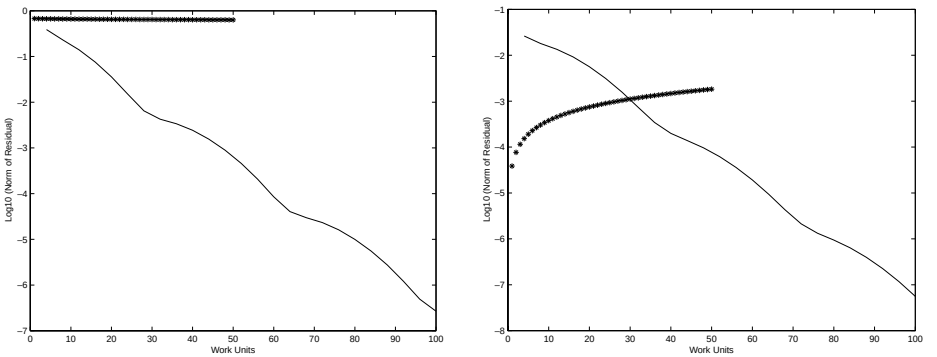


FIG. 1. *Comparison of the GSP-multigrid method* (*with* "–") *and the GSP scheme* (*with* "*") *for solving the state equation (left) and the costate equation (right) of* (OPC1). (*Use* $\delta = \beta = 6.0$ *and* $\nu = 5 \cdot 10^{-3}$.)

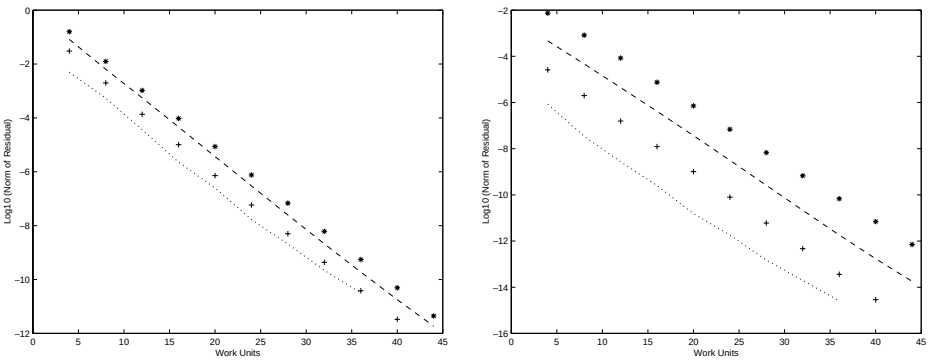


FIG. 2. *Convergence history of the GSN-multigrid method for solving the state equation (left) and the costate equation (right) of* (OPC1) *for different* ν. *We have* $\nu = 10^{-3}$ *with* "*," $\nu = 10^{-5}$ *with* "–," $\nu = 10^{-7}$ *with* "+," $\nu = 10^{-9}$ *with* "...."

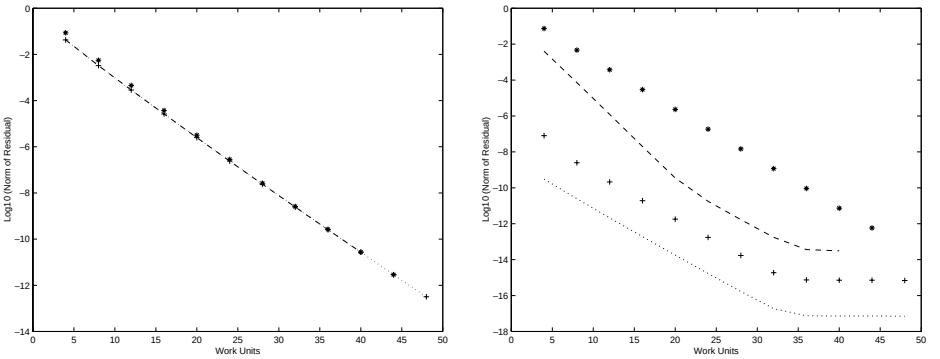


FIG. 3. *Convergence history of the GSN-multigrid method for solving the state equation (left) and the costate equation (right) of* (OPC2) *for different* ν. *We have* $\nu = 10^{-3}$ *with* "*," $\nu = 10^{-5}$ *with* "–," $\nu = 10^{-7}$ *with* "+," $\nu = 10^{-9}$ *with* "...."

TABLE 9

Tracking and accuracy results with (OPC1) *and* (OPC2) *obtained by the GSP-multigrid method. Use* $\beta = \delta = 6.8$ *and* $\nu = 10^{-1}$.

	OPC1				OPC2			
	3-FMG				3-FMG			
k	$\|\|u^h - z^h\|\|_0$	$\|\|E^H\|\|_0$	$\|\|V^H\|\|_0$	$\|\|f^h\|\|_{L^2}$	$\|\|u^h - z^h\|\|_0$	$\|\|E^H\|\|_0$	$\|\|V^H\|\|_0$	$\|\|f^h\|\|_{L^2}$
3	0.34	0.20(-2)	0.14(-2)	2.57	0.25	0.12(-2)	0.18(-2)	3.04
4	0.34	0.39(-3)	0.30(-3)	2.58	0.25	0.28(-3)	0.41(-3)	3.05
5	0.34	0.99(-4)	0.75(-4)	2.58	0.25	0.72(-4)	0.10(-3)	3.05
6	0.34	0.24(-4)	0.19(-4)	2.58	0.25	0.18(-4)	0.25(-4)	3.05
7	0.34	–	–	2.58	0.25	–	–	3.05
	5-FMG				5-FMG			
k	$\|\|u^h - z^h\|\|_0$	$\|\|E^H\|\|_0$	$\|\|V^H\|\|_0$	$\|\|f^h\|\|_{L^2}$	$\|\|u^h - z^h\|\|_0$	$\|\|E^H\|\|_0$	$\|\|V^H\|\|_0$	$\|\|f^h\|\|_{L^2}$
3	0.33	0.15(-2)	0.12(-2)	2.57	0.25	0.11(-2)	0.16(-2)	3.04
4	0.34	0.39(-3)	0.30(-3)	2.58	0.25	0.28(-3)	0.41(-3)	3.05
5	0.34	0.98(-4)	0.75(-4)	2.58	0.25	0.72(-4)	0.10(-3)	3.05
6	0.34	0.24(-4)	0.18(-4)	2.58	0.25	0.18(-4)	0.26(-4)	3.05
7	0.34	–	–	2.58	0.25	–	–	3.05

TABLE 10

Tracking and accuracy results with (OPC1) *and* (OPC2) *obtained by the GSP-multigrid method. Use* $\beta = \delta = 6.8$ *and* $\nu = 10^{-3}$.

	OPC1				OPC2			
	3-FMG				3-FMG			
k	$\|\|u^h - z^h\|\|_0$	$\|\|E^H\|\|_0$	$\|\|V^H\|\|_0$	$\|\|f^h\|\|_{L^2}$	$\|\|u^h - z^h\|\|_0$	$\|\|E^H\|\|_0$	$\|\|V^H\|\|_0$	$\|\|f^h\|\|_{L^2}$
3	0.29(-1)	0.72(-3)	0.46(-4)	4.95	0.54(-2)	0.31(-3)	0.30(-3)	5.56
4	0.30(-1)	0.18(-3)	0.11(-4)	4.95	0.55(-2)	0.12(-3)	0.12(-3)	5.65
5	0.30(-1)	0.47(-4)	0.29(-5)	4.96	0.56(-2)	0.35(-4)	0.35(-4)	5.67
6	0.30(-1)	0.19(-4)	0.73(-6)	4.96	0.56(-2)	0.93(-5)	0.93(-5)	5.67
7	0.30(-1)	–	–	4.96	0.56(-2)	–	–	5.68
	5-FMG				5-FMG			
k	$\|\|u^h - z^h\|\|_0$	$\|\|E^H\|\|_0$	$\|\|V^H\|\|_0$	$\|\|f^h\|\|_{L^2}$	$\|\|u^h - z^h\|\|_0$	$\|\|E^H\|\|_0$	$\|\|V^H\|\|_0$	$\|\|f^h\|\|_{L^2}$
3	0.29(-1)	0.72(-3)	0.45(-4)	4.93	0.54(-2)	0.30(-3)	0.30(-3)	5.56
4	0.30(-1)	0.18(-3)	0.11(-4)	4.95	0.55(-2)	0.12(-3)	0.12(-3)	5.65
5	0.30(-1)	0.47(-4)	0.29(-5)	4.96	0.56(-2)	0.35(-4)	0.35(-4)	5.67
6	0.31(-1)	0.12(-4)	0.73(-6)	4.96	0.56(-2)	0.93(-5)	0.93(-5)	5.67
7	0.31(-1)	–	–	4.96	0.56(-2)	–	–	5.68

TABLE 11
Tracking and accuracy results with (OPC1) *and* (OPC2) *obtained by the GSP-multigrid method. Use* $\beta = \delta = 6.8$ *and* $\nu = 10^{-5}$.

	OPC1				OPC2			
	3-FMG				3-FMG			
k	$\|\|u^h - z^h\|\|_0$	$\|\|E^H\|\|_0$	$\|\|V^H\|\|_0$	$\|\|f^h\|\|_{L^2}$	$\|\|u^h - z^h\|\|_0$	$\|\|E^H\|\|_0$	$\|\|V^H\|\|_0$	$\|\|f^h\|\|_{L^2}$
3	0.26(-2)	0.41(-3)	0.35(-5)	5.72	0.64(-4)	0.27(-4)	0.33(-6)	5.82
4	0.30(-2)	0.15(-3)	0.12(-5)	5.81	0.65(-4)	0.11(-5)	0.11(-5)	6.05
5	0.33(-2)	0.45(-4)	0.30(-6)	5.83	0.66(-4)	0.29(-5)	0.28(-5)	6.15
6	0.34(-2)	0.11(-4)	0.77(-7)	5.84	0.67(-4)	0.35(-5)	0.35(-5)	6.20
7	0.34(-2)	–	–	5.84	0.67(-4)	–	–	6.21
	5-FMG				5-FMG			
k	$\|\|u^h - z^h\|\|_0$	$\|\|E^H\|\|_0$	$\|\|V^H\|\|_0$	$\|\|f^h\|\|_{L^2}$	$\|\|u^h - z^h\|\|_0$	$\|\|E^H\|\|_0$	$\|\|V^H\|\|_0$	$\|\|f^h\|\|_{L^2}$
3	0.26(-2)	0.42(-3)	0.35(-5)	5.72	0.64(-4)	0.33(-6)	0.33(-6)	5.82
4	0.30(-2)	0.15(-3)	0.12(-5)	5.81	0.65(-4)	0.11(-5)	0.11(-5)	6.05
5	0.33(-2)	0.45(-4)	0.30(-6)	5.83	0.66(-4)	0.29(-5)	0.28(-5)	6.15
6	0.33(-2)	0.11(-4)	0.77(-7)	5.84	0.67(-4)	0.35(-5)	0.35(-5)	6.20
7	0.34(-2)	–	–	5.84	0.67(-4)	–	–	6.21

TABLE 12
Convergence results with (OPC1) *and with* (OPC2).

OPC1						
	3-FMG			5-FMG		
ν	$\|\|r(u)^h\|\|_{L^2}$	$\|\|r(\lambda)^h\|\|_{L^2}$	WU	$\|\|r(u)^h\|\|_{L^2}$	$\|\|r(\lambda)^h\|\|_{L^2}$	WU
10^{-1}	9.42(-7)	1.35(-7)	15.98	6.56(-9)	2.10(-9)	26.63
10^{-3}	9.56(-7)	1.70(-8)	15.98	6.72(-9)	1.53(-10)	26.63
10^{-5}	1.29(-6)	1.09(-8)	15.98	9.19(-9)	6.74(-11)	26.63
OPC2						
	3-FMG			5-FMG		
ν	$\|\|r(u)^h\|\|_{L^2}$	$\|\|r(\lambda)^h\|\|_{L^2}$	WU	$\|\|r(u)^h\|\|_{L^2}$	$\|\|r(\lambda)^h\|\|_{L^2}$	WU
10^{-1}	9.45(-7)	3.29(-8)	15.98	6.57(-9)	3.54(-10)	26.63
10^{-3}	1.48(-6)	8.27(-9)	15.98	9.00(-9)	4.92(-11)	26.63
10^{-5}	1.55(-7)	7.71(-11)	15.98	5.43(-11)	6.12(-13)	26.63

$||V^H||_0$ reported in the Tables 9, 10, and 11, we can state that the GSN-multigrid algorithm provides second-order convergent solutions to the optimal control systems. In fact, provided $M > 3$, we already observe for the 3-FMG method a ν-independent reduction of $||E^H||_0$ and $||V^H||_0$ by a factor of four from one mesh to the next. Moreover, from Table 12 we conclude that only a few work units are employed by the 3-FMG method and by the 5-FMG method to reduce the values of the residuals below a small tolerance. (Here, we used $\beta = \delta = 6.8$.)

In Table 13 we depict the convergence factors for the state and the costate equations of (OPC1) and (OPC2), with various values for δ, $\beta = \delta$, and $\nu = 10^{-5}$. We observe that for a wide range of δ-values the convergence factors are quite independent of the number of variables. Note here that the controlled system is considered for values of δ significantly larger than the critical value δ^* beyond which the uncontrolled system does not admit solutions. A slight deterioration is revealed as the coefficient of the nonlinearity becomes very large.

When solving (OPC1) a value of the scaling factor β must be chosen. In Table 14 the influence of this factor on the numerical solution of (OPC1) is documented. Note that $||u^h - z^h||^2_{L^2(\Omega)}$ decreases as β is increased.

The convergence properties of the GSN-multigrid method discussed above have been tested against a wide range of choices for z. For example, take $z_a(x, y) =$

TABLE 13
Convergence factors of the GSN-multigrid method on different grids and various δ.

OPC1				
	M=4		M=7	
δ	ρ_u^*	ρ_λ^*	ρ_u^*	ρ_λ^*
1	0.09	0.10	0.09	0.09
10	0.07	0.09	0.08	0.09
50	0.15	0.16	0.12	0.12
100	0.19	0.19	0.13	0.12
OPC2				
	M=4		M=7	
δ	ρ_u^*	ρ_λ^*	ρ_u^*	ρ_λ^*
1	0.09	0.09	0.10	0.11
10	0.09	0.09	0.10	0.05
50	0.09	0.09	0.10	0.05
100	0.07	0.07	0.09	0.05

TABLE 14
Convergence results for solving (OPC1) *using the* 3*-FMG method with various β, $\delta = 6.0$, and $M = 7$.*

OPC1					
β	$\|\|r(u)^h\|\|_{L^2}$	$\|\|r(\lambda)^h\|\|_{L^2}$	$\|\|E^H\|\|_0$	$\|\|u^h - z^h\|\|_0$	$\|\|f^h\|\|_{L^2}$
1	9.39(-7)	2.87(-9)	0.10(-4)	0.59(-2)	4.89
6	1.13(-6)	8.80(-9)	0.10(-4)	0.31(-2)	5.04
24	1.38(-6)	2.69(-8)	0.12(-4)	0.17(-2)	5.16
48	1.87(-6)	4.69(-8)	0.10(-4)	0.12(-2)	5.20
96	2.53(-6)	7.50(-8)	0.10(-4)	0.85(-3)	5.24
200	2.93(-6)	1.23(-7)	0.96(-5)	0.59(-3)	5.28
400	3.88(-6)	1.95(-7)	0.98(-5)	0.42(-3)	5.30
1000	2.61(-5)	1.13(-6)	0.71(-6)	0.26(-3)	5.33

TABLE 15
Convergence results for solving (OPC1) *and* (OPC2) *using the* 3*-FMG algorithm with $\delta = \beta = 6.8$, $\nu = 10^{-5}$, and $M = 7$.*

OPC1					
a	$\|\|u^h - z^h\|\|_0$	$\|\|r(u)^h\|\|_{L^2}$	$\|\|r(\lambda)^h\|\|_{L^2}$	ρ_u^*	ρ_λ^*
1	0.34(-2)	1.29(-6)	1.09(-8)	0.08	0.08
3	0.33(-2)	1.28(-6)	1.09(-8)	0.08	0.08
6	0.33(-2)	1.27(-6)	1.08(-8)	0.08	0.08
9	0.32(-2)	1.26(-6)	1.08(-8)	0.08	0.08
18	0.31(-2)	1.22(-6)	1.11(-8)	0.08	0.08
144	0.26(-2)	9.50(-6)	4.11(-4)	0.06	0.10
OPC2					
a	$\|\|u^h - z^h\|\|_0$	$\|\|r(u)^h\|\|_{L^2}$	$\|\|r(\lambda)^h\|\|_{L^2}$	ρ_u^*	ρ_λ^*
1	0.67(-4)	1.55(-7)	7.70(-11)	0.01	0.01
3	0.67(-4)	1.55(-7)	1.15(-10)	0.01	0.01
6	0.66(-4)	1.55(-7)	1.98(-10)	0.01	0.01
9	0.66(-4)	1.55(-7)	2.28(-10)	0.01	0.01
18	0.66(-4)	1.55(-7)	5.68(-10)	0.01	0.01
144	7.93	3.43(-6)	3.36(-6)	0.08	0.02

$\frac{a}{\pi^2}\sin(\pi x)\sin(\pi y)$. The contraction factors, the level of residuals reached, and the values of $||u^h - z^h||_0$ remain almost invariant when increasing the coefficient a; see Table 15.

TABLE 16
Convergence results for solving (OPC1) *and* (OPC2) *using the* 5*-FMG algorithm with objective functions* z_1 *and* z_2*. Here is* $\delta = \beta = 6.8$, $\nu = 10^{-4}$, *and* $M = 7$.

OPC1						
z	$\|\|u^h - z^h\|\|_0$	$\|\|r(u)^h\|\|_{L^2}$	$\|\|r(\lambda)^h\|\|_{L^2}$	$\|\|f^h\|\|_{L^2}$	ρ_u^*	ρ_λ^*
z_1	0.10(-1)	2.82(-8)	3.98(-10)	6.06	0.08	0.08
z_2	0.10	6.31(-8)	3.46(-9)	5.89	0.08	0.08
OPC2						
z	$\|\|u^h - z^h\|\|_0$	$\|\|r(u)^h\|\|_{L^2}$	$\|\|r(\lambda)^h\|\|_{L^2}$	$\|\|f^h\|\|_{L^2}$	ρ_u^*	ρ_λ^*
z_1	0.64(-3)	4.10(-8)	3.15(-11)	6.59	0.06	0.06
z_2	0.10	4.07(-8)	3.80(-10)	6.53	0.06	0.06

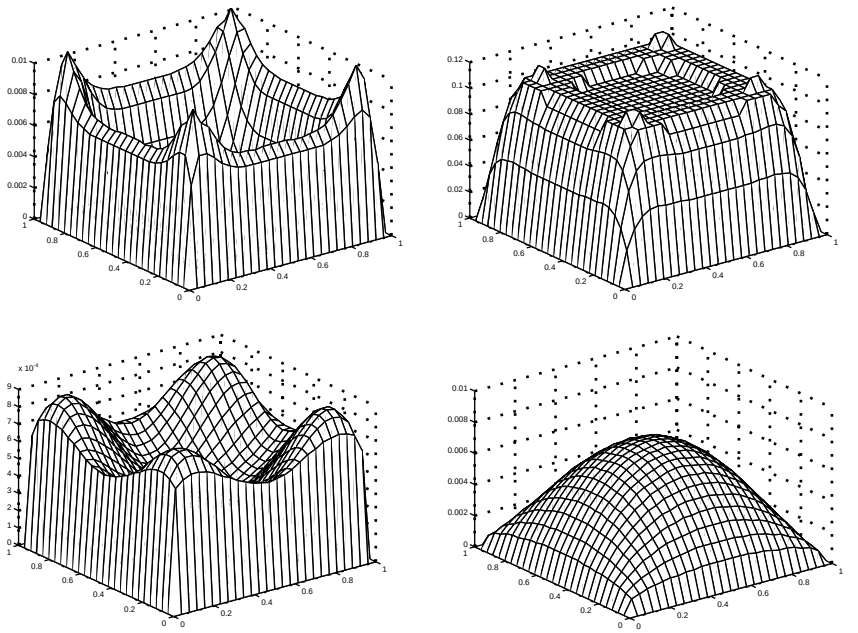


FIG. 4. *Numerical solutions of* (OPC1) *(top) and* (OPC2) *(bottom) with objective functions* z_1 *(left) and* z_2 *(right).*

Other examples were tested using the following objective functions:

$$z_1(x,y) = \frac{1}{\pi^2}\sin(\pi x)\sin(\pi y)(x-1/2)^2(y-1/2)^2$$

and

$$z_2(x,y) = \frac{1}{\pi^2}.$$

Here z_2 does not satisfy homogeneous Dirichlet boundary conditions and therefore is not attainable by any control. In both cases the GSN-multigrid scheme performs well, as can be seen in Table 16. To better interpret these last results see Figure 4.

7. Conclusions. Optimal solution and control algorithms for the steady state solid fuel ignition model were presented. They are based on finite difference discretization and multigrid techniques. Two alternative formulations of the optimal control

problem were considered and in both cases the desired optimal control behavior has been obtained. The numerical solution process was based on the results of our analysis: We gave necessary and sufficient conditions for existence of solutions to the direct problem and proved second-order accuracy of the numerical solution. The main tool was the method of upper and lower solutions that was also used to analyze a suitable numerical iterative method. We considered two types of smoothers in the nonlinear multigrid scheme. This method solves the optimal control problem to the level of truncation errors in only a few cycles. Moreover, in the refining process, second-order accuracy of the solution to the direct problem as well as to optimality systems was observed. The method appears to be very efficient and robust with respect to changes in the value of the weights in the cost functionals.

Acknowledgments. The authors would like to thank Dr. A. Kauffmann, Dr. M. Vanmaele, and Dr. S. Volkwein for helpful discussions.

REFERENCES

[1] R.E. Alcouffe, A. Brandt, J.E. Dendy, and J.W. Painter, *The multigrid method for the diffusion equation with strongly discontinuous coefficients*, SIAM J. Sci. Statist. Comput., 2 (1981), pp. 430–454.

[2] E. Arian and S. Ta'asan, *Smoothers for optimization problems*, in Seventh Copper Mountain Conference on Multigrid Methods, Vol. CP3339, N.D. Melson, T.A. Manteuffel, S.F. McCormick, and C.C. Douglas, eds., NASA Conference Publication, NASA, Hampton, VA, 1995, pp. 15–30.

[3] E. Arian and S. Ta'asan, *Shape optimization in one shot*, in Optimal Design and Control, J. Boggaard, J. Bunkardt, M. Gunzburger, J. Peterson, eds., Birkhauser, Boston, 1995, pp. 23–40.

[4] J. Bebernes and D. Eberly, *Mathematical Problems from Combustion Theory*, Springer-Verlag, New York, 1988.

[5] J.H. Bramble, *Error estimates for difference methods in forced vibration problems*, SIAM J. Numer. Anal., 3 (1966), pp. 1–12.

[6] A. Brandt, *Multi-level adaptive solutions to boundary-value problems*, Math. Comput., 31 (1977), pp. 333–390.

[7] A. Brandt, *Multigrid Techniques:* 1984 *Guide with Applications to Fluid Dynamics*, GMD-Studien 85, St. Augustin, Germany, 1984.

[8] A. Brandt, *Algebraic multigrid theory: The symmetric case*, Appl. Math. Comput., 19 (1986), pp. 23–56.

[9] T.F.C. Chan and H.B. Keller, *Arc-length continuation and multigrid techniques for nonlinear elliptic eigenvalue problem*, SIAM J. Sci. Statist. Comput., 3 (1982), pp. 173–194.

[10] H. Fujita, *On the nonlinear equations* $\Delta u + \exp(u) = 0$ *and* $v_t = \Delta v + \exp(u)$, Bull. Amer. Math. Soc., 75 (1969) pp. 132–135.

[11] R. Glowinski, H. B. Keller, and L. Rheinhart, *Continuation-conjugate gradient methods for the least-squares solution of nonlinear boundary value problems*, SIAM J. Sci. Statist. Comput., 6 (1985), pp. 793–832.

[12] W. Hackbusch, *Multi-Grid Methods and Applications*, Springer-Verlag, New York, 1985.

[13] W. Hackbusch, *Fast solution of elliptic control problems*, J. Optim. Theory Appl., 31 (1980), pp. 565–581.

[14] K. Ito and K. Kunisch, *Optimal Control of the Solid Fuel Model with* H^1*-Cost*, preprint, 1998.

[15] A. Kauffmann, *Optimal Control of the Solid Fuel Ignition Model*, Ph.D. thesis, Technical University of Berlin, Berlin 1998.

[16] G.S. Ladde, V. Lakshmikantham, and A.S. Vatsala, *Monotone Iterative Techniques for Nonlinear Differential Equations*, Pitman, Boston, 1985.

[17] J.-L. Lions, *Control of Distributed Singular Systems*, Gauthier-Villars, Paris, 1985.

[18] Th. Meis, H. Lehmann, and H. Michael, *Application of the multigrid method to a nonlinear indefinite problem*, in Multigrid Methods, W. Hackbusch and U. Trottenberg, eds., Lecture Notes in Math. 960, Springer-Verlag, New York, 1982.

[19] A.R. Mitchell and D.F. Griffiths, *The Finite Difference Method in Partial Differential Equations*, John Wiley and Sons, Chichester, UK, 1985.
[20] J.M. Ortega, *Numerical Analysis*, Academic Press, New York, 1972.
[21] C.V. Pao, *Monotone iterative methods for finite difference reaction diffusion equations*, Numer. Math., 46 (1985), pp. 571–586.
[22] C.V. Pao, *Nonlinear Parabolic and Elliptic Equations*, Plenum Press, New York, 1992.
[23] D.H. Sattinger, *Monotone methods in nonlinear elliptic and parabolic boundary value problems*, Indiana Univ. Math. J., 21 (1972), pp. 979–1000.
[24] K. Stüben and U. Trottenberg, *Multigrid methods: Fundamental algorithms, model problem analysis and applications*, in Multigrid Methods, W. Hackbusch and U. Trottenberg, eds., Lecture Notes in Math. 960, Springer-Verlag, New York, 1982.
[25] R.S. Varga, *Matrix Iterative Analysis*, Prentice-Hall, Englewood Cliffs, NJ, 1962.

SIAM J. SCI. COMPUT.
Vol. 22, No. 1, pp. 285–311
© 2000 Society for Industrial and Applied Mathematics

CONSTRUCTION OF SOLUTION CURVES FOR LARGE TWO-DIMENSIONAL PROBLEMS OF STEADY-STATE FLOWS OF INCOMPRESSIBLE FLUIDS*

VALMOR F. DE ALMEIDA† AND JEFFREY J. DERBY†

Abstract. This work represents a step toward advancing classical methods of bifurcation analysis in conjunction with very large-scale scientific computing needed to model realistically processes which involve laminar and steady flows of incompressible fluids. A robust method of analysis based on pseudoarclength continuation, Newton's method, and direct solution of linear systems is proposed and applied to the analysis of a representative system of fluid mechanics, namely the tilted lid driven cavity. Accurate solution curves, possessing simple singular points, were computed for Reynolds numbers varying from 0 to 10,000 and for different angles of tilt. The results demonstrate that two-dimensional models with up to 1,000,000 algebraic equations can be studied feasibly using the methods described here with state-of-the-art vector supercomputers.

Key words. incompressible and viscous fluids, Navier–Stokes equations, parameter continuation, path following, factorization of large unsymmetric sparse matrices, lid driven cavity

AMS subject classifications. 76M10, 65M30, 35B60

PII. S1064827598334514

1. Introduction.

Steady-state flows of incompressible fluids are at the heart of many materials and chemical processing systems [Derby et al., 1994; Salinger et al., 1994]. Typically, the fluid flows in such processes can be very complicated, three-dimensional (3D), time-dependent, and most often capricious. Their study, along with coupled physical phenomena such as heat transfer, mass transfer, and chemical reaction, often requires extremely large-scale computations, with fine discretizations, to resolve phenomena occurring at disparate length scales. Analysis, design, and optimization demand an understanding of the system's behavior under systematic change of its physical parameters—a prime example of the utility of bifurcation analysis but, at the same time, an enormous challenge due to the computational effort involved in obtaining accurate steady-state solutions.

Classical bifurcation analysis has been extremely useful for the understanding of mathematical models that depend on parameters. The most successful method is a combination of pseudoarclength parameter continuation [Keller, 1992], Newton's method, and factorization of the matrices of coefficients of the underlying systems of linear algebraic equations. Unfortunately, the high cost of this approach has delayed computer-aided bifurcation analysis of fluid systems in 3-D space. The bottleneck and most expensive part of the solution process is the factorization of large and sparse matrices. However, this situation is changing rapidly in view of the increasing computational power of parallel computers and important developments in computational

*Received by the editors February 20, 1998; accepted for publication (in revised form) October 15, 1998; published electronically June 20, 2000. This work was supported in part by the University of Minnesota Supercomputer Institute (MSI) and the Army High Performance Computing Research Center under the auspices of the Department of the Army, Army Research Laboratory cooperative agreement DAAH04-95-2-0003/contract DAAH04-95-C-0008.

http://www.siam.org/journals/sisc/22-1/33451.html

†Department of Chemical Engineering & Materials Science, Army High Performance Computing Research Center, and Minnesota Supercomputer Institute, University of Minnesota, Minneapolis, MN 55455-0132 (dalmeida@msi.umn.edu, http://www.msi.umn.edu/∼dalmeida; derby@tc.umn.edu). Current address: Oakridge National Laboratory, Oak Ridge, TN.

linear algebra. Of importance to this investigation are high performance BLAS2/3 [Basic Linear Algebra Subprograms, Dongarra et al., 1990] kernels supported by many vendors of high performance computers, fast and high quality graph partitioning and ordering [Karypis and Kumar, 1995], sparse factorization methods that exploit parallelism, and combined direct/iterative methods for solution of really challenging linear algebra problems [Duff, 1996].

Our overall goal is to develop classical methods of bifurcation analysis so that they become viable for realistic problems arising from modeling of materials processing systems in 3-D space. This article takes a first step in that direction by analyzing very large-scale two-dimensional flow problems. Its primary concern is to seek for modifications of the classical methods of bifurcation analysis that substantially increase their performance at a lower cost. These modifications must be simple enough to allow straightforward implementation on modern message passing parallel computers. In addition, the resulting method ought to be tested extensively with representative test cases.

Section 2 describes a representative two-dimensional system of fluid mechanics, namely, the lid driven cavity (LDC), which is subjected to analysis in the remaining sections. The corresponding 3-D system is the topic of [de Almeida and Derby, 1999]. The mathematical model is characterized by a single parameter, namely, the Reynolds number.

A classical method of approximate solution of the Navier–Stokes equations, namely, the Galerkin finite element method, is presented in section 3—the method is iterative owing to nonlinearity. The iterates satisfy a full linearization of the original equations. This iterative method, known as Newton's method, converges quadratically for a suitable initial guess, and it is the most well-known robust method for solving nonlinear systems of equations. Its cost, however, is high and is dominated by the factorization of a large and sparse Jacobian matrix at each iteration. Although a multitude of alternative methods [Gomes-Ruggiero, Martinez, and Moretti, 1992] appeared in the literature promising lower costs, we concluded that the complexity of the resultant algorithms hinders their message passing parallel implementation.

The purpose of section 3 is twofold. First, it presents Newton's method in the context of solving Galerkin's weak form of the Navier–Stokes equations for a fixed Reynolds number. The iterates are then approximated with the finite element method. Second, it advocates the use of an inner iteration within each (outer) Newton's iteration. Each inner iteration attempts to accelerate convergence by solving the linear system of equations with the existent factorization of the Jacobian matrix. The maximum number of inner iterations must cost only a prescribed fraction of a factorization and each inner iteration must reduce the magnitude of the iterates' correction. Otherwise, the inner iteration is interrupted and the outer iteration is resumed with a new factorization of the Jacobian matrix. It is shown that for the particular case of the Navier–Stokes equations, Newton's method combined with inner iterations (NMI) delivers higher convergence rates at a cost lower than that of Newton's method alone. This is no accident. According to [Brent, 1973], the method just described was first proposed by [Shamanskii, 1967] as a modification of Newton's method.

The method was severely tested on a state-of-the-art vector supercomputer, the Cray C-90, in order to determine its limitations in realistic large-scale numerical simulations. The geometry of the domain where the Navier–Stokes equations apply was chosen so that the underlying system of algebraic equations was the most challenging for current reordering methods. Nevertheless, identical convergence rates were ob-

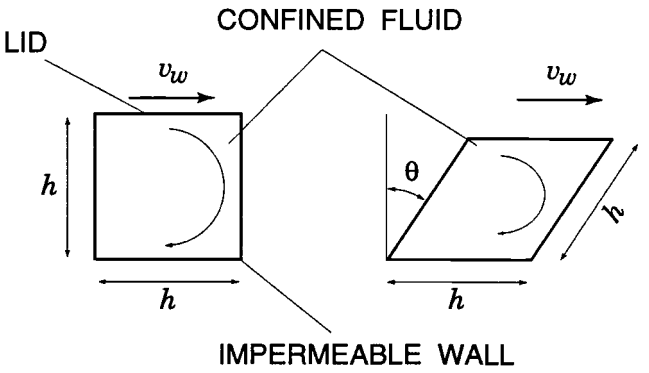


FIG. 2.1. *Lid driven cavity with rhombus geometry.*

tained for very refined distinct finite element partitions. It was also concluded that with the current powerful vector supercomputers and fast algorithms for factorization of sparse matrices, nonlinear systems of up to 1,000,000 algebraic equations resulting from the discretization of Navier–Stokes equations can be solved.

In section 4 a predictor-corrector method for construction of solution curves of the Navier–Stokes equations is put forward. The method consists of several simple modifications of the classical pseudoarclength continuation of [Keller, 1992] aimed at lowering cost while preserving robustness. The predictor step is an explicit Euler method. Because this first-order accurate method requires the solution of a linear system, it is only employed in curved portions of the solution curve. Otherwise, in flat portions of the solution curve, the predictor step is approximated by simple finite differences. Similarly, the corrector step with pseudoarclength parametrization of the solution curve is used only in curved portions of the solution curve, while the method described in section 3, NMI, is the corrector method of choice for flat portions of the solution curve.

A dramatic cost reduction was obtained by recycling an existent factorization of the Jacobian matrix along the solution curve. This was possible because of the generous radius of convergence of Newton's method when applied to the Navier–Stokes equations. The predictor-corrector method (section 4) was also subjected to extensive tests, with the LDC problem, (section 5) that constructed solution curves for Reynolds numbers up to 10,000 in the presence of simple singular points. The method performed outstandingly.

Last but not least, section 6 briefly discusses necessary modifications for the implementation of the algorithm on message passing parallel computers and extensions that allows the analysis of 3-D problems.

2. Governing equations of fluid motion in a cavity. The confined flow of an incompressible fluid in a lid driven cavity (LDC) was chosen as a representative system of fluid mechanics (Figure 2.1). Despite the system's simple boundary conditions and geometry, the fluid flow can be surprisingly nontrivial for certain speeds of the lid and/or some variations in the geometry of the cavity. In fact, the flow is characterized by multiple vortices, boundary layers with separation and reattachment, internal layers, and point singularities (see Figures 5.3 and 5.4).

Denoting Ω the region of two-dimensional Euclidean space occupied by the con-

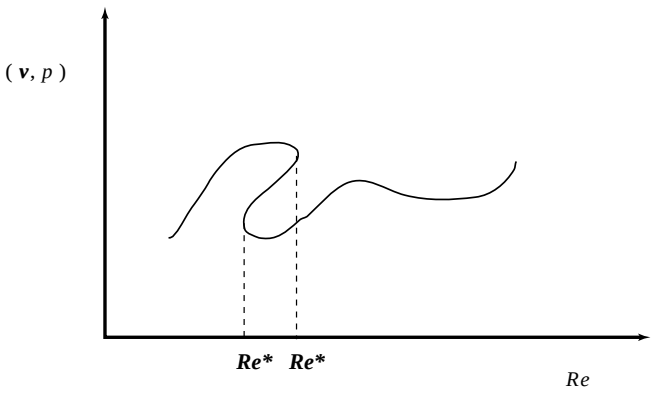


FIG. 2.2. *Sketch of solution curve.*

fined fluid in the cavity, the governing equations of steady-state motion of a Newtonian and incompressible fluid are the Navier–Stokes system,

$$
\begin{aligned}
&\text{(2.1a)} \qquad Re\ \nabla_{\boldsymbol{x}}\, \boldsymbol{v}\, \boldsymbol{v}(\boldsymbol{x}) = \operatorname{div}_{\boldsymbol{x}} \boldsymbol{T} \qquad \forall \quad \boldsymbol{x} \in \Omega\ ,\\
&\text{(2.1b)} \qquad \operatorname{div}_{\boldsymbol{x}} \boldsymbol{v} = 0 \qquad \forall \quad \boldsymbol{x} \in \Omega\ ,\\
&\text{(2.1c)} \qquad \boldsymbol{v}(\boldsymbol{x}) = \boldsymbol{v}_b(\boldsymbol{x}) \qquad \forall \quad \boldsymbol{x} \in \partial\Omega\,.
\end{aligned}
$$

Here $Re := \frac{\rho v_w h}{\mu}$ denotes the Reynolds number, where ρ is the density of the fluid and μ, its dynamic viscosity; $\boldsymbol{v}$ is the spatial velocity field in units of v_w;

$$\boldsymbol{T}(\boldsymbol{x}) := -p(\boldsymbol{x})\boldsymbol{I} + \left(\nabla_{\boldsymbol{x}}\, \boldsymbol{v} + \nabla_{\boldsymbol{x}}\, \boldsymbol{v}^{\mathsf{T}}\right)$$

is the stress in units of $\frac{\mu v_w}{h}$ at the point $\boldsymbol{x}$ in a Newtonian fluid; p is the pressure field modified by a conservative body force; and $\boldsymbol{v}_b$ is the boundary data for velocity—it is zero everywhere on $\partial\Omega$ except on the lid where its constant magnitude is one. Therefore, $\boldsymbol{v}_b$ satisfies the global mass balance compatibility condition.

A collection of solutions to (2.1) for different values of Re is best represented as a "curve" in an abstract solution space $(\boldsymbol{v}, p) \times Re$ (Figure 2.2). A parametrization of this curve with respect to the naturally appearing parameter Re,

$$\left(\boldsymbol{v}\,(\cdot, Re)\,, p\,(\cdot, Re)\right), \tag{2.2}$$

is inconvenient in virtue of the existence of simple singular points Re^*, where

$$\boldsymbol{v}'(\cdot, Re^*) := \frac{\partial \boldsymbol{v}}{\partial Re}(\cdot, Re^*) = \infty \quad \text{and} \quad p'(\cdot, Re^*) := \frac{\partial p}{\partial Re}(\cdot, Re^*) = \infty\ , \tag{2.3}$$

also known as turning points. Other possible singular points are those where $\boldsymbol{v}'(\cdot, Re^*)$ and $p'(\cdot, Re^*)$ are multivalued. In any case the singularity is often associated to the onset of multiple pairs, as indicated in Figure 2.2, for the same value of Re in (2.2). Nevertheless, this parametrization can be safely used away from such points inasmuch as a particular branch of the curve is unambiguously selected.

The solution curve of (2.1) for the LDC does exhibit singular points, particularly when the angle of tilt is not zero. The values of angle of tilt studied here are illustrated

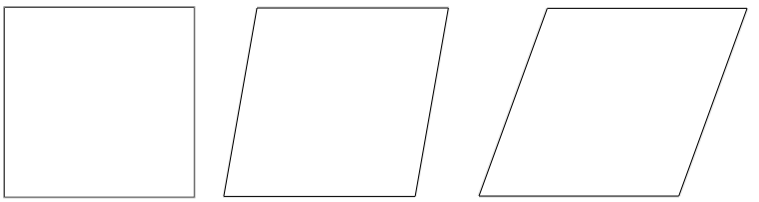


FIG. 2.3. *Angles of tilt examined.*

in Figure 2.3 (they all preserve the same aspect ratio of one). The reason for choosing this particular aspect ratio is explained in section 3.2.

An adequate parametric representation of a solution curve of (2.1) is obtained by introducing a reparametrization with a new parameter λ, that is,

$$\big(\boldsymbol{v}\,(\cdot, Re(\lambda))\,, p\,(\cdot, Re(\lambda))\big) \tag{2.4a}$$

or

$$\big(\widehat{\boldsymbol{v}}(\cdot,\lambda), \widehat{p}(\cdot,\lambda)\big), \tag{2.4b}$$

where $\widehat{\boldsymbol{v}}(\cdot,\lambda) := \boldsymbol{v}\,(\cdot, Re(\lambda))$ and $\widehat{p}(\cdot,\lambda) := p\,(\cdot, Re(\lambda))$. The reparametrization is defined (sections 4.1.2 and 4.2.2) so that simple singular points are eliminated.

This article is primarily concerned with showing cost-effective means of constructing (2.4) when solving (2.1) is extremely computationally intensive. Therefore the solution curves computed here are not used for an in-depth discussion of the physics of the LDC as modeled by (2.1).

3. Approximate solution to Galerkin's weak form. The most common weak form of (2.1) follows: find $(\boldsymbol{v}, p) \in \mathbf{V} \times \mathrm{L}^2_0(\Omega)$ such that

$$Re\ \mathrm{c}(\boldsymbol{v}\,,\boldsymbol{v}\,,\boldsymbol{u}) + \mathrm{a}(\boldsymbol{v}\,,\boldsymbol{u}) + \mathrm{b}(p\,,\boldsymbol{u}) = 0 \qquad \forall \quad \boldsymbol{u} \in \mathbf{H}^1_0(\Omega)\,, \tag{3.1a}$$

$$\mathrm{b}(q\,,\boldsymbol{v}) = 0 \qquad \forall \quad q \in \mathrm{L}^2_0(\Omega)\,, \tag{3.1b}$$

where

$$\mathrm{c}(\boldsymbol{w}\,,\boldsymbol{v}\,,\boldsymbol{u}) := \int_\Omega \nabla_{\boldsymbol{x}}\,\boldsymbol{w}\,\boldsymbol{v}(\boldsymbol{x}) \cdot \boldsymbol{u}(\boldsymbol{x})\,d\boldsymbol{x}\,, \quad \mathrm{a}(\boldsymbol{v}\,,\boldsymbol{u}) := \int_\Omega \left(\nabla_{\boldsymbol{x}}\,\boldsymbol{v} + \nabla_{\boldsymbol{x}}\,\boldsymbol{v}^{\mathsf{T}}\right) \bullet \nabla_{\boldsymbol{x}}\,\boldsymbol{u}\,d\boldsymbol{x}\,,$$

$$\mathrm{b}(p\,,\boldsymbol{u}) := \int_\Omega -p(\boldsymbol{x})\,\mathrm{div}_{\boldsymbol{x}}\,\boldsymbol{u}\,d\boldsymbol{x}\,,$$

$\mathbf{V} := \big\{\boldsymbol{v} \in \mathbf{H}^1(\Omega) \mid \boldsymbol{v} \equiv \boldsymbol{v}_b \text{ on } \partial\Omega\big\}$, and the spaces $\mathbf{H}^1(\Omega)$ and $\mathrm{L}^2_0(\Omega)$ are defined in the appendix. The inner product between two tensors is denoted $\boldsymbol{A} \bullet \boldsymbol{B}$.

3.1. Newton's method. A solution to (3.1) for a fixed Re can be found iteratively by computing a sequence $\big\{(\boldsymbol{v}^{(n)}, p^{(n)}) \in \mathbf{V} \times \mathrm{L}^2_0(\Omega) \mid n = 1, 2, \dots\big\}$, with a suitable initial guess for the velocity field $\boldsymbol{v}^{(0)} \in \mathbf{V}$—an initial guess for a pressure field can be arbitrary by virtue of the linearity of (3.1) with respect to p. Each iterate $(\boldsymbol{v}^{(n)}, p^{(n)})$ is computed by correcting its predecessor, $(\boldsymbol{v}^{(n-1)}, p^{(n-1)})$, with a

correction $(\boldsymbol{v}_c^{(n)}, p_c^{(n)}) \in \mathbf{H}_0^1(\Omega) \times \mathrm{L}_0^2(\Omega)$ so that

$$\boldsymbol{v}^{(n)} \equiv \boldsymbol{v}^{(n-1)} + \boldsymbol{v}_c^{(n)}\,, \tag{3.2a}$$

$$p^{(n)} \equiv p^{(n-1)} + p_c^{(n)}\,. \tag{3.2b}$$

The correction satisfies a linearization of (3.1),

(3.3a)

$$Re\Big\{\overbrace{\mathrm{c}\Big(\boldsymbol{v}_c^{(n)}\,,\,\boldsymbol{v}^{(n-1)}\,,\,\boldsymbol{u}\Big) + \mathrm{c}\Big(\boldsymbol{v}^{(n-1)}\,,\,\boldsymbol{v}_c^{(n)}\,,\,\boldsymbol{u}\Big)}^{\text{linearized convection}}\Big\} + \mathrm{a}\Big(\boldsymbol{v}_c^{(n)}\,,\,\boldsymbol{u}\Big) + \mathrm{b}\Big(p_c^{(n)}\,,\,\boldsymbol{u}\Big)$$

$$= -\Big\{\overbrace{Re\;\mathrm{c}\Big(\boldsymbol{v}^{(n-1)}\,,\,\boldsymbol{v}^{(n-1)}\,,\,\boldsymbol{u}\Big) + \mathrm{a}\Big(\boldsymbol{v}^{(n-1)}\,,\,\boldsymbol{u}\Big) + \mathrm{b}\Big(p^{(n-1)}\,,\,\boldsymbol{u}\Big)}^{\text{residual of (3.1a) at } (\boldsymbol{v}^{(n-1)},p^{(n-1)})}\Big\} \quad \forall\ \boldsymbol{u} \in \mathbf{H}_0^1(\Omega)\,,$$

(3.3b)

$$\mathrm{b}\Big(q\,,\,\boldsymbol{v}_c^{(n)}\Big) = -\overbrace{\mathrm{b}\Big(q\,,\,\boldsymbol{v}^{(n-1)}\Big)}^{\text{residual of (3.1b) at } \boldsymbol{v}^{(n-1)}} \qquad \forall \quad q \in \mathrm{L}_0^2(\Omega)\ .$$

The magnitude of the corrections, $\big\|\boldsymbol{v}_c^{(n)}\big\|_{\mathbf{H}^1(\Omega)}$ and $\big\|p_c^{(n)}\big\|_{\mathrm{L}^2(\Omega)}$, converges quadratically to zero in a few number of iterations, provided that the right and left sides of (3.3) are small enough for $n = 1$. The iterative method (3.2)–(3.3), known as Newton's method, is the most successful method for solving (3.1); however, it is also the most expensive. This investigation searches for simple modifications that substantially reduce the cost of the method without sacrificing its robustness. In fact the results obtained showed that it is possible to lower the cost while improving the convergence rate.

3.2. Finite dimensional approximation. It is natural to search for an approximate solution of (3.3), at the nth iteration, in a finite dimensional subspace $\mathbf{V}_c^h \times \mathrm{P}_c^h$ of $\mathbf{H}_0^1(\Omega) \times \mathrm{L}_0^2(\Omega)$, where h is a parameter associated to the dimension of the subspaces $\mathbf{V}_c^h$ and P_c^h. Let $\{\boldsymbol{u}_j \mid j = 1, 2, \ldots, N\}$ be a basis of $\mathbf{V}_c^h$ and $\{q_l \mid l = 1, 2, \ldots, M\}$, a basis of P_c^h; then the approximate solution $(\boldsymbol{v}_c^h, p_c^h)^{(n)}$ can be formally expressed as a linear combination of these bases,

$$\boldsymbol{v}_c^h \equiv \sum_{j=1}^{N} \alpha_j^{(n)}\, \boldsymbol{u}_j\,, \tag{3.4a}$$

$$p_c^h \equiv \sum_{l=1}^{M} \beta_l^{(n)}\, q_l\,, \tag{3.4b}$$

where $\alpha_j^{(n)}$ and $\beta_l^{(n)}$ are real coefficients. Similarly the test functions $\boldsymbol{u}$ and q in (3.3) can be represented in terms of the same bases—this is the Galerkin's method. After substitution of (3.4), and similar representations for $\boldsymbol{u}$ and q, into (3.3) and invoking the arbitrariness of $\boldsymbol{u}$ and q, one obtains a system of linear equations for the

coefficients α_j and β_l:

$$\sum_{j=1}^{N} A_{i,j}^{(n-1)}\,\alpha_j^{(n)} + \sum_{l=1}^{M} B_{l,i}\,\beta_l^{(n)} = -m_i^{(n-1)} \qquad i = 1, 2, \ldots, N, \tag{3.5a}$$

$$\sum_{j=1}^{N} B_{k,j}\,\alpha_j^{(n)} = -c_k^{(n-1)} \qquad k = 1, 2, \ldots, M, \tag{3.5b}$$

where

$$A_{i,j}^{(n-1)} := Re\Big\{\mathrm{c}\Big(\boldsymbol{u}_j\,,\,\boldsymbol{v}^{(n-1)}\,,\,\boldsymbol{u}_i\Big) + \mathrm{c}\Big(\boldsymbol{v}^{(n-1)}\,,\,\boldsymbol{u}_j\,,\,\boldsymbol{u}_i\Big)\Big\} + \mathrm{a}(\boldsymbol{u}_j\,,\,\boldsymbol{u}_i)\,, \tag{3.6a}$$

$$B_{k,j} := \mathrm{b}(q_k\,,\,\boldsymbol{u}_j)\,, \tag{3.6b}$$

$$m_i^{(n-1)} := Re\,\mathrm{c}\Big(\boldsymbol{v}^{(n-1)}\,,\,\boldsymbol{v}^{(n-1)}\,,\,\boldsymbol{u}_i\Big) + \mathrm{a}\Big(\boldsymbol{v}^{(n-1)}\,,\,\boldsymbol{u}_i\Big) + \mathrm{b}\Big(p^{(n-1)}\,,\,\boldsymbol{u}_i\Big)\,, \tag{3.6c}$$

$$c_k^{(n-1)} := \mathrm{b}\Big(q_k\,,\,\boldsymbol{v}^{(n-1)}\Big)\,. \tag{3.6d}$$

Remarks. In view of (3.2a) and (3.3b), $\boldsymbol{v}^{(n)}$ satisfies the weak mass balance (3.1b) for every iteration n—and so does $\boldsymbol{v}^{(n-1)}$. Therefore the residual of the mass balance $\mathrm{b}\big(q\,,\,\boldsymbol{v}^{(n-1)}\big)$ is zero for all n and $c_k^{(n-1)} = 0$ (up to round-off error).

The accuracy and stability of the approximation (3.4) depends upon the choice for the velocity and pressure correction spaces, $\mathbf{V}_c^h$ and P_c^h [Brezzi and Fortin, 1991]. A systematic method for choosing and constructing these spaces is provided by the finite element method. The choice made here employs piecewise biquadratic basis functions for velocity and linear piecewise discontinuous functions for pressure, both of them with compact support over the finite element; this is the so-called $Q_2{-}P_1$ mixed finite element.

The system of linear equations (3.5) can be put into a matrix form

$$\begin{pmatrix} \mathbf{A}^{(n-1)} & \mathbf{B}^{\mathsf{T}} \\ \mathbf{B} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \boldsymbol{\alpha}^{(n)} \\ \boldsymbol{\beta}^{(n)} \end{pmatrix} = - \begin{pmatrix} \mathbf{m}^{(n-1)} \\ \mathbf{c}^{(n-1)} \end{pmatrix} \tag{3.7}$$

for each iteration in (3.3). The elements of the submatrices $\mathbf{A}$ and $\mathbf{B}$, and the residual vectors $\mathbf{m}$ and $\mathbf{c}$ are exhibited in (3.6). $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are the vectors of coefficients α_j and β_l, respectively. Owing to the choice of the basis functions with compact support, $\mathbf{A}$ and $\mathbf{B}$ are extremely sparse. The density of nonzero elements in these matrices is typically on the order of 0.01% for large N and M. The method of choice to solve (3.7) is a triangular factorization of the coefficient matrix

$$\mathbf{J}^{(n-1)} := \begin{pmatrix} \mathbf{A}^{(n-1)} & \mathbf{B}^{\mathsf{T}} \\ \mathbf{B} & \mathbf{0} \end{pmatrix} \tag{3.8}$$

such that

$$\mathbf{P}^{(n-1)}\,\mathbf{J}^{(n-1)}\,\mathbf{Q}^{(n-1)} = \mathbf{L}^{(n-1)}\,\mathbf{U}^{(n-1)}\,, \tag{3.9}$$

where $\mathbf{P}$ and $\mathbf{Q}$ are, respectively, row and column permutation matrices; $\mathbf{L}$ is an upper triangular unit matrix and $\mathbf{U}$ is a lower triangular matrix. The factorization method employed here is similar to that of [Duff and Scott, 1993]. Typical performance of the method is shown in Table 3.1 as applied to the test cases of sections 3.4 and 5. Of

TABLE 3.1
Solution of $\mathbf{J}\,\mathbf{x} = \mathbf{b}$ *on a CRAY C-90 (one CPU). Floating point operation count is* 99% *accurate when compared to Cray Hardware Performance utility hpm.*

	Number of equations		
	110,802	441,602	992,402
Factorization CPU time (min)	1.05	14.33	66.02
Factorization MFLOPS	570	662	718
Forward substitution CPU time (s)	1.54	16.59	54.68
Backward substitution CPU time (s)	0.46	8.40	26.76
Storage of LU (Gbytes)	1.35	10.56	35.08

course $\mathbf{J}$ needs to be conveniently reordered [Sloan, 1989; Duff, Reid, and Scott, 1989; Karypis, 1996, see METIS] so that its triangular factors are also sparse. In view of the aspect ratio chosen for the LDC, the methods of [Sloan, 1989] and [Duff, Reid, and Scott, 1989] do not perform any better than simply reordering the finite elements along either the horizontal or vertical directions. The multilevel nested dissection method of METIS reduces the sparsity of the $\mathbf{L}$ and $\mathbf{U}$ factors by a factor of two, however, it's not unusual for the operation count of profile-based factorization methods to exhibit a tenfold increase. In summary, the coefficient matrix $\mathbf{J}$ associated to the aspect ratio chosen for the LDC is the worst possible case for profile-reduction reordering methods.

The solution of

$$\mathbf{J}^{(n-1)}\,\mathbf{x}^{(n)} = \mathbf{b}^{(n-1)}, \tag{3.10}$$

where

$$\mathbf{b}^{(n-1)} := -\begin{pmatrix}\mathbf{m}^{(n-1)}\\ \mathbf{c}^{(n-1)}\end{pmatrix} \quad \text{and} \quad \mathbf{x}^{(n)} := \begin{pmatrix}\boldsymbol{\alpha}^{(n)}\\ \boldsymbol{\beta}^{(n)}\end{pmatrix}, \tag{3.11}$$

is obtained by forward and backward substitutions, respectively,

$$\mathbf{L}^{(n-1)}\,\mathbf{y}^{(n)} = \mathbf{P}^{(n-1)}\,\mathbf{b}^{(n-1)} \quad \text{solve for } \mathbf{y}^{(n)}, \tag{3.12a}$$

$$\mathbf{U}^{(n-1)}\,\mathbf{z}^{(n)} = \mathbf{y}^{(n)} \quad \text{solve for } \mathbf{z}^{(n)}, \tag{3.12b}$$

$$\mathbf{x}^{(n)} = \mathbf{Q}^{(n-1)}\,\mathbf{z}^{(n)}\,. \tag{3.12c}$$

3.3. Inner iteration in Newton's method. The cost of factoring $\mathbf{J}$ is the most expensive part in computing a converged solution of (3.1). This cost can be drastically reduced by exploiting multiple forward/backward substitutions with the existing factors of $\mathbf{J}$ in order to reduce the residuals and the magnitude of the corrections in (3.3). Let

$$\mathbf{J}^{(n-1)}\,\mathbf{x}^{(n)} = \mathbf{b}^{(n-1)} \tag{3.13}$$

represent a finite approximation of (3.3) at the nth iteration. After a factorization of $\mathbf{J}^{(n-1)}$ is obtained, k inner iterations of forward/backward substitutions are performed:

$$\mathbf{L}^{(n-1)}\,\mathbf{y}^{(i)} = \mathbf{P}^{(n-1)}\,\mathbf{b}^{(i)}\,, \tag{3.14a}$$

$$\mathbf{U}^{(n-1)}\,\mathbf{z}^{(i)} = \mathbf{y}^{(i)}\,, \tag{3.14b}$$

$$\mathbf{x}^{(i)} = \mathbf{Q}^{(n-1)}\,\mathbf{z}^{(i)}, \quad i = 1, 2, \ldots, k\,. \tag{3.14c}$$

The number of inner iterations is determined by the total cost of (3.14) relative to the cost of the factorization of $\mathbf{J}^{(n-1)}$, provided that the magnitude of the corrections and residuals in (3.3) are monotonically decreasing during the inner iterations (3.14).

If the cost of factoring $\mathbf{J}^{(n-1)}$ is $\mathcal{C}$ and the cost of one iteration in (3.14) is $\mathcal{C}_i$, then the maximum number of inner iterations, $k_{\max}$, is computed by

$$k_{\max} = f\,\frac{\mathcal{C}}{\mathcal{C}_i},$$

where f is the fraction of $\mathcal{C}$ that (3.14) is allowed to cost. In the experiments reported in later sections, $f = 0.5$ is used and the resultant k is typically smaller than $k_{\max}$.

It is observed that the reasons why k is often smaller than $k_{\max}$ are either because the magnitude of the residuals and corrections passed the convergence criteria before k reached $k_{\max}$ or because the magnitude of the corrections or residuals increased in the kth inner iteration (sections 3.4 and 5). When the latter occurs, the solution $\mathbf{x}^{(k)}$ is discarded and $\mathbf{x}^{(k-1)}$ is used to factor $\mathbf{J}^{(n)}$ and to resume the $(n+1)$th outer iteration.

The convergence criteria of (3.3) are

$$\frac{\left\|\boldsymbol{v}_c^{(n)}\right\|_{\mathbf{H}^1(\Omega)}}{\left\|\boldsymbol{v}^{(n)}\right\|_{\mathbf{H}^1(\Omega)}} \leq 10^{-5}, \tag{3.15a}$$

$$\frac{\left\|p_c^{(n)}\right\|_{\mathrm{L}^2(\Omega)}}{\left\|p^{(n)}\right\|_{\mathrm{L}^2(\Omega)}} \leq 10^{-5}, \tag{3.15b}$$

$$\sqrt{\frac{\sum_{i=1}^{N}\left(m_i^{(n)}\right)^2}{N}} \leq 10^{-5}. \tag{3.15c}$$

Convergence is declared if all of the above are satisfied simultaneously.

Inner iterations within Newton's method is not a new idea. The method just described belongs to a class of high-order methods, that is, methods that converge q-superlinear with q-order larger than two. According to [Brent, 1973], Shamanskii [Shamanskii, 1967] first introduced the modification (3.14) in Newton's method. In fact, the q-order of convergence is exactly $k+1$. We were unaware of Shamanskii's work and it is our belief that the method has been rediscovered by others also [Fortin, Fortin, and Gervais, 1987; Fortin and Fortin, 1985].

3.4. Results and comments. The performance of Newton's method combined with inner iterations (NMI) is reported below. The method is employed to solve (3.1) with $\theta = 0$ for three values of Re, namely, 0, 250, and 500 and three finite element approximations: (a) 10,000 finite elements and 110,802 equations, (b) 40,000 finite elements and 441,602 equations, and (c) 90,000 finite elements and 992,402 equations. The finite elements are distributed uniformly on the cavity. The initial guess $\boldsymbol{v}^{(0)}$ for the case with $Re = 250$ was the velocity field obtained from $Re = 0$. Similarly, $\boldsymbol{v}^{(0)}$ for the case with $Re = 500$ was the solution obtained from $Re = 250$.

The convergence of NMI as a function of outer iterations is illustrated in Figures 3.1 and 3.2 and compared against Newton's method. It was observed that when the tolerances (3.15) for the magnitude of the correction fields were satisfied, the root-mean-square (3.15c) of the residual vector associated to the weak momentum

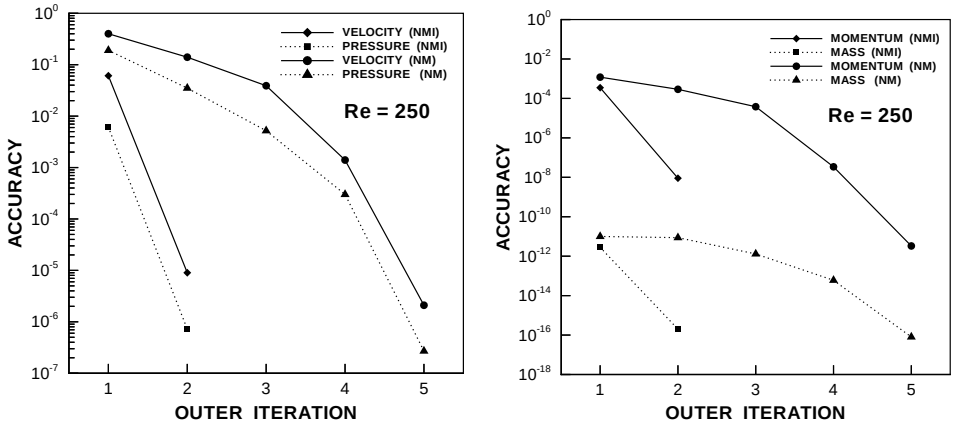


FIG. 3.1. *Comparison of Newton's method (NM) and its improvement by inner iterations (NMI).* (a) *Relative norms of velocity and pressure correction fields, respectively* (3.15a) *and* (3.15b)*, per outer iteration at* $Re = 250$ *with* $110,802$ *equations.* (b) *Corresponding root-mean-square of momentum and mass residuals.*

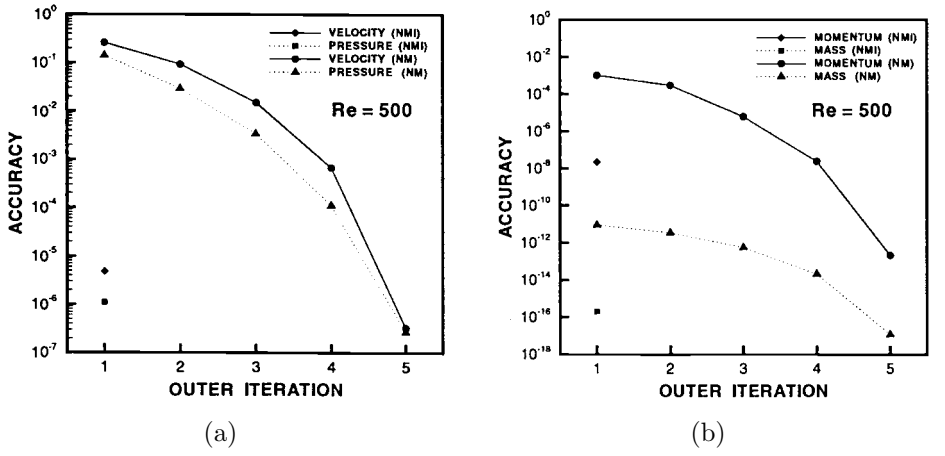


FIG. 3.2. *Comparison of Newton's method (NM) and its improvement by inner iterations (NMI).* (a) *Relative norms of velocity and pressure correction fields, respectively* (3.15a) *and* (3.15b)*, per outer iteration at* $Re = 500$ *with* $110,802$ *equations.* (b) *Corresponding root-mean-square of momentum and mass residuals.*

balance (3.3a) was on the order of 10^{-8} or less. Furthermore, regardless of what the tolerances in (3.15) were, the root-mean-square of the residual vector associated to the weak mass balance (3.3b) was on the order of 10^{-13} or less (this is in agreement with the Remarks in section 3.2). Therefore, it was concluded that the criteria in (3.15) were conservative and gave rise to well-converged solutions with respect to the fixed-point iteration (3.3).

The NMI (3.14) had an impressive higher rate of convergence than Newton's method alone with the bonus of *a lower cost* (Figure 3.3). The lower cost arose from the fact that the inner iterations were inexpensive—compared to the cost of a full factorization (Figure 3.4), yet they were sufficient to guarantee convergence of the

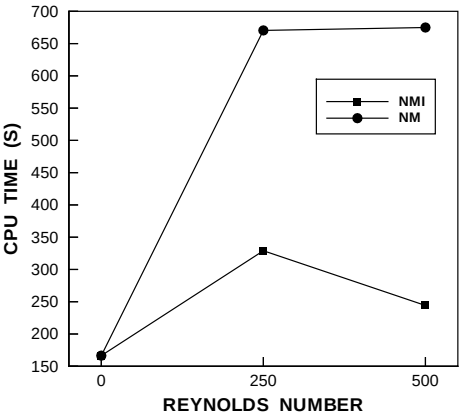


FIG. 3.3. *Comparison of CPU time in seconds (CRAY C-90) between Newton's method (NM) and its improvement by inner iterations (NMI) for the case with* $110,802$ *equations.*

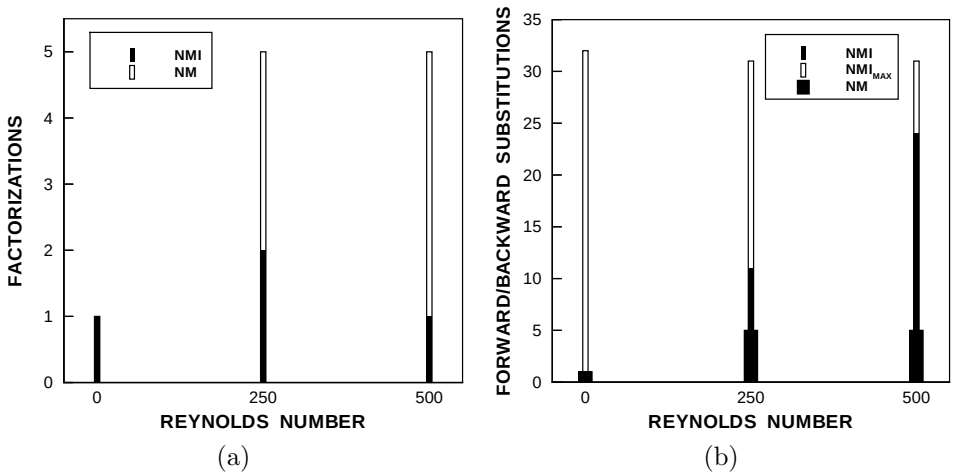


FIG. 3.4. *Comparison of Newton's method (NM) and its improvement by inner iterations (NMI).* (a) *Factorizations of* $\mathbf{J}$ *of dimension* $n \times n$ *at three values of Reynolds number;* $n = 110,802$. (b) *Corresponding number of forward/backward substitutions. Maximum number of substitutions allowed are represented with outlined bars.*

method.

Another observed property of the NMI was the invariance with respect to the finite element approximation. Convergence results for 441,602 and 992,402 equations were exactly the same as those obtained with 110,802 equations (see Appendix B in [de Almeida and Derby, 1997]). The most attractive result is shown in Figure 3.2, where one single factorization aided by many forward/backward substitutions which cost less than 50% of a factorization, sufficed to obtain accurate convergence. The result was independent of the finite element approximation (see Figures B.2 and B.5 in [de Almeida and Derby, 1997]).

From the foregoing, a natural question arises: Can one do better by recycling an existing factorization from a previous value of Re to a future value of Re? This question motivated the remaining part of this article which shows that in the context of

a predictor-corrector continuation method the inner iteration performs outstandingly. It was found that several solutions of (3.1) for different values of Re can be computed with a single factorization of $\mathbf{J}$ when Re is far from singular points Re^*, while, when Re is close to Re^*, one or two factorizations suffice to obtain convergence.

4. A predictor-corrector method. A class of methods for construction of solution curves (2.2) or (2.4) is known under the generic name of predictor-corrector methods or simply continuation methods.

Let $(\boldsymbol{v}_{(m-1)}, p_{(m-1)})$ be a solution of (3.1) for a particular value of $Re = Re_{(m-1)}$, where m is an index varying from one to a prescribed number of solution points. A solution curve of (3.1) through $(\boldsymbol{v}_{(m-1)}, p_{(m-1)})$ can be constructed via a two-step approach. The first step predicts a solution at a new value of the Reynolds number, $Re^{(0)}_{(m)}$, typically by marching tangentially to the solution curve. The second step corrects the predicted solution, $Re^{(0)}_{(m)}$, iteratively until a converged solution, $Re_{(m)}$, is found.

The predictor-corrector method proposed below employs two different parametrizations of the solution curve, namely, *natural* and *pseudoarclength* parametrizations. The former is used away from singular points of the solution curve and the latter is used in the vicinity of simple singular points. The resultant method combines the robustness of the pseudoarclength parametrization with the inexpensiveness of the natural parametrization.

4.1. First-order predictor step. The most common predictor method requires the computation of the tangent vector to the solution curve at $(\boldsymbol{v}_{(m-1)}, p_{(m-1)})$. This constitutes a first-order predictor method. Other higher order methods exist but they are not cost-effective for the Navier–Stokes problem treated here.

4.1.1. Natural parametrization. Denoting the tangent vector to the solution curve illustrated in Figure 2.2

$$\big(1, (\boldsymbol{v}'_{(m-1)}, p'_{(m-1)})\big), \tag{4.1}$$

where $\boldsymbol{v}'_{(m-1)} := \boldsymbol{v}'(\cdot, Re_{(m-1)}) := \frac{\partial \boldsymbol{v}}{\partial Re}(\cdot, Re_{(m-1)})$ and $p'_{(m-1)} := p'(\cdot, Re_{(m-1)}) := \frac{\partial p}{\partial Re}(\cdot, Re_{(m-1)})$, and assuming that the tangent exists, then the tangent vector must satisfy

$$Re_{(m-1)}\Big\{\overbrace{\mathrm{c}\big(\boldsymbol{v}'_{(m-1)}\,,\,\boldsymbol{v}_{(m-1)}\,,\,\boldsymbol{u}\big) + \mathrm{c}\big(\boldsymbol{v}_{(m-1)}\,,\,\boldsymbol{v}'_{(m-1)}\,,\,\boldsymbol{u}\big)}^{\text{linearized convection}}\Big\} + \mathrm{a}\big(\boldsymbol{v}'_{(m-1)}\,,\,\boldsymbol{u}\big)$$
$$+\,\mathrm{b}\Big(p'_{(m-1)}\,,\,\boldsymbol{u}\Big) = -\overbrace{\mathrm{c}\big(\boldsymbol{v}_{(m-1)}\,,\,\boldsymbol{v}_{(m-1)}\,,\,\boldsymbol{u}\big)}^{\text{partial derivative of (3.1a) w.r.t } Re} \qquad \forall\ \boldsymbol{u} \in \mathbf{H}^1_0(\Omega)\,, \tag{4.2a}$$

$$\mathrm{b}\big(q\,,\,\boldsymbol{v}'_{(m-1)}\big) = 0 \quad \forall \quad q \in \mathrm{L}^2_0(\Omega)\,, \tag{4.2b}$$

which is the total derivative of (3.1) with respect to Re, evaluated at $(\boldsymbol{v}_{(m-1)}, p_{(m-1)}, Re_{(m-1)})$. Note that the solution pair of (4.2), $(\boldsymbol{v}'_{(m-1)}, p'_{(m-1)})$, belongs to $\mathbf{H}^1_0(\Omega) \times \mathrm{L}^2_0(\Omega)$ (see also 4.6).

A finite-dimensional approximation of (4.2) can be readily obtained by following the contents of section 3.2, which leads to a system of linear equations,

$$\begin{pmatrix} \mathbf{A} & \mathbf{B}^{\mathsf{T}} \\ \mathbf{B} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \overline{\boldsymbol{\alpha}} \\ \overline{\boldsymbol{\beta}} \end{pmatrix} = - \begin{pmatrix} \overline{\mathbf{m}} \\ \mathbf{0} \end{pmatrix}, \tag{4.3}$$

where $\overline{\boldsymbol{\alpha}}$ and $\overline{\boldsymbol{\beta}}$ are the vectors of coefficients of the finite dimensional approximations of $\boldsymbol{v}'$ and p', respectively. The elements of the vector $\overline{\mathbf{m}}$ are

$$\overline{m}_i := \mathrm{c}\big(\boldsymbol{v}_{(m-1)}\,,\, \boldsymbol{v}_{(m-1)}\,,\, \boldsymbol{u}_i\big)\ . \tag{4.4}$$

It is clear that the matrix of coefficients in (4.3) is equal to the matrix $\mathbf{J}$ of (3.7) evaluated at $(\boldsymbol{v}_{(m-1)}, p_{(m-1)}, Re_{(m-1)})$. If a factorization of $\mathbf{J}$ at this solution point is available, the tangent vector to the solution curve at this point can be computed at the cost of one forward/backward substitution. If a factorization of $\mathbf{J}$ is not available, then solving (4.2) should be avoided by computing an approximate tangent vector,

$$\big(1, (\boldsymbol{v}'_{(m-1)}, p'_{(m-1)})\big) \approx \left(1, \Big(\frac{\boldsymbol{v}_{(m-1)} - \boldsymbol{v}_{(m-2)}}{Re_{(m-1)} - Re_{(m-2)}}, \frac{p_{(m-1)} - p_{(m-2)}}{Re_{(m-1)} - Re_{(m-2)}}\Big)\right), \tag{4.5}$$

given by the secant that joins the previous solution point $(\boldsymbol{v}_{(m-2)}, p_{(m-2)}, Re_{(m-2)})$ to the current point. This approximation often works well in practice inasmuch as the curvature of the solution curve is small. In fact, (4.5) is the method of choice, introduced here to compute the tangent direction. An accurate tangent is computed by solving (4.3) only if a factorization of $\mathbf{J}$ is already available or when the corrector step demands for it (section 4.3).

An approximation to the next solution point on the curve is then obtained by

$$\boldsymbol{v}_{(m)} \equiv \boldsymbol{v}_{(m-1)} + \Delta Re_{(m-1)}\, \boldsymbol{v}'_{(m-1)} \quad \text{and} \tag{4.6a}$$

$$p_{(m)} \equiv p_{(m-1)} + \Delta Re_{(m-1)}\, p'_{(m-1)}\,, \tag{4.6b}$$

where $\Delta Re_{(m-1)}$ is a prescribed steplength of Re (section 4.3) at the $(m-1)$th continuation step (see Figure 4.1).

4.1.2. Arclength parametrization. As mentioned in section 2, a parametrization of the solution curve by Re is inconvenient in the neighborhood of singular points Re^*. It was suggested that a reparametrization should take place such that the solution curve can be constructed beyond those points. It is proposed here that the reparametrization should occur when the curvature of the solution curve reaches a threshold. Owing to the high cost of computing the curvature of the solution curve, it is recommended that the angle α between successive tangents along the solution curve be computed alternatively. Thus, in the experiments reported below, when

$$\alpha := \arccos\left(\frac{1 + \boldsymbol{v}'_{(m-1)} \cdot \boldsymbol{v}'_{(m-2)} + p'_{(m-1)}\, p'_{(m-2)}}{\sqrt{1 + \big\|\boldsymbol{v}'_{(m-1)}\big\|^2_{\mathbf{H}^1(\Omega)} + \big\|p'_{(m-1)}\big\|^2_{\mathrm{L}^2(\Omega)}}\sqrt{1 + \big\|\boldsymbol{v}'_{(m-2)}\big\|^2_{\mathbf{H}^1(\Omega)} + \big\|p'_{(m-2)}\big\|^2_{\mathrm{L}^2(\Omega)}}}\right) \tag{4.7}$$

is greater than 10°, the solution curve is automatically reparametrized by λ (section 2) and the construction of the curve is resumed immediately. If α becomes smaller than 10°, then the parametrization of the solution curve is reverted to the natural

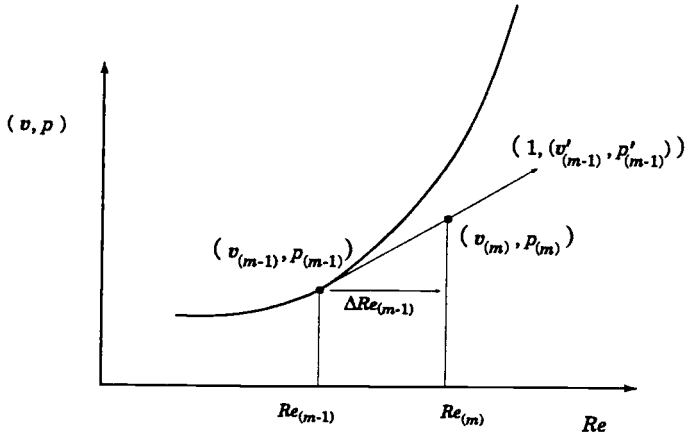


FIG. 4.1. *Predictor step with solution curve parametrized by the natural parameter Re.*

parametrization. Note that in (4.7)

$$\boldsymbol{v}'_{(m-1)} \cdot \boldsymbol{v}'_{(m-2)} := \int_\Omega \boldsymbol{v}'_{(m-1)}(\boldsymbol{x}) \cdot \boldsymbol{v}'_{(m-2)}(\boldsymbol{x})\, d\boldsymbol{x} \quad \text{and} \tag{4.8a}$$

$$p'_{(m-1)}\, p'_{(m-2)} := \int_\Omega p'_{(m-1)}(\boldsymbol{x})\, p'_{(m-2)}(\boldsymbol{x})\, d\boldsymbol{x}\,. \tag{4.8b}$$

The tangent vector to the solution curve in terms of λ is obtained by the chain rule of differentiation,

$$\dot{\boldsymbol{v}}_{(m-1)} := \dot{\boldsymbol{v}}(\cdot, \lambda_{(m-1)}) := \frac{\partial \widehat{\boldsymbol{v}}}{\partial \lambda}(\cdot, \lambda_{(m-1)}) = \boldsymbol{v}'_{(m-1)}\, \dot{Re}_{(m-1)}, \tag{4.9a}$$

$$\dot{p}_{(m-1)} := \dot{p}(\cdot, \lambda_{(m-1)}) := \frac{\partial \widehat{p}}{\partial \lambda}(\cdot, \lambda_{(m-1)}) = p'_{(m-1)}\, \dot{Re}_{(m-1)}, \tag{4.9b}$$

where $\dot{Re}_{(m-1)} := \frac{dRe}{d\lambda_{(m-1)}}$, with the independent assumption of

$$\left\|\dot{\boldsymbol{v}}_{(m-1)}\right\|^2_{\mathbf{H}^1(\Omega)} + \left\|\dot{p}_{(m-1)}\right\|^2_{\mathrm{L}^2(\Omega)} + \dot{Re}^2_{(m-1)} = 1\,, \tag{4.10}$$

necessary to determine $\dot{Re}$ uniquely (modulo a sign). Because the magnitude of the tangent as a function of λ is one, λ is the so-called arclength parameter.

Solving for $\dot{Re}_{(m-1)}$ in (4.9)–(4.10) yields

$$\dot{Re}_{(m-1)} = \frac{\pm 1}{\sqrt{1 + \left\|\boldsymbol{v}'_{(m-1)}\right\|^2_{\mathbf{H}^1(\Omega)} + \left\|p'_{(m-1)}\right\|^2_{\mathrm{L}^2(\Omega)}}}\,. \tag{4.11}$$

According to Keller's recommendation [Keller, 1992], the correct choice of the sign in (4.11) is the one that delivers $0 \le \alpha \le \frac{\pi}{2}$ (see Figure 4.2) so that the construction of the solution curve moves "forward." Therefore,

$$\left(\dot{Re}_{(m-1)}, \left(\dot{\boldsymbol{v}}_{(m-1)}, \dot{p}_{(m-1)}\right)\right) \cdot \left(\dot{Re}_{(m-2)}, \left(\dot{\boldsymbol{v}}_{(m-2)}, \dot{p}_{(m-2)}\right)\right) > 0\,,$$

$$\dot{Re}_{(m-1)} \left(1, \left(\boldsymbol{v}'_{(m-1)}, p'_{(m-1)}\right)\right) \cdot \left(\dot{Re}_{(m-2)}, \left(\dot{\boldsymbol{v}}_{(m-2)}, \dot{p}_{(m-2)}\right)\right) > 0\,.$$

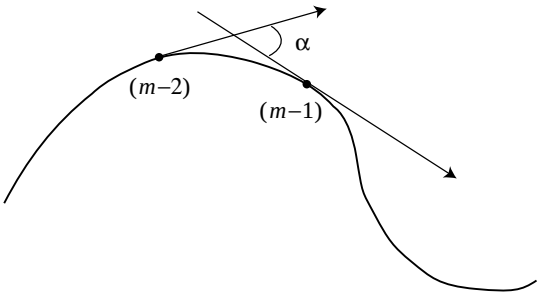


FIG. 4.2. *Correct choice of the angle α between successive tangent vectors on the solution curve.*

Thus,

$$\operatorname{sign}\big(\dot{Re}_{(m-1)}\big) = \operatorname{sign}\big(\dot{Re}_{(m-2)} + \boldsymbol{v}'_{(m-1)} \cdot \dot{\boldsymbol{v}}_{(m-2)} + p'_{(m-1)}\, \dot{p}_{(m-2)}\big)\,.$$

Substituting the above into (4.11), the working formula to find $\dot{Re}_{(m-1)}$ is

$$\boxed{\dot{Re}_{(m-1)} = \frac{\operatorname{sign}\big(\dot{Re}_{(m-2)} + \boldsymbol{v}'_{(m-1)} \cdot \dot{\boldsymbol{v}}_{(m-2)} + p'_{(m-1)}\, \dot{p}_{(m-2)}\big)}{\sqrt{1 + \big\|\boldsymbol{v}'_{(m-1)}\big\|^2_{\mathbf{H}^1(\Omega)} + \big\|p'_{(m-1)}\big\|^2_{\mathrm{L}^2(\Omega)}}}\,.} \tag{4.12}$$

In view of (4.9) and (4.12), a first-order approximation in λ to the mth solution point is simply

$$\boldsymbol{v}_{(m)} \equiv \boldsymbol{v}_{(m-1)} + \Delta\lambda_{(m-1)}\, \dot{\boldsymbol{v}}_{(m-1)}\,, \tag{4.13a}$$

$$p_{(m)} \equiv p_{(m-1)} + \Delta\lambda_{(m-1)}\, \dot{p}_{(m-1)}, \quad \text{and} \tag{4.13b}$$

$$Re_{(m)} = Re_{(m-1)} + \Delta\lambda_{(m-1)}\, \dot{Re}_{(m-1)}\,, \tag{4.13c}$$

where $\Delta\lambda_{(m-1)}$ is the steplength of the pseudoarclength parameter at the mth continuation step (section 4.3).

4.2. Corrector step. The role of the corrector is to solve (3.1) for a given initial guess obtained in the predictor step. The corrector methods described below differ on the kind of parametrization employed.

4.2.1. Natural parametrization. Given the initial guess $(\boldsymbol{v}^{(0)}_{(m)}, p^{(0)}_{(m)})$ at $Re_{(m)}$, obtained with the predictor method of section 4.1.1 (4.6), a solution of (3.1) can be computed (sections 3.1 and 3.2). An illustration of the predictor-corrector process for the mth continuation step is presented in Figure 4.3, where $(\boldsymbol{v}_{(m)}, p_{(m)})$ denotes a converged solution. This corrector method will fail in the vicinity of simple singular points such as turning points because the direction of search for a solution is perpendicular to the abscissa of the graph.

4.2.2. Pseudoarclength parametrization. A widely used corrector method [Keller, 1992] to construct solution curves in the neighborhood of turning points and other simple singular points consists of searching for a solution of (3.1) by constraining (3.3) such that the iterates $(\boldsymbol{v}^{(n)}_{(m)}, p^{(n)}_{(m)}, Re^{(n)}_{(m)})$ lie in the direction perpendicular to

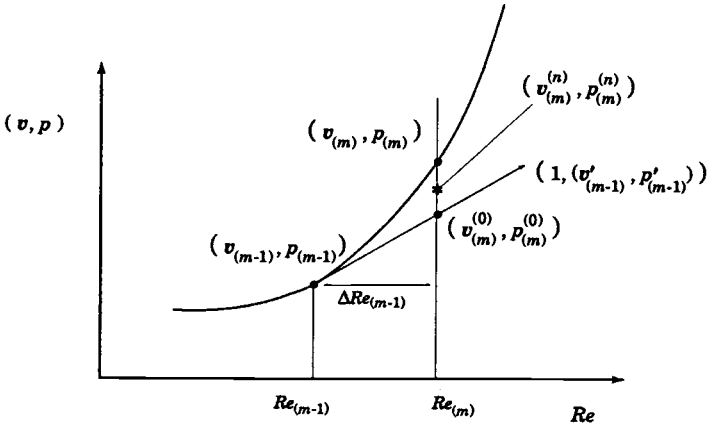


FIG. 4.3. *Predictor-corrector step with natural parametrization.*

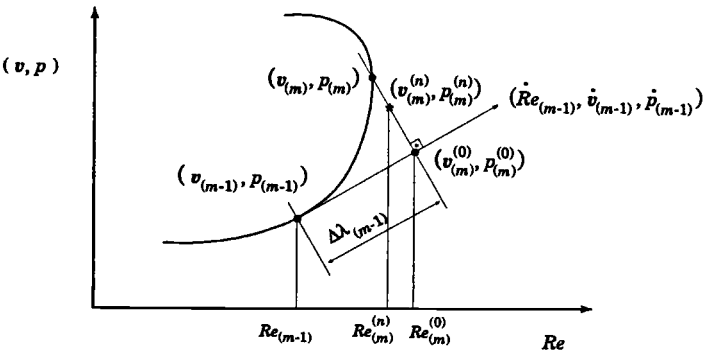


FIG. 4.4. *Predictor-corrector step with pseudoarclength parametrization.*

the tangent vector (see Figure 4.4). Therefore,

$$\left(Re^{(n)}_{(m)} - Re_{(m-1)}\right)\dot{Re}_{(m-1)} + \left(\boldsymbol{v}^{(n)}_{(m)} - \boldsymbol{v}_{(m-1)}\right)\cdot\dot{\boldsymbol{v}}_{(m-1)} + \left(p^{(n)}_{(m)} - p_{(m-1)}\right)\dot{p}_{(m-1)} = \Delta\lambda_{(m-1)}\,. \tag{4.14}$$

It is clear that neither the converged solution nor any of the iterates $(\boldsymbol{v}^{(n)}_{(m)}, p^{(n)}_{(m)}, Re^{(n)}_{(m)})$ satisfies the arclength condition, that is,

$$\sqrt{\left\|\dot{\boldsymbol{v}}^{(n)}_{(m)}\right\|^2_{\mathbf{H}^1(\Omega)} + \left\|\dot{p}^{(n)}_{(m)}\right\|^2_{\mathrm{L}^2(\Omega)} + \left(\dot{Re}^{(n)}_{(m)}\right)^2} \neq 1 \quad \forall\, n\,.$$

For this reason, λ is not the arclength parameter. However, because $\Delta\lambda$ is measured along the tangent direction on the solution curve, λ is known in the literature as the "pseudoarclength parameter." The distinction must be made clear whenever the context generates confusion. Although rarely used, a genuine arclength parameter continuation merits attention.

The mth corrector step is as follows: compute the sequence $\big\{(\boldsymbol{v}_{(m)}^{(n)}, p_{(m)}^{(n)}, Re_{(m)}^{(n)}) \in \mathbf{V} \times \mathrm{L}_0^2(\Omega) \times \mathbb{R} \mid n = 1, 2, \dots\big\}$ for a given predictor guess $(\boldsymbol{v}_{(m)}^{(0)}, p_{(m)}^{(0)}, Re_{(m)}^{(0)})$ such that

$$\boldsymbol{v}_{(m)}^{(n)} \equiv \boldsymbol{v}_{(m)}^{(n-1)} + \boldsymbol{v}_c^{(n)}\,, \tag{4.15a}$$

$$p_{(m)}^{(n)} \equiv p_{(m)}^{(n-1)} + p_c^{(n)}\,, \tag{4.15b}$$

$$Re_{(m)}^{(n)} \equiv Re_{(m)}^{(n-1)} + Re_c^{(n)}\,, \tag{4.15c}$$

where the correction $(\boldsymbol{v}_c^{(n)}, p_c^{(n)}, Re_c^{(n)}) \in \mathrm{H}_0^1(\Omega) \times \mathrm{L}_0^2(\Omega) \times \mathbb{R}$ satisfies

(4.16a)
$$\begin{aligned} Re_{(m)}^{(n-1)} \Big\{ \mathrm{c}\big(\boldsymbol{v}_c^{(n)}\,,\, \boldsymbol{v}_{(m)}^{(n-1)}\,,\, \boldsymbol{u}\big) + \mathrm{c}\big(\boldsymbol{v}_{(m)}^{(n-1)}\,,\, \boldsymbol{v}_c^{(n)}\,,\, \boldsymbol{u}\big) \Big\} + \mathrm{a}\big(\boldsymbol{v}_c^{(n)}\,,\, \boldsymbol{u}\big) + \mathrm{b}\big(p_c^{(n)}\,,\, \boldsymbol{u}\big) \\ + Re_c^{(n)}\, \mathrm{c}\big(\boldsymbol{v}_{(m)}^{(n-1)}\,,\, \boldsymbol{v}_{(m)}^{(n-1)}\,,\, \boldsymbol{u}\big) = -\Big\{ Re_{(m)}^{(n-1)}\, \mathrm{c}\big(\boldsymbol{v}_{(m)}^{(n-1)}\,,\, \boldsymbol{v}_{(m)}^{(n-1)}\,,\, \boldsymbol{u}\big) \\ + \mathrm{a}\big(\boldsymbol{v}_{(m)}^{(n-1)}\,,\, \boldsymbol{u}\big) + \mathrm{b}\big(p_{(m)}^{(n-1)}\,,\, \boldsymbol{u}\big) \Big\} \quad \forall\ \boldsymbol{u} \in \mathbf{H}_0^1(\Omega)\,, \end{aligned}$$

(4.16b)
$$\mathrm{b}\big(q\,,\, \boldsymbol{v}_c^{(n)}\big) = -\mathrm{b}\big(q\,,\, \boldsymbol{v}_{(m)}^{(n-1)}\big) \quad \forall \quad q \in \mathrm{L}_0^2(\Omega)\,,$$

(4.16c)
$$\begin{aligned} \boldsymbol{v}_c^{(n)} \cdot \dot{\boldsymbol{v}}_{(m-1)} + p_c^{(n)}\, \dot{p}_{(m-1)} + Re_c^{(n)}\, \dot{Re}_{(m-1)} \\ = \Delta\lambda_{(m-1)} - \Big\{ \big(\boldsymbol{v}_{(m)}^{(n-1)} - \boldsymbol{v}_{(m-1)}\big) \cdot \dot{\boldsymbol{v}}_{(m-1)} \\ + \big(p_{(m)}^{(n-1)} - p_{(m-1)}\big)\, \dot{p}_{(m-1)} + \big(Re_{(m)}^{(n-1)} - Re_{(m-1)}\big)\, \dot{Re}_{(m-1)} \Big\}\,. \end{aligned}$$

The magnitude of the corrections $\big\|\boldsymbol{v}_c^{(n)}\big\|_{\mathbf{H}^1(\Omega)}$, $\big\|p_c^{(n)}\big\|_{\mathrm{L}^2(\Omega)}$, and $\big|Re_c^{(n)}\big|$ converge quadratically to zero in a few number of iterations, provided that the right and left sides of (4.16) are small enough for $n = 1$.

A finite dimensional approximation of (4.16) (section 3.2) leads to a linear system of equations with a sparse matrix of coefficients,

$$\begin{pmatrix} \mathbf{J} & \mathbf{d} \\ \mathbf{a} & \dot{Re} \end{pmatrix} \begin{pmatrix} \mathbf{x} \\ Re_c \end{pmatrix} = \begin{pmatrix} \mathbf{b} \\ N \end{pmatrix}\,, \tag{4.17}$$

for each iteration n in (4.16), holding $m-1$ fixed, where

$$\begin{aligned} d_i &:= \mathrm{c}\big(\boldsymbol{v}_{(m)}^{(n-1)}\,,\, \boldsymbol{v}_{(m)}^{(n-1)}\,,\, \boldsymbol{u}_i\big)\,, \\ \mathbf{a} &:= \dot{Re}_{(m-1)}\, (\overline{\boldsymbol{\alpha}}_{(m-1)}, \overline{\boldsymbol{\beta}}_{(m-1)})^{\mathsf{T}} \text{ recall (4.3), (4.9) and (4.16c)}, \\ \mathbf{x} &:= (\boldsymbol{\alpha}^{(n)}, \boldsymbol{\beta}^{(n)})^{\mathsf{T}} \text{ recall (3.7) and (4.16a)}, \\ N &:= \Delta\lambda_{(m-1)} - \Big\{ \big(\boldsymbol{v}_{(m)}^{(n-1)} - \boldsymbol{v}_{(m-1)}\big) \cdot \dot{\boldsymbol{v}}_{(m-1)} \\ &\quad + \big(p_{(m)}^{(n-1)} - p_{(m-1)}\big)\, \dot{p}_{(m-1)} + \big(Re_{(m)}^{(n-1)} - Re_{(m-1)}\big)\, \dot{Re}_{(m-1)} \Big\}\,. \end{aligned}$$

The method of choice for solving (4.17) consists of a triangular factorization of $\mathbf{J}$ followed by block elimination. Thus,

$$\mathbf{J}\,\mathbf{v} = \mathbf{b} \qquad \text{solve for } \mathbf{v}, \tag{4.18a}$$

$$\mathbf{J}\,\mathbf{w} = \mathbf{d} \qquad \text{solve for } \mathbf{w}, \tag{4.18b}$$

$$Re_c = \frac{N - \mathbf{a}\cdot\mathbf{v}}{\dot{R}e - \mathbf{a}\cdot\mathbf{w}}\,, \tag{4.18c}$$

$$\mathbf{x} = \mathbf{v} - Re_c\,\mathbf{w}\,. \tag{4.18d}$$

Compared to (3.12), the above is twice as expensive in virtue of an extra forward-backward substitution. The strategy and convergence criteria to solve (4.17) is identical to that described in sections 3.2–3.3. Here, however, the cost of a factorization of $\mathbf{J}$ is greatly amortized (section 5) along the solution curve. That is, the $\mathbf{L}$ and $\mathbf{U}$ factors of $\mathbf{J}$ are recycled in several/many continuation steps inasmuch as the convergence criteria and the cost criterion are satisfied.

The predictor-corrector method based on the pseudoarclength parameter requires the evaluation of a number of scalar and vector products of field quantities (4.16c). In addition the angle α (4.7) needs to be computed at every predictor step. Frequent evaluation of the inner products (4.8) incurs extra costs that can be as high as, or higher than, as the cost of a forward-backward substitution. Therefore, an effort must be made to compute these products efficiently.

4.3. Steplength control and corrector failure measures. An efficient steplength strategy is vital to parameter analysis. If the steplength is too small, only a small portion of the solution curve is constructed. On the other hand, too large a steplength hinders the convergence of the corrector step. Steplength control in predictor-corrector methods of parameter analysis is typically done by observing the convergence rate of the corrector step during the computation of past solution points [Gunzburger and Peterson, 1991; Allgower and Georg, 1990; Bank and Mittelmann, 1989; Schwetlick and Cleve, 1987; Deuflhard, Fiedler, and Kunkel, 1987; Heijer and Rheinboldt, 1981; Rheinboldt, 1980].

Presumably, the collected information helps to estimate a priori the radius of convergence of the corrector step in the next solution point. One drawback of this approach is that it often fails in the vicinity of turning points because the estimates of the radius of convergence are based on portions of the solution curve that are relatively flat. In practice, one resorts to heuristic methods for adjusting the steplength. For instance, a heuristic steplength control can be designed and improved in the process of mesh refinement.

The Navier–Stokes equations have a generous radius of convergence in large portions of the solution curve up to the proximity of turning points (section 5). Only very close to a turning point, the radius of convergence diminishes dramatically. This behavior makes it difficult to make reliable estimates of the radius of convergence near turning points.

The heuristic steplength control advocated here enforces a monotonic rate of convergence on the corrector step. That is, the magnitude of the norms in (3.15), in addition to $\left|Re_c^{(n)}\right|$ in the case of the pseudo arclength parametrization, must decrease for each outer iteration n in the corrector step. Otherwise, a corrector failure is declared. The steplength is reduced by a factor of four and an accurate computation of the tangent direction is made by factoring $\mathbf{J}$ and solving (4.3). A maximum number of three failures is allowed in the corrector step before the iterations are interrupted.

If the corrector step converges without requiring a factorization of **J**, the steplength is increased by a factor of two. The natural parameter steplength is computed and compared to an upperbound. A valid steplength is the smallest value between the upperbound and the increased stepsize. In the results shown in section 5, corrector failures occur most often for solution points close to turning points. The amount of reduction or increase of the parameter steplength is problem dependent and also important to keep the method's cost low. Note that several corrector failures will incur a prohibitive cost to the predictor-corrector method. Each failure adds to the existing expenses the cost of a factorization of **J** used to compute an accurate tangent. The situation is not so critical because, as shown in section 5, the cost of factoring **J** is amortized in the corrector iterations.

4.4. Transition between parametrizations. When the angle α (4.7) between two successive tangent vectors in the solution curve is greater than 10°, the pseudoarclength parametrization of the curve is employed (sections 4.1.2 and 4.2.2). Otherwise, the natural parametrization is the method of choice. When a transition from the natural parametrization to the pseudoarclength parametrization occurs, the initial pseudoarclength parameter steplength is selected by

$$\Delta\lambda_{(m-1)} = \frac{\Delta Re_{(m-2)}}{\dot{Re}_{(m-1)}}, \tag{4.19}$$

where $(m-2)$ is the continuation step where $\alpha > 10°$ occurred. When the converse takes place, the initial natural parameter steplength is selected by

$$\Delta Re_{(m-1)} = \Delta\lambda_{(m-2)}\, \dot{Re}_{(m-2)}\,. \tag{4.20}$$

In this last case the initial natural parameter steplength is considered valid only if it is less than or equal to an upperbound. In the experiments reported below the upperbound is 100.

5. Results and discussion. Solution curves to the LDC problem (section 2) possessing 100 solution points were constructed by the methods presented in sections 3 and 4. Employing the heuristic steplength control described in section 4.3, it was found that the Reynolds number ranged between 0 and 10,000. The results were accurate with respect to the finite element approximations shown in Figure 5.1, and also with respect to the fixed-point iterations (Figures 5.6 and 5.8). Results obtained with $\theta = 20°$ and 40,000 finite elements (441,602 equations) are identical to the results presented below (see Appendix C in [de Almeida and Derby, 1997]).

The kinetic energy of the system $\mathcal{K}$, measured in units of $\frac{1}{2}\,\rho\, v_w^2\, h^2$, is defined as

$$\mathcal{K} := \int_\Omega \boldsymbol{v}(\boldsymbol{x}) \cdot \boldsymbol{v}(\boldsymbol{x})\, d\boldsymbol{x} = \big\|\boldsymbol{v}\big\|^2_{\mathbf{L}^2(\Omega)},$$

where $\boldsymbol{v}$ is measured in units of v_w and length is measured in units of h. Therefore $0 < \mathcal{K} < 1$ for any value of Re. Alternatively, the kinetic energy can be measured in units of $\frac{1}{2}\mu v_w h$, in this case

$$\widehat{\mathcal{K}} = Re\big\|\boldsymbol{v}\big\|^2_{\mathbf{L}^2(\Omega)}\,.$$

Figure 5.2 shows the kinetic energy as a function of the Reynolds number. For angles of tilt $\theta = 0$ and $\theta = 10$ the kinetic energy increases monotonically with Re

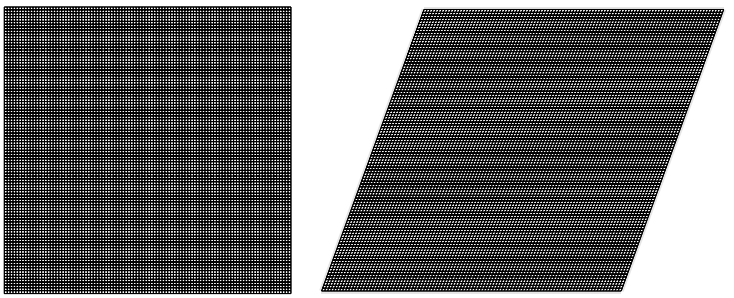


FIG. 5.1. *Finite element partition with* 10000 *finite elements and* 110,802 *equations.*

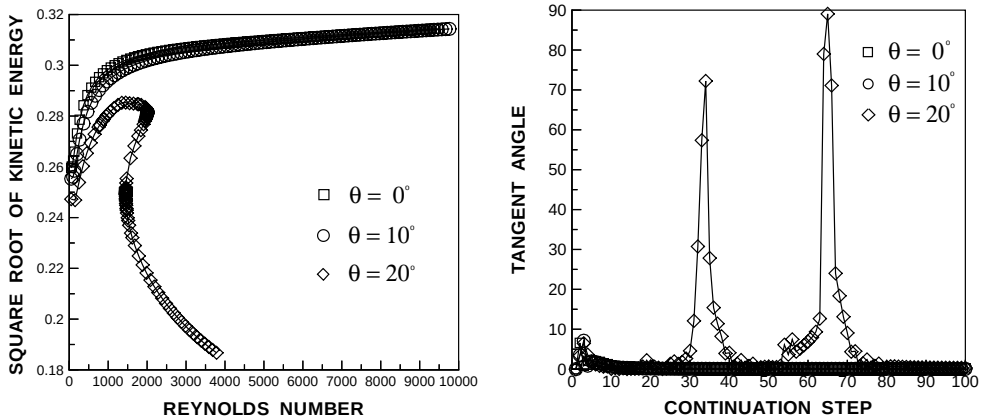
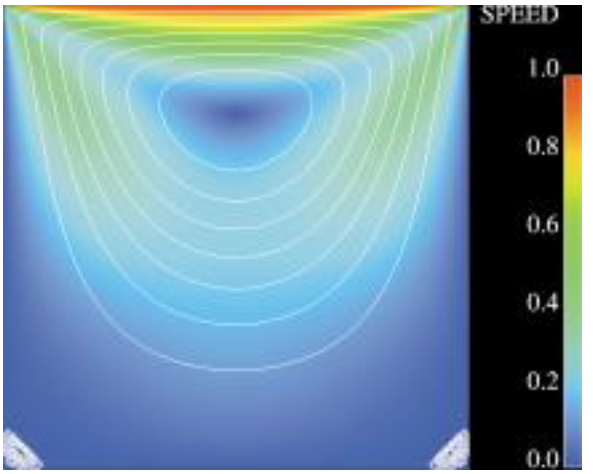


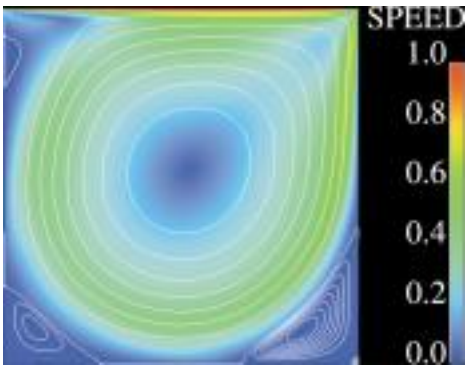
FIG. 5.2. *Solution curve with* 100 *points and* 110,802 *equations.* (a) *Square root of kinetic energy* $\mathcal{K}$ *of the LDC system for different geometries of the cavity.* (b) *Angle* α *between successive tangent vectors on the solution curves corresponding to the geometries in* (a).

and appears to approach an asymptotic limit. This behavior is a consequence of the evolution of the primary vortex in the cavity with an increasing Reynolds number. As Figure 5.3 shows, the primary vortex grows and fills a substantial portion of the cavity, therefore the system attains a greater kinetic energy level. The growth in size and strength is initially fast, with respect to the Reynolds number, for $0 < Re < 2000$ and moderate to slow for $Re > 2000$. The secondary vortices, present at the lower corners and upper left corner do not show a significant tendency to grow; therefore they do not contend for space in the cavity. In the absence of competition, the primary vortex expands and occupies the area of the circle circumscribed in the cavity.

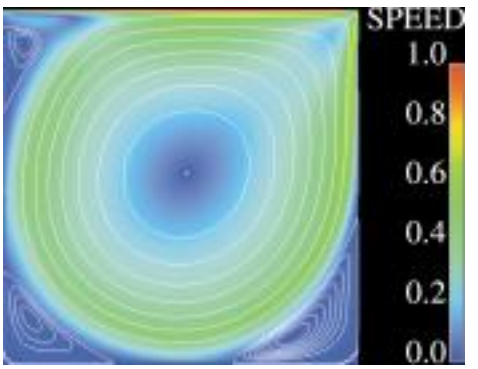
The variation of the kinetic energy with the Reynolds number for the angle of tilt $\theta = 20$ is qualitatively and quantitatively different owing to the presence of turning points. Multiple values of $\mathcal{K}$ in the range $1449.7 < Re < 2002.8$ result from the the existence of distinct configurations of vortices in the cavity with same level of kinetic energy. Figure 5.4 shows three different flow configurations at the same value of Re. The kinetic energy of the system lowers significantly for $Re > 2000$ in virtue of the expansion of secondary vortices carrying stagnant flow. In addition, $\mathcal{K}$ decreases with increasing Reynolds number in the range $1550 \leq Re \leq 2000$ on the upper branch,



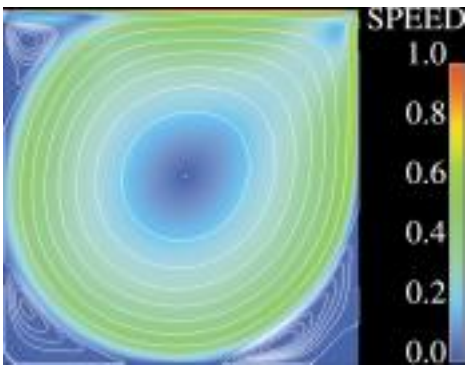
$Re = 0$



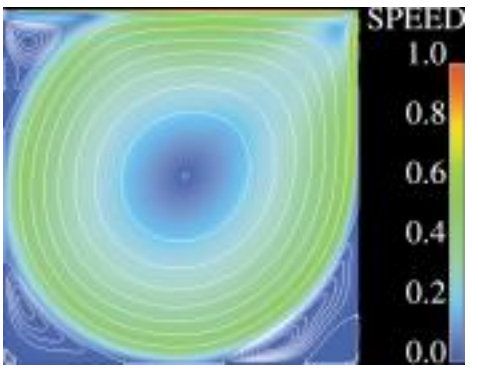
$Re = 2066.6$



$Re = 4066.6$



$Re = 8066.6$

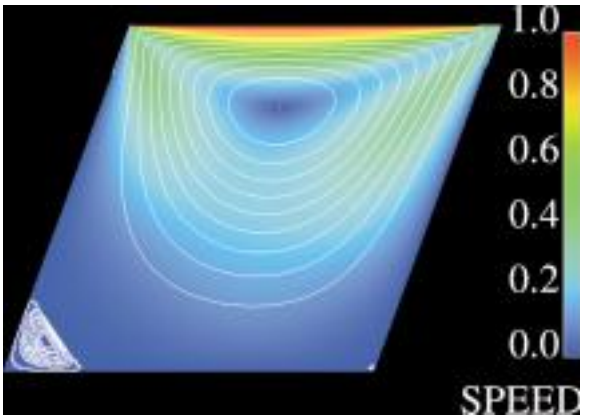


$Re = 9466.6$

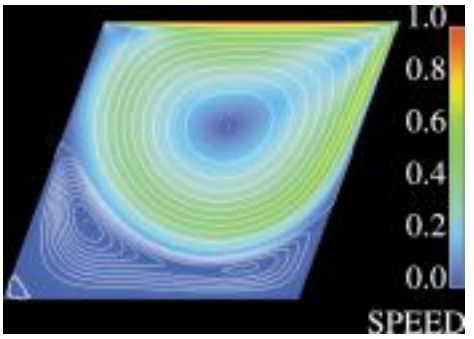
FIG. 5.3. *Flow speed and stream function contours for tilt angle* $\theta = 0$ *with finite element partition corresponding to* $110,802$ *equations.*

indicating that a change of sign on $\frac{d\mathcal{K}}{dRe}$ occurs prior to the first turning point.

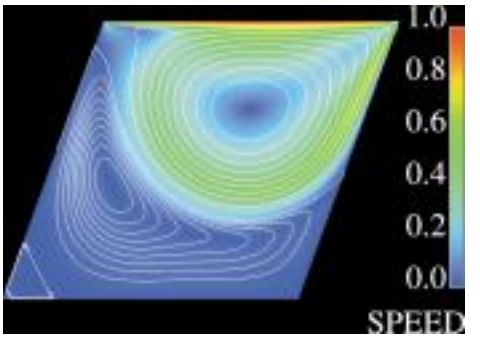
Qualitative changes in the flow with increasing Reynolds number are reflected in the performance of the predictor-corrector method (Figures 5.5 and 5.6). When the flow is rapidly changing with the Reynolds number, factorizations of $\mathbf{J}$ are needed more often but still more than one solution point can be obtained with a single



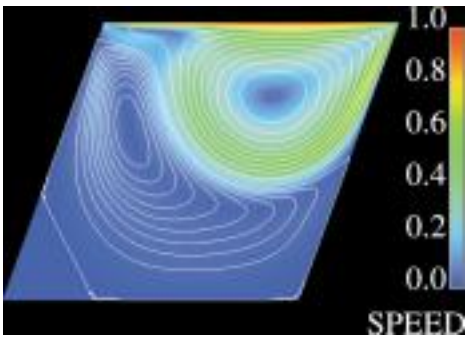
$Re = 0$



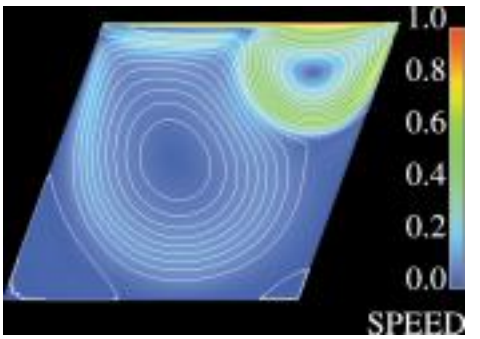
$Re = 1550$



$Re = 1550$



$Re = 1550$



$Re = 3784$

FIG. 5.4. *Flow speed and stream function contours for tilt angle* $\theta = 20$ *with finite element partition corresponding to* $110,802$ *equations. Turning points occur at* $Re = 1449.7$ *and* $Re = 2002.8$.

factorization assisted by a few inner iterations that rarely approach 50% of the cost of a factorization. For $\theta = 0$ the average number of converged solutions per factorization is 5.8, and for $\theta = 10$ it is 7.1. This shows that exploitation of a factorization of $\mathbf{J}$ along the solution curve is cost effective. Evidently, whenever a factorization takes place a high-order convergence results.

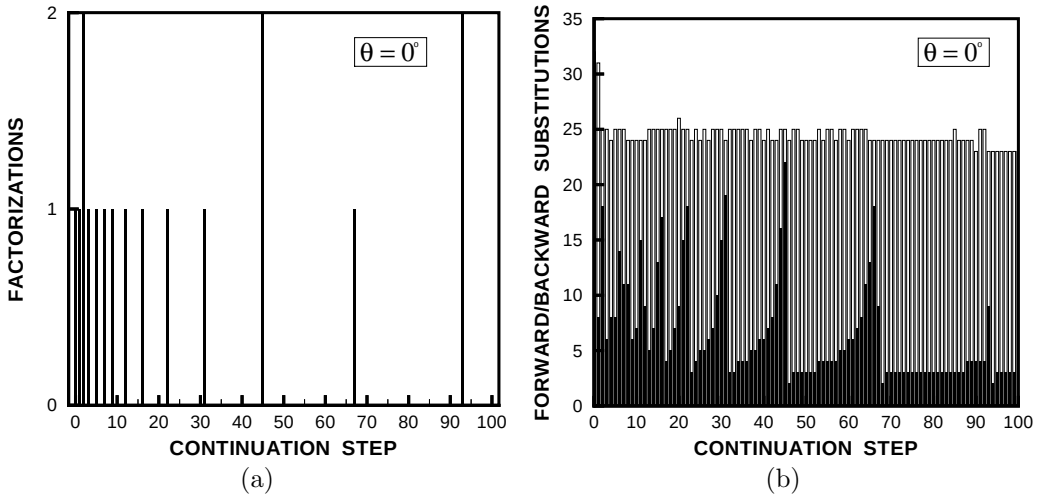


FIG. 5.5. *Performance of predictor-corrector method with* 110,802 *equations.* (a) *Factorizations of* **J** *per continuation step for tilt angle* $\theta = 0$. (b) *Corresponding forward/backward substitutions per continuation step represented with filled bars. Maximum allowed forward/backward substitutions represented with outlined bars.*

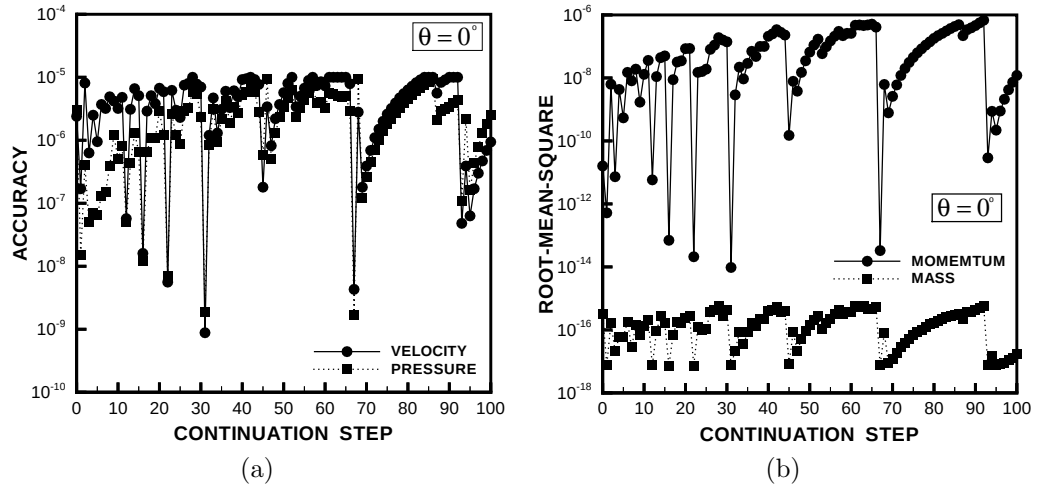


FIG. 5.6. *Performance of predictor-corrector method with* 110,802 *equations.* (a) *Relative norms of velocity and pressure correction fields, respectively,* (3.15a) *and* (3.15b)*, per continuation step for tilt angle* $\theta = 0$. (b) *Corresponding root-mean-square of momentum and mass residuals.*

The most demanding case is when the flow change with Re is more dramatic, particularly in the vicinity of turning points (Figures 5.7 and 5.8). Here the corrector step automatically employs the more expensive pseudoarclength parameter continuation as long as $\alpha < 10^\circ$ (Figure 5.2b). Fortunately $\alpha > 10$ for only a few solution points. In addition, corrector failures are likely to occur. The additional expenses, mentioned in section 4.3, are reflected on extra factorizations and forward/backward substitutions that may exceed 50% of the cost of a factorization (Figure 5.7). Nevertheless the method is still attractive because the average number of converged solutions per factorization is 1.3. This is especially surprising considering the fact that the stepsize

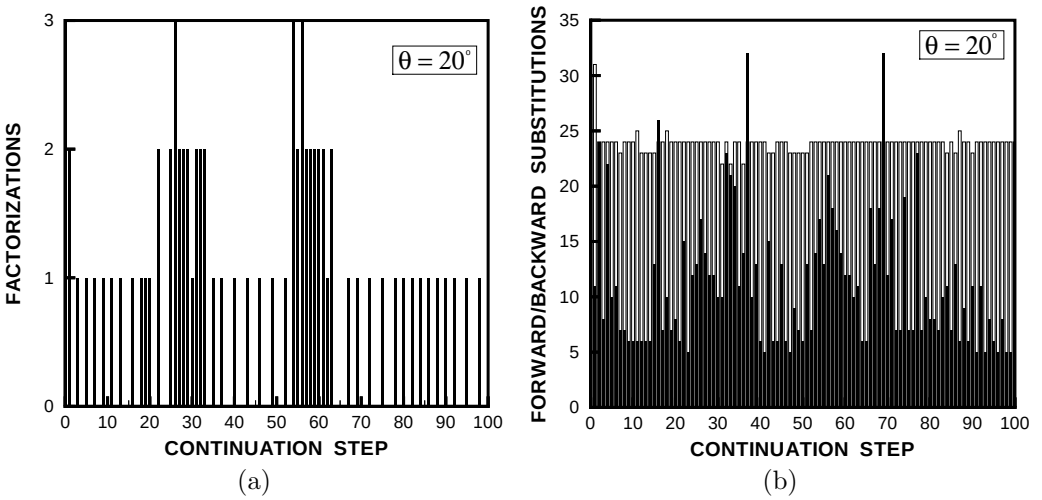


FIG. 5.7. *Performance of predictor corrector method with* 110,802 *equations.* (a) *Factorizations of* **J** *per continuation step for tilt angle* $\theta = 20$. (b) *Corresponding forward/backward substitutions per continuation step represented with filled bars. Maximum allowed forward/backward substitutions represented with outlined bars.*

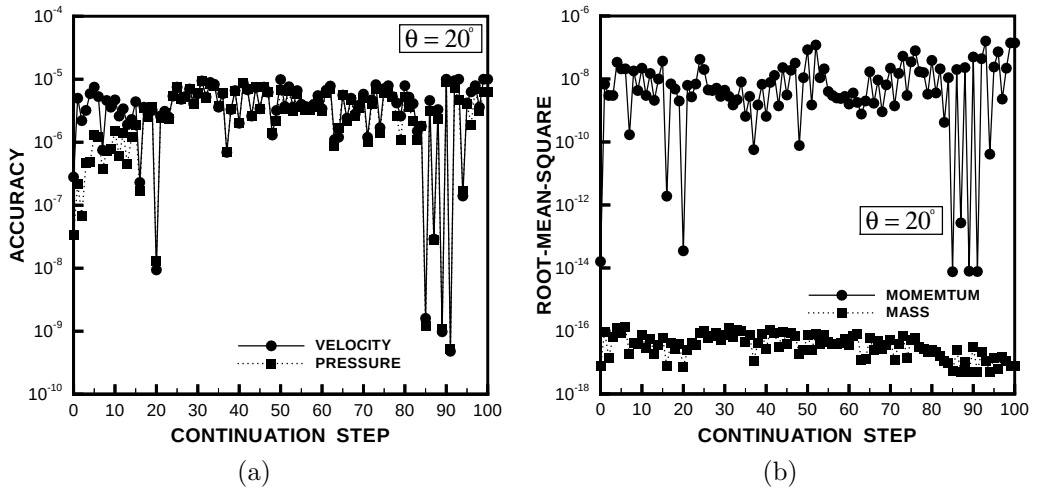


FIG. 5.8. *Performance of predictor corrector method with* 110,802 *equations.* (a) *Relative norms of velocity and pressure correction fields, respectively* (3.15a) *and* (3.15b)*, per continuation step for tilt angle* $\theta = 20$. (b) *Corresponding root-mean-square of momentum and mass residuals.*

on Re reduces four orders of magnitude before the continuation passes through the turning points.

6. Conclusions and final remarks. A cost-effective procedure for applying parametric continuation to the steady-state analysis of very large-scale process models which contain the Navier–Stokes equations to describe laminar, incompressible fluid flow was presented. The efficiency of this scheme relies on the judicious use of matrix factorizations and appropriate simplifications to the calculation of solution tangent vectors. The cost of the method, in terms of both arithmetic operations count and

data storage, is dominated by the solution of large linear systems of equations with sparse matrices of coefficients. When constructing a solution curve of the Navier–Stokes system, a factorization can be made cost effective, that is, several or many solutions can be accurately computed by recycling an existing factorization along the solution path.

The cost of constructing a solution curve can be further reduced by alternating automatically between the less expensive natural parametrization of the solution curve and the pseudoarclength parametrization. The angle between successive tangents to the solution curve can be used effectively, as an alternative to the curvature, to control the transition between parametrizations. To lessen the cost of computing the tangent angle, the tangent to the solution curve can be approximated by the secant joining two previously computed solution points; as long as the points are not in the neighborhood of a singular point—only *simple* singular points are considered here.

The proposed method was tested on a series of two-dimensional prototype problems of various sizes. In light of the results obtained, we estimate that with state-of-the-art vector supercomputers, analysis of two-dimensional models of materials and chemical processing will be feasible for nonlinear systems of up to 1,000,000 equations. This statement must be qualified by two potential limitations. For the test cases undertaken here, the limiting factor is data *storage* rather than arithmetic operation count. However, for the largest system tested, the CPU time (operation count) borders on the limit of what may be considered acceptable—the factorization of the system with 992,402 equations required approximately one hour of CPU time and the construction of the steady-state solution curve from $Re = 0$ to 10,000 and tilt angle $\theta = 0^\circ$ required a total of 34 CPU hours. The results are encouraging, since our experience indicates that discretizations of this magnitude (less than 1,000,000 equations) are more than adequate for producing numerically convergent solutions to realistic two-dimensional models. However, this satisfaction with two-dimensional model performance will certainly not hold for the analysis of 3-D counterparts. We outline our approach to such problems below.

Today's parallel supercomputers promise a substantial increase in CPU memory and computational speed. Our work [de Almeida and Derby, 1999] toward utilizing parallel computers' resources to solve 3-D models is twofold. First, the method of bifurcation analysis proposed here can be readily implemented in parallel by employing a substructuring subdomain decomposition algorithm [Chan and Mathew, 1994] and factoring the resultant subdomain matrices in parallel. The equivalent linear algebra description of this approach is the method of multiple fronts [Duff and Scott, 1994]. The difficulty with this approach is how to exploit parallelism to factor the associated Schur complement matrix. A feasible alternative is to factor the Schur matrix in parallel via nearest neighbor comunication. Again, data storage requirement is the bottleneck of a direct method of solution of the linear system.

Second, there exists a class of models of materials processing systems involving several field quantities (multiphysics applications) for which even the proposed method of analysis will likely be inadequate. For such models, a promising alternative is to resort to an iterative method of solution of linear systems of equations as a preconditioner for an existing direct method—so-called nonconventional preconditioning [Duff, 1996]. Our work towards both of these ends is reported in [de Almeida and Derby, 1999].

Appendix. Functional spaces and norms. Definitions of the standard functional spaces and norms used heretofore.

$$\mathrm{L}^2(\Omega) := \left\{ \pi \colon \Omega \to \mathbb{R} \;\mid\; \|\pi\|^2_{\mathrm{L}^2(\Omega)} < +\infty \right\},$$
$$\|\pi\|_{\mathrm{L}^2(\Omega)} := \left(\int_\Omega \pi(\boldsymbol{x})^2 \, d\boldsymbol{x} \right)^{\frac{1}{2}},$$

$$\mathrm{H}^1(\Omega) := \left\{ \psi \colon \Omega \to \mathbb{R} \;\mid\; \|\psi\|^2_{\mathrm{H}^1(\Omega)} < +\infty \right\},$$
$$\|\psi\|_{\mathrm{H}^1(\Omega)} := \left(\int_\Omega \psi(\boldsymbol{x})^2 + \nabla_{\boldsymbol{x}} \psi \cdot \nabla_{\boldsymbol{x}} \psi \, d\boldsymbol{x} \right)^{\frac{1}{2}},$$

$$\mathbf{H}^1(\Omega) := \left\{ \boldsymbol{\psi} \colon \Omega \to \mathbb{R}^2 \;\mid\; \boldsymbol{\psi}(\boldsymbol{x}) = \sum_i \psi_i(\boldsymbol{x}) \, \boldsymbol{i}_i \,, \quad \psi_i \in \mathrm{H}^1(\Omega) \text{ for } i = 1, 2 \right\},$$
$$\mathbf{H}^1_0(\Omega) := \left\{ \boldsymbol{\psi} \colon \Omega \to \mathbb{R}^2 \;\mid\; \boldsymbol{\psi} \in \mathbf{H}^1(\Omega) \text{ and } \boldsymbol{\psi} \equiv \mathbf{0} \text{ on } \partial\Omega \right\},$$
$$\|\boldsymbol{\psi}\|_{\mathbf{H}^1(\Omega)} := \left(\int_\Omega \boldsymbol{\psi}(\boldsymbol{x}) \cdot \boldsymbol{\psi}(\boldsymbol{x}) + \nabla_{\boldsymbol{x}} \boldsymbol{\psi} \bullet \nabla_{\boldsymbol{x}} \boldsymbol{\psi} \, d\boldsymbol{x} \right)^{\frac{1}{2}},$$
$$\|\boldsymbol{\psi}\|_{\mathbf{L}^2(\Omega)} := \left(\int_\Omega \boldsymbol{\psi}(\boldsymbol{x}) \cdot \boldsymbol{\psi}(\boldsymbol{x}) \, d\boldsymbol{x} \right)^{\frac{1}{2}}.$$

Acknowledgment. The content of this paper does not necessarily reflect the position or policy of the government, and no official endorsement should be inferred. VFdA gratefully acknowledges support from the research scholars program of MSI.

REFERENCES

E. L. Allgower and K. Georg (1990), *Numerical Continuation Methods*, Springer Ser. Comput. Math. 15, Springer-Verlag, New York.

V. F. de Almeida and J. J. Derby (1997), *Construction of Solution Curves for Large 2-D Problems of Steady-State Flows of Incompressible Fluids*, Minnesota Supercomputer Institute Research Report 97/070, University of Minnesota, Department of Chemical Engineering and Materials Science, Minneapolis, MN 55455-0132, 1997; also available at (de_almeida@na-net.ornl.gov) by request or at http://www.msi.umn.edu/~dalmeida.

V. F. de Almeida and J. J. Derby (1999), *Preferred Method for Setting a Pressure Level in Computations of 3-D Flows of Incompressible Fluids with GFEM*, Minnesota Supercomputer Institute Research Report 99/005, University of Minnesota, Department of Chemical Engineering and Materials Science, Minneapolis, MN; also available at (de_almeida@na-net.ornl.gov) by request and at http://www.msi.umn.edu/~dalmeida.

R. E. Bank and H. D. Mittelmann (1989), *Stepsize selection in continuation procedures and damped Newton's method*, J. Comput. Appl. Math., 26, pp. 67–77.

R. Brent (1973), *Some efficient algorithms for solving systems of nonlinear equations*, SIAM J. Numer. Anal., 10, pp. 327–44.

F. Brezzi and M. Fortin (1991), *Mixed and Hybrid Finite Element Methods*, Springer Ser. Comput. Math. 15, Springer-Verlag, New York.

T. F. Chan and T. P. Mathew (1994), *Domain decomposition algorithms*, Acta Numerica, Cambridge University Press, Cambridge, UK, pp. 61–143.

J. J. Derby, S. Brandon, A. G. Salinger, and Q. Xiao (1994), *Large-scale numerical analysis of materials processing systems: High-temperature crystal growth and molten glass flows*, Comput. Methods Appl. Mech. Engrg., 112, pp. 69–89.

P. Deuflhard, B. Fiedler, and P. Kunkel (1987), *Efficient numerical pathfollowing beyond critical points*, SIAM J. Numer. Anal., 24, pp. 912–927.

J. J. Dongarra, J. Du Croz, I. S. Duff, and S. Hammarling (1990), *A set of level* 3 *basic linear algebra subprograms*, ACM Trans. Math. Software, 16, pp. 1–17.

I. S. Duff (1996), *Sparse Numerical Algebra: Direct Methods and Preconditioning*, Research Report RAL-TR 96-047, Department for Computation and Information, Atlas Centre, Rutherford Appleton Laboratory, Oxon, U.K..

I. S. Duff and J. A. Scott (1993), *MA*42*—A New Frontal Code for Solving Sparse Unsymmetric Systems*, Research Report RAL 93-064, Central Computing Department, Atlas Centre, Rutherford Appleton Laboratory, Oxon, U.K.

I. S. Duff and J. A. Scott (1994), *The Use of Multiple Fronts in Gaussian Elimination*, Research Report RAL 94-040, Central Computing Department, Rutherford Appleton Laboratory, Oxon, U.K.

I. S. Duff, J. K. Reid, and J. A. Scott (1989), *The use of profile algorithms with a frontal code*, Internat. J. Numer. Methods Engrg., 28, pp. 2555–2568.

A. Fortin, M. Fortin, and J. Gervais (1987), *A numerical simulation of the transition to turbulence in a two-dimensional flow*, J. Comp. Phys., 70, pp. 295–310.

M. Fortin and A. Fortin (1985), *A generalization of Uzawa's algorithm for the solution of the Navier-Stokes equations*, Commun. Appl. Numer. Methods, 1, pp. 205–208.

M. A. Gomes-Ruggiero, J. M. Martínez, and A. C. Moretti (1992), *Comparing algorithms for solving sparse nonlinear systems of equations*, SIAM J. Sci. Stat. Comput., 13, pp. 459–483.

M. D. Gunzburger and J. S. Peterson (1991), *Predictor and steplength selection in continuation methods for the Navier-Stokes equations*, Comput. Math. Appl., 22, pp. 73–81.

C. D. Heijer and W. C. Rheinboldt (1981), *On steplength algorithms for a class of continuation methods*, SIAM J. Numer. Anal., 18, pp. 925–948.

G. Karypis (1996), *Graph Partitioning and Its Applications to Scientific Computing*, Ph.D. dissertation, University of Minnesota, Department of Computer Science.

G. Karypis and V. Kumar (1995), *A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs*, Research Report 95-035, University of Minnesota, Department of Computer Science, Minneapolis, MN; also available at http://www.cs.umn.edu/~karypis.

H. B. Keller (1992), *Numerical Methods for Two-Point Boundary-Value Problems*, Dover Books in Advanced Mathematics, Dover Publications, Inc., New York. Corrected and expanded edition of the work first published by Blaisdell Publishing Company, New York, 1964.

W. C. Rheinboldt (1980), *Solution fields of nonlinear equations and continuation methods*, SIAM J. Numer. Anal., 17, pp. 221–237.

A. G. Salinger, Q. Xiao, Y. Zhou, and J. J. Derby (1994), *Massively parallel finite element computations of three-dimensional, time-dependent, incompressible flows in materials processing systems*, Comput. Methods Appl. Mech. Engrg., 119, pp. 139–156.

H. Schwetlick and J. Cleve (1987), *Higher order predictors and adaptive steplength control in path following algorithms*, SIAM J. Numer. Anal., 24, pp. 1382–1393.

V. E. Shamanskii (1967), *A modification of Newton's method*, Ukran. Mat. Zh., 19, pp. 133–138 (in Russian).

S. W. Sloan (1989), *A fortran program for profile and wavefront reduction*, Internat. J. Numer. Methods Engrg., 28, pp. 2651–2679.

SIAM J. SCI. COMPUT.
Vol. 22, No. 1, pp. 312–329
© 2000 Society for Industrial and Applied Mathematics

AN ITERATIVE ALGORITHM FOR SOLVING HAMILTON–JACOBI TYPE EQUATIONS*

JERRY MARKMAN† AND I. NORMAN KATZ†

Abstract. Solutions of the optimal control and H_∞-control problems for nonlinear affine systems can be found by solving Hamilton–Jacobi equations. However, these first-order nonlinear partial differential equations can, in general, not be solved analytically. This paper introduces an iterative algorithm which solves these equations numerically for points near the origin. The procedure converges to the stabilizing solution exponentially with respect to the iteration variable. The algorithm is implemented on both illustrative and comparative examples.

Key words. Hamilton–Jacobi equations, iterative algorithms

AMS subject classifications. 65M12, 93C10, 49N35

PII. S1064827598344315

1. Introduction.

We consider partial differential equations of the type

$$p^T f(x) \pm p^T R(x) p + l(x) = 0, \tag{1.1}$$

where $x \in \mathbb{R}^n, p = V_x(x)$, and $R(x), l(x) \geq 0$. Equation (1.1) is a Hamilton–Jacobi type equation which arises in nonlinear theory. For example, consider the optimal control problem

$$J(x_0, u) = \int_0^\infty l(x(t, x_0)) + \|u(x(t, x_0))\|^2 dt, \tag{1.2}$$

where $l : \mathbb{R}^n \to \mathbb{R}$ is a positive definite, monotonically increasing function, $u \in \mathbb{R}^m$, and x satisfies the differential equation

$$\begin{aligned} \dot{x} &= f(x) + g(x)u, \\ x(0) &= x_0. \end{aligned} \tag{1.3}$$

It is well known [1] that the optimal control can be directly found to be $u^* = -\frac{1}{2} g^T \frac{\partial V}{\partial x}$, where V satisfies the Hamilton–Jacobi–Bellman equation

$$\frac{\partial V^T}{\partial x} f(x) - \frac{1}{4} \frac{\partial V^T}{\partial x} g(x) \left(\frac{\partial V^T}{\partial x} g(x) \right)^T + l(x) = 0. \tag{1.4}$$

*Received by the editors September 8, 1998; accepted for publication (in revised form) Janauary 6, 2000; published electronically June 22, 2000. This work was supported in part by the National Science Foundation under grant DMS-9626202 and in part by the Defense Advanced Research Projects Agency (DARPA) and Air Force Research Laboratory, Air Force Materiel Command, USAF, under agreement F30602-99-2-0551. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright annotation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of DARPA, the Air Force Research Laboratory, or the U.S. Government.

http://www.siam.org/journals/sisc/22-1/34431.html

†Department of Systems Science and Mathematics, Washington University, Campus Box 1040, St. Louis, MO 63130 (jerry@zach.wustl.edu, katz@zach.wustl.edu). The results here are part of the doctoral dissertation of the first author.

Equations similar to (1.4) are derived when solving various H_∞ problems. In this case, in addition to (1.3), an output equation for $y \in \mathbb{R}^p$ is given by

$$y = h(x). \tag{1.5}$$

The system (1.3) is said to have L_2 gain $\leq \gamma$ if

$$\|y(x)\| \leq \gamma_0 \|u(x)\| \tag{1.6}$$

for all $u \in L_2[0, \infty]$, for all $t > 0$, for some $\gamma_0, 0 \leq \gamma_0 \leq \gamma$. If $V_x(x) = p$ satisfies the Hamilton–Jacobi–Isaacs equation

$$p^T f(x) + \frac{1}{4\gamma^2} p^T g(x) g^T(x) p + h^T(x) h(x) = 0, \tag{1.7}$$

then the system (1.3) has L_2-gain γ for all $u(x)$ satisfying $\|u(x)\| \leq \delta$, for sufficiently small $\delta > 0$. See [2], [3] for details. In this case, equality occurs if $u^*(x) = \frac{1}{2\gamma^2} g^T(x) p$.

Because (1.1) is a first-order nonlinear equation, closed form solutions cannot be found in general. Furthermore, the only information known about $V(x)$ is that it is a C^2 positive definite solution locally in a neighborhood around the equilibrium point $x = 0$ and $V(0) = 0$. The size of this neighborhood is not known in advance, and no boundary conditions are known, so traditional numerical methods are inapplicable. Approximate solutions of (1.1) have been found either through a power series method [4], [5] or a successive approximation approach [6], [7], [8]. In this paper, an iterative procedure is introduced which computes the stabilizing solution of (1.1) for any x_0 near the origin. Convergence is shown to be exponential, allowing the user to find a solution of any desired accuracy. By applying the algorithm to sufficiently many points near the origin, an approximate solution over the entire neighborhood can be pieced together through, for example, polynomial interpolation.

The paper is organized as follows. In section 2, the motivation and the algorithm are given. Section 3 discusses the numerical issues involved in the implementation of the procedure. Section 4 states the results pertaining to existence and rate of convergence, and gives an example highlighting these results. In section 5, two numerical examples are given. Section 6 illustrates the convergence rate graphically. Another simulation is presented comparing the algorithm to different methods in the literature. Finally it is shown how the computation along an initial manifold can be used to calculate an approximate solution in an entire neighborhood of the origin, by applying the method of characteristics.

2. Algorithm.

2.1. Motivation. First, the problem is formally defined. For the dynamic system

$$\begin{aligned} \dot{x} &= f(x) + g(x)u, \\ y &= h(x), \end{aligned} \tag{2.1}$$

with $x \in \mathbb{R}^n, u \in \mathbb{R}^m, h(x) \in \mathbb{R}^p$, where $f(x)$, $g(x)$, $h(x)$ are smooth and $f(0) = h(0) = 0$, a positive semidefinite function $V^-(x) : U_\rho \to \mathbb{R}$ is sought in an open neighborhood U_ρ around the origin which satisfies the Hamilton–Jacobi equation

$$H^*(x, V_x) \equiv V_x^T f(x) + \frac{1}{k} V_x^T g(x) g^T(x) V_x + h^T(x) h(x) = 0 \tag{2.2}$$

and renders (2.1) asymptotically stable with $u = \frac{2}{k} g^T(x) V_x^-(x)$. Here, k is a parameter which depends on the type of problem being considered. For instance, in the H_∞ problem, $k = 4\gamma^2$, while for the optimal control problem $k = -4$.

The algorithm operates in the Hamiltonian space of (2.2), i.e., it considers the $2n$-dimensional ODE

$$\begin{aligned} \frac{dx}{dt} &= H_p^*, \\ \frac{dp}{dt} &= -H_x^*. \end{aligned} \tag{2.3}$$

It can be shown that any solution of the Isaacs equation $H^*(x, V_x) = 0$ corresponds to an invariant manifold of the Hamiltonian system [2]. Furthermore, under suitable conditions for the optimal control and H_∞ problems, the Hamiltonian system is restricted to the hyperbolic case, i.e., the linearization of (2.3) has no eigenvalues on the imaginary axis. This insures the existence of an n-dimensional stable manifold. Finally, for x near the origin, the stable manifold can be expressed as a graph of x. Therefore, seeking the stabilizing solution of (2.2) at a fixed point near the origin is equivalent to identifying the stable invariant manifold of (2.3) at that point.

Define

$$\begin{aligned} z(t, z_0) &\triangleq \begin{pmatrix} x(t, x_0, p_0) \\ p(t, x_0, p_0) \end{pmatrix}, \\ z_0 &\triangleq \begin{pmatrix} x_0 \\ p_0 \end{pmatrix}, \end{aligned} \tag{2.4}$$

where $x(t, x_0, p_0)$ and $p(t, x_0, p_0)$ are the solutions of (2.3) at time t with initial conditions x_0 and p_0. Also define $\bar{H}$ as the linearization of (2.3) around the equilibrium point $(x, p)^T = (0, 0)^T$. Then (2.3) can be rewritten as

$$\begin{aligned} \dot{z} &= \bar{H} z + h(z), \quad h \in O(\|z\|^2), \\ \bar{H} &= \frac{\partial}{\partial z} \begin{pmatrix} H_p^* \\ -H_x^* \end{pmatrix}_{z=0}. \end{aligned} \tag{2.5}$$

Fix a point $x = x_0$, $x_0 \in U_\rho$. Finding the stabilizing solution of (2.2) at x_0, defined as $p_0^* \triangleq V_x^-(x_0)$, is equivalent to finding the vector p_0 which causes $z(t, z_0) \to 0$ as $t \to \infty$.

The algorithm uses this asymptotic behavior of the stabilizing solution of (2.2) to find p_0^*. For a fixed time t, it evaluates each possible solution vector p by measuring how close the trajectory of the Hamiltonian system (2.3) passing through (x_0, p) comes to the origin. This is repeated for larger and larger times to produce a sequence of vectors which converge to a point on the stable manifold.

Define the distance function

$$F(t, p_0) \triangleq \|z(t, z_0)\|^2 = \|x(t, x_0, p_0)\|^2 + \|p(t, x_0, p_0)\|^2. \tag{2.6}$$

F measures the distance from the solution of (2.5), $z(t, z_0)$ to the origin at time t with initial condition z_0. Because x_0 is fixed, F depends solely on the terminal time t and the initial condition p_0.

Since the solution of (2.2), $p_0^* = V_x^-(x_0)$, lies on the stable manifold of (2.5), $\lim_{t\to\infty} F(t, x_0, p_0^*) \to 0$. Because F is a positive semidefinite function, finding p_0^*, or

where the stable manifold intersects $x = x_0$, is equivalent to solving

$$\inf_{p_0} \lim_{t\to\infty} F(t, x_0, p_0). \tag{2.7}$$

However, because the Hamiltonian system (2.5) is nonlinear, a closed form solution for $z(t, z_0)$ is generally not available, and the minimization cannot be done analytically. To approximate a solution numerically, a sequence of times which monotonically increases towards infinity is chosen, and for each element in this sequence F is minimized with respect to p_0. For each $i = 1, 2, 3, \ldots$ define

$$t^i \uparrow \infty, \tag{2.8a}$$

$$F^i \triangleq \min_{p_0} F(t^i, p_0), \tag{2.8b}$$

$$p_0^i \triangleq \operatorname*{argmin}_{p_0} F(t^i, p_0). \tag{2.8c}$$

As i becomes larger, $t^i \to \infty$, and F can be considered, in a sense, as measuring the asymptotic behavior of candidate solutions. It will be shown in section 4 that $F^i \to 0$ and $p_0^i \to p_0^*$.

2.2. Initial guess. In order to solve (2.8b) and (2.8c) for fixed i, an iterative procedure is, in general, required. For each i, a starting estimate is taken to be the converged value p_0^{i-1}. Since the algorithm is an iterative procedure, a starting guess for p_0^* is needed. A good estimate for p_0^* which is taken to be p_0^0, can be found by considering the linearization of (2.1). For any linear system, the solution of the corresponding Hamilton–Jacobi equation is $V^L(x) = x^T P x$, where P is the stabilizing solution of the algebraic Ricatti equation

$$PA + A^T P + \frac{1}{k} PBB^T P + C^T C = 0, \tag{2.9}$$

where $A = \frac{\partial f}{\partial x}(0)$, $B = g(0)$, $C = \frac{\partial h}{\partial x}(0)$. It is assumed that $\sigma(A) \subset C^-$.

Thus, the initial guess for the algorithm is $V_x^L(x_0) = p_0^L = 2x_0^T P$.

This has a nice geometric interpretation because p_0^L is the intersection of the stable eigenspace of $\bar{H}$ with the linear variety $x = x_0$. Because x_0 is always chosen near the origin, p_0^L will lie close to p_0^*.

3. Implementation.

3.1. Minimizing the distance function over the costate variable. For each element in a sequence t^i, the algorithm solves the problem

$$p_0^i = \operatorname*{argmin}_{p_0} \quad F(t^i, p_0). \tag{3.1}$$

The minimization is done iteratively via a Gauss–Newton method. For a fixed $t = t^i$, define j as the iteration variable for the Gauss–Newton method. Then, Newton's equation is

$$(p^{j+1})^T \triangleq (p^j)^T - \nabla F (\nabla^2 F)^{-1}, \tag{3.2}$$

where ∇F and $\nabla^2 F$ are the gradient and Hessian with respect to the covector p,

$$\nabla F = z^T(t, p^j) \frac{\partial z}{\partial p},$$

$$\nabla^2 F = \frac{\partial z}{\partial p}^T \frac{\partial z}{\partial p} + z^T(t, p^j) \frac{\partial^2 z}{\partial p^2}, \tag{3.3}$$

and the derivative terms $\frac{\partial z}{\partial p}$ and $\frac{\partial^2 z}{\partial p^2}$ are also evaluated at the point (t^i, p^j). Here, the second derivative $\frac{\partial^2 z}{\partial p^2}$ represents a bilinear operator on z. More specifically, $z^T(t,p^j)\frac{\partial^2 z}{\partial p^2} = \sum_{i=1}^{2n} z_i(t,p^j)\frac{\partial^2 z_i}{\partial p^2}$, where z_i is the ith component of z.

The idea behind the Gauss–Newton method is that for points close to the minimizing point, the second derivative term of the Hessian, $\frac{\partial^2 z}{\partial p^2}$, is dominated by the first-order term. Therefore, if a starting guess near the optimum can be found, the Hessian can be replaced by the approximation

$$\tilde{\nabla}^2 F \triangleq \frac{\partial z}{\partial p}^T \frac{\partial z}{\partial p}, \tag{3.4}$$

and the iteration equation (3.2) becomes

$$\begin{aligned}(p^{j+1})^T &= (p^j)^T - z^T \frac{\partial z}{\partial p}\left(\frac{\partial z}{\partial p}^T \frac{\partial z}{\partial p}\right)^{-1} \\ &= (p^j)^T - z^T(t,p^j)\frac{\partial z}{\partial p}^+(t,p^j),\end{aligned} \tag{3.5}$$

where $\frac{\partial z}{\partial p}^+$ is the generalized inverse.

Here, the optimum costate vector for the previous time in the sequence t^i is used for the initial guess, i.e., p^0 at time t^i is the optimal initial condition p^* for time t^{i-1}. If $\delta t \triangleq (t^i - t^{i-1})$ is small, the corresponding minimizing initial conditions will lie near each other as well, i.e., $\|p^i - p^{i-1}\|$ will also be small. Since the sequence of times is chosen by the user, a "good" starting guess can always be constructed.

Because the Hamiltonian system is nonlinear, the two terms in (3.5) need to be calculated numerically as well. The vector $z(t,p^j)$ is a solution of the Hamiltonian system (2.3) with initial condition (x_0, p^j), which can be integrated using standard Runge–Kutta methods. A greater difficulty lies in calculating $\frac{\partial z}{\partial p}$. Two approximation methods are presented.

In the first method, the variational equation is adjoined to the Hamiltonian system. Write the Hamiltonian system (2.2) in terms of its linearization,

$$\dot{z} = \bar{H}z + h(z), \tag{3.6}$$

with $h(z) \in O(\|z\|^2)$. Then the variational matrix $\frac{\partial z}{\partial p}$ satisfies the equation

$$\frac{d}{dt}\frac{\partial z}{\partial p} = \left(\bar{H} + \frac{\partial h}{\partial z}(z(t,p^j))\right)\frac{\partial z}{\partial p}, \tag{3.7}$$

$$\frac{\partial z}{\partial p}\Big|_{t=0} = \begin{pmatrix}0\\ I\end{pmatrix}. \tag{3.8}$$

The initial condition is derived by noting that $\frac{\partial z}{\partial p}$ is "half" of the full variational matrix $\frac{\partial z}{\partial z_0} = (\frac{\partial z}{\partial x_0} \vdots \frac{\partial z}{\partial p_0})$.

Equation (3.7) is linear but time-varying, so in general it cannot be solved directly. But it can be rearranged and adjoined to (3.6) to form a $2n(n+1)$-dimensional system, which is then integrated numerically.

The second method to calculate $\frac{\partial z}{\partial p}$ is through a simple finite difference approximation. Define $e_i \in \mathbb{R}^n$ as the unit vector at coordinate i and let $\delta > 0$ be a

perturbation factor. Then, a forward difference approximation to the derivative is

$$\frac{\partial z}{\partial p_i}(t, p^j) \approx \frac{z(t, p^j + \delta e_i) - z(t, p^j)}{\delta e_i}. \tag{3.9}$$

Note that the forward difference approximation requires integrating a $2n$-dimensional system $n+1$ times, therefore it has the same computational cost as the first method. If higher accuracy is desired, a central difference approximation can be employed. However, in that case, the $2n$-dimensional system will need to be integrated $2n$ times, resulting in a higher computational cost. δ must be chosen small enough to approximate the derivative but not too small to lose precision.

3.2. Time of closest approach. Choosing the sequence of times used in the algorithm will have a major influence on its performance. For example, suppose the sequence of times tend to infinity too quickly, say $t^{i+1} - t^i \geq K >> 0$. Then there may be numerical problems in finding the corresponding initial conditions, or the elements in the sequence of initial conditions will have such wide spacing between them that confidence in the solution may be reduced. On the other hand, choosing $t^{i+1} - t^i$ very small may put unnecessary computational burden on the algorithm to get a satisfactory answer.

A method for choosing the sequence of times is introduced in this section. It balances the issues raised above by taking an element in p_0^i and finding the time which brings the trajectory passing through it closest to the origin. This new time, called the time of closest approach (TOCA), is chosen as the next time, t^{i+1}, in the sequence. Because the solutions of a differential equation are continuous with respect to the time and the initial condition, it follows that p_0^{i+1} lies close to p_0^i. Furthermore, the TOCA "optimizes" each initial condition p_0^i, in the sense that it pulls the trajectory passing through it as close to the origin as possible.

DEFINITION 3.1. *Consider a system of differential equations $\dot{z} = f(z)$, $z(0) = z_0$, and assume a unique solution, $z(t, z_0)$, passing through z_0 exists for $t \in [0, \infty)$. Then TOCA, t^*, is defined as the time at which the trajectory comes closest to the origin for all time greater than or equal to zero, i.e.,*

$$\|z(t^*, z_0)\|^2 \leq \|z(t, z_0)\|^2 \quad \forall t \in [0, \infty). \tag{3.10}$$

The point of closest approach (POCA) is defined as the solution of the differential equation at t^* with initial condition z_0, i.e., $z^* \triangleq z(t^*, z_0)$.

Remark 3.1. For a fixed initial condition z_0, the TOCA can be calculated by minimizing the distance function F over all positive times, i.e.,

$$t^* \triangleq \underset{t \geq 0}{\operatorname{argmin}}\ F(t, z_0). \tag{3.11}$$

The TOCA is well defined as long as a solution to the differential equation exists. Note that if z_0 lies on an asymptotically stable manifold of the system, then the POCA is zero and the TOCA is defined as ∞. Intuitively, this indicates that the sequence of times using TOCA will diverge, satisfying the conditions of the algorithm.

The TOCA can also be thought of as the time where a trajectory "turns away" from the origin. Since the algorithm seeks the trajectory which approaches the origin, the TOCA allows the algorithm to "eliminate" certain initial conditions by measuring when their trajectories stop approaching the equilibrium point.

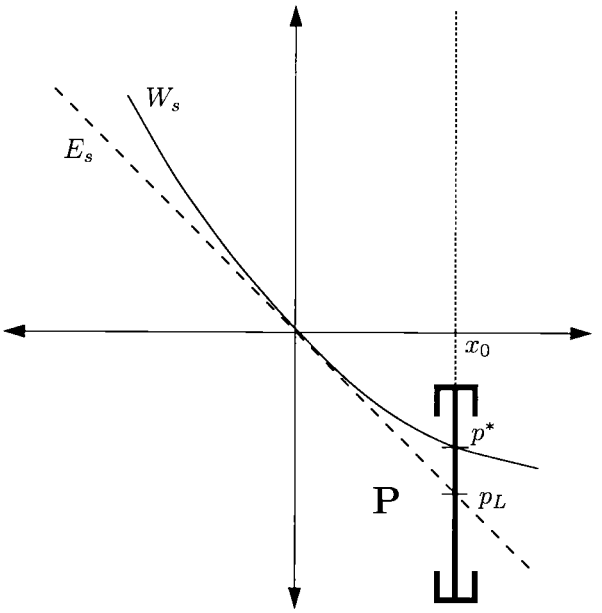


FIG. 1. *Restricting the search space of* p_0^*.

This minimization is also performed using a Gauss–Newton method, and analogously the iteration equation is

$$t^{j+1} = t^j - z^T(t^j, p_0)\frac{\partial z}{\partial t}^+(t^j, p_0). \tag{3.12}$$

As above, the initial guess for the iteration is the optimal time for the previous initial condition, i.e., $t^{j_0} = t^{i-1}$. The "interlocking" nature of the TOCA with the sequence p_0^i makes this initial guess close to the optimal in practice.

The terms in (3.12) can be found directly. $z^T(t^j, p_0)$ can be determined by integrating system (2.2). And note that $\frac{\partial z}{\partial t}(t^j, p_0) = \frac{dz}{dt}(z(t^j, x_0, p_0))$, which can be found exactly from the right-hand side of the Hamiltonian system (2.2). Therefore, for each iteration only $2n$ integrations are required.

4. Convergence results. p_0^L is the intersection of the stable eigenspace of H with the linear variety $x = x_0$. p_0^L can be found directly from the solution of the algebraic Ricatti equation (2.9) and is used as the initial guess in the iteration formulas defined by the algorithm. Because x_0 is chosen to be near the origin, at x_0 the stable manifold of (2.2) should be "close" to the stable eigenspace, i.e., p_0^* should lie near p_0^L. Therefore, instead of considering all vectors p as candidates to lie on the stable manifold, the algorithm considers the set $P \triangleq \{p : \|p - p_0^L\| \leq \delta\}$ for δ suitable large. See Figure 1.

Remark 4.1. Restricting the state space to the set P is not needed in the implementation of the algorithm, but is made here to insure the existence of accumulation points of the sequence p_0^i.

LEMMA 4.1. *$F^i \to 0$, as $t^i \to \infty$.*

Proof. p_0^* lies on the stable manifold of (2.2), so $\lim_{t^i\to\infty}\|z(t^i,x_0,p_0^*)\|\to 0$. For each t^i,

$$F^i=\|z(t^i,x_0,p_0^i)\|^2=\min_{p_0\in P}\|z(t^i,x_0,p_0)\|^2\le\|z(t^i,x_0,p_0^*)\|^2. \tag{4.1}$$

Therefore, as $t^i\to\infty$,

$$0\le\lim F^i\le\lim\|z(t^i,x_0,p_0^*)\|^2=0. \tag{4.2}$$

That is, $F^i\to 0$.

Because p_0^i lies in the closed bounded set P, it has an accumulation point $\bar p$ in P.

Assumption. For any sequence (t^j,x_0,p_0^j) and $\bar p=\lim_{j\to\infty}p_0^j$ the following holds:

$$z\lim_{j\to\infty}(t^j,x_0,p_0^j)=\lim_{j\to\infty}(t^j,x_0,\bar p).$$

This assumption will prevent situations such as that depicted in [9].

The next lemma states that any accumulation point must lie on the stable manifold of (2.3).

LEMMA 4.2. *If $F^i\to 0$, then any accumulation point of p_0^i lies on the stable manifold. Furthermore, there exists exactly one accumulation point, i.e., p_0^i is a convergent sequence.*

Proof. Define p_0^j as a subsequence of p_0^i which converges to an accumulation point $\bar p$. Then

$$\begin{aligned}\lim_{j\to\infty}\|z(t^j,x_0,p_0^j)\|&=\|\lim_{j\to\infty}z(t^j,x_0,p_0^j)\|\\&=\|z(\lim_{j\to\infty}t^j,x_0,\lim_{j\to\infty}p_0^j)\|\\&=\lim_{t\to\infty}\|z(t,x_0,\bar p)\|,\end{aligned} \tag{4.3}$$

which goes to zero from Lemma 4.1. Therefore, $\bar p$ must lie on the stable manifold.

Now suppose two accumulation points $\bar p_1$, $\bar p_2$ exist. Both points must lie on the stable manifold, but x_0 was chosen to lie inside the neighborhood where the stable manifold can be written as a graph, i.e., $M^-\triangleq p=V_x^-(x_0)$. Therefore, $\bar p_1=\bar p_2=p^*$, and the entire sequence converges to the stable manifold.

Together, the above insure convergence of the algorithm to the stabilizing solution of (2.2). The following lemma, which is proved in [9], [10], gives an upper bound on the rate of convergence.

LEMMA 4.3. *Define λ_1 as the eigenvalue with the largest negative real part of $\bar H$. Then*

$$\|p_0^i-p_0^*\|=O(e^{\mathrm{Re}\lambda_1 t}). \tag{4.4}$$

The proof of Lemma 4.3 is based on the variations of constants formula applied to the Hamiltonian system and the bound is not sharp. However, if the nonlinearities of the Hamiltonian system are removed, i.e., $\dot z=\bar Hz$, the sequence of costates generated by the algorithm can be calculated analytically and the convergence rate can be found directly. While this does not provide a "true" rate of convergence for the nonlinear case, by moving x_0 close to the origin, the nonlinear terms can be considered as a perturbation of the linearized Hamiltonian, and the following result [10] can be viewed as a better estimate for the convergence of the algorithm.

LEMMA 4.4. *For a fixed time, the minimization problem*

$$\min_{p_0} F(t, x_0, p_0) = z_0 e^{\bar{H}^T t} e^{\bar{H} t} z_0$$
$$\text{such that} \quad (I \quad 0) z_0 = x_0$$

can be directly solved. Furthermore, as $t \to \infty$,

$$\|p_0(t) - p_0^*\| = O(e^{2\mathrm{Re}\lambda_1 t}).$$

Hence, if x_0 is chosen such that the nonlinear terms have a "small" effect on the dynamics, the convergence rate can effectively double. This can be seen in the following example.

4.1. Example. Consider the H_∞ problem for the nonlinear system

$$\dot{x} = Ax + \epsilon f(x) + Bu,$$
$$y = Cx.$$

Increasing ϵ is equivalent to strengthening the effect of the nonlinear terms on the system dynamics.

Define

$$A = \begin{pmatrix} 0 & 1 \\ -1 & -2 \end{pmatrix}, \quad B = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \quad C = \begin{pmatrix} 1 & 0 \end{pmatrix},$$

$$f(x) = \begin{pmatrix} x_1^2 \\ x_1 x_2 - x_2^2 \end{pmatrix}, \quad \gamma = 1.3.$$

The algorithm was used to solve the corresponding Hamilton–Jacobi equation at the point $(0.1 \quad 0)^T$ for different values of ϵ. For all cases, the first terminal time in the sequence t^i in (2.8a) used in the program was 2.0. Since the solution to the Hamilton–Jacobi equation is not known when $\epsilon \neq 0$, for all cases the simulation was run until five significant digits were obtained in the solution. The resulting state trajectories are given in Figure 2.

This indicates that the algorithm is indeed finding a stabilizing trajectory. Figure 3 shows the error graph of the algorithm for each iteration for each value of ϵ.

The eigenvalues of the corresponding Hamiltonian matrix are $\pm 1.31, \pm 0.48$. For the linear case $\epsilon = 0$, the straight line of the error curve on the semilogarithmic axis implies exponential convergence with respect to the iteration variable. The slope of the line indicates the rate of convergence, which can be calculated using a least square fit. In this case, the rate of convergence is -0.96, which is indeed $O(e^{2\lambda_1 t})$ as expected. As ϵ increases, the nonlinear terms have a stronger effect on the algorithm, and the rate of convergence decreases accordingly.

5. Illustrative examples.

5.1. Optimal control problem with nonquadratic cost. The first example is a linear system with a nonquadratic cost function. The example is from [5],

$$\dot{x} = -ax + u, \tag{5.1}$$
$$J = \int_0^\infty x^2 + x^4 + u^2 dt.$$

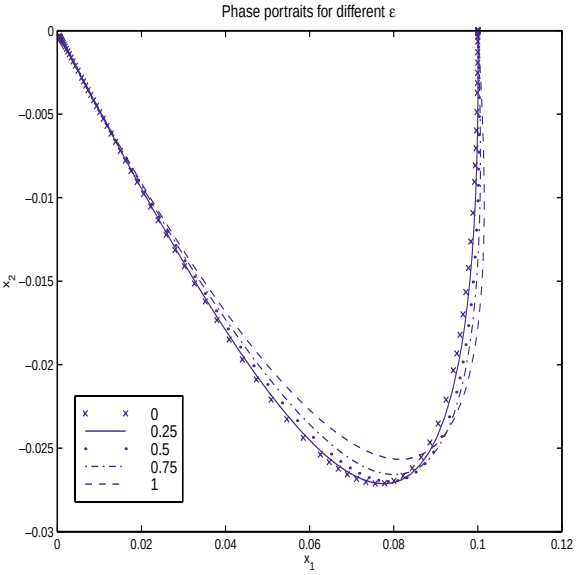


FIG. 2. *State trajectories of system with algorithmic control.*

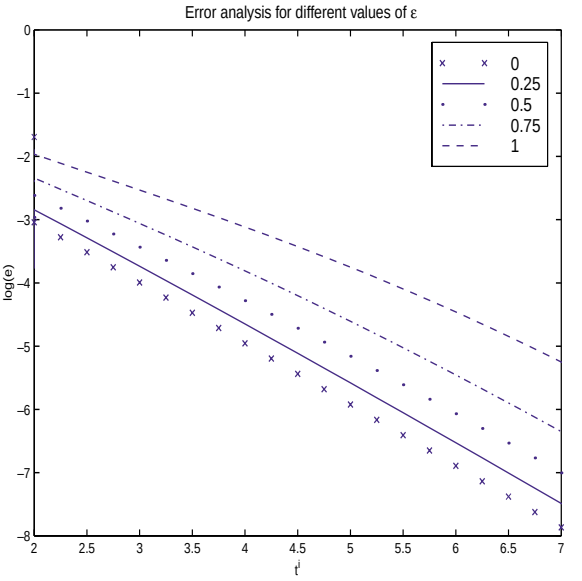


FIG. 3. *Effect of nonlinear terms on convergence.*

The corresponding Bellman equation is

$$H^*(x, V_x) = -axV_x - \frac{1}{4}V_x^2 + x^2 + x^4 = 0, \tag{5.2}$$

and the optimal control of (5.1) is $u^* = -\frac{1}{2}V_x^-$, where

$$V_x^-(x) = -2x(a - \sqrt{a^2 + 1 + x^2}). \tag{5.3}$$

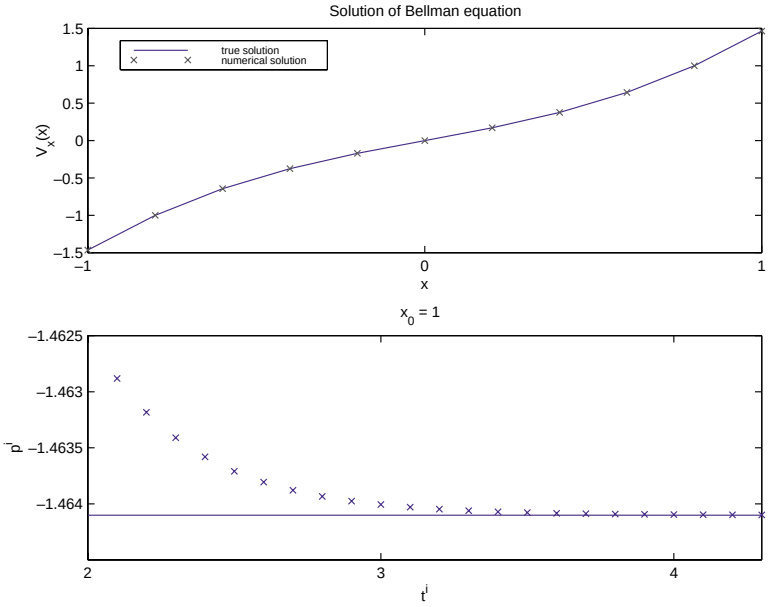


FIG. 4. *Numerical solution for system* (5.1).

In the case $a = 1$, (5.2) was solved by the algorithm for $x \in [-1, 1]$. The initial guess for $V_x^-(x)$ was chosen to be $-2x$. The starting time in the sequence t^i in (2.8a) was chosen as $t^0 = 2$ and the times were increased at each iteration by 0.1.

The top graph of Figure 4 shows that the algorithm finds the stabilizing solution of (5.2) for different fixed values of x. The graph on the bottom shows the progress of the algorithm to the correct solution for $x = -1$, which is $V_x^-(-1) = -1.46410$.

Figure 5 displays two error estimates used when the true solution is not known in advance. The graph on the top displays the first error estimate which is the residual of the Bellman equation for each minimizing initial condition p_0^i. Since the algorithm seeks a solution to the Bellman equation, each iteration should cause the residual to decrease to zero. The bottom graph shows the second error estimate which shows the progress of the distance function $F(t^i, p_0^i) = \|x(t^i, x_0, p_0^i)\|^2 + \|p(t^i, x_0, p^i)\|^2$ as t^i increases. Clearly, this function should monotonically decrease to zero as well as t^i grows bigger.

5.2. A two-dimensional linear H_∞ problem. For a linear system

$$\begin{aligned} \dot{x} &= Ax + Bu, \\ y &= Cx, \end{aligned} \tag{5.4}$$

the Hamilton–Jacobi–Isaacs equation whose solution insures the $\mathcal{L}_2$ gain from u to y is less than or equal to γ reduces to the algebraic Ricatti equation (2.9). Specifically, with $k = 4\gamma^2$, the stabilizing solution is $V_x^- = x^T X$ and the corresponding control is $u^* = \frac{1}{2\gamma^2} B^T (V_x^-)^T = \frac{1}{2\gamma^2} B^T X x$.

Although (2.9) can be solved directly, the optimal control can also be found at a

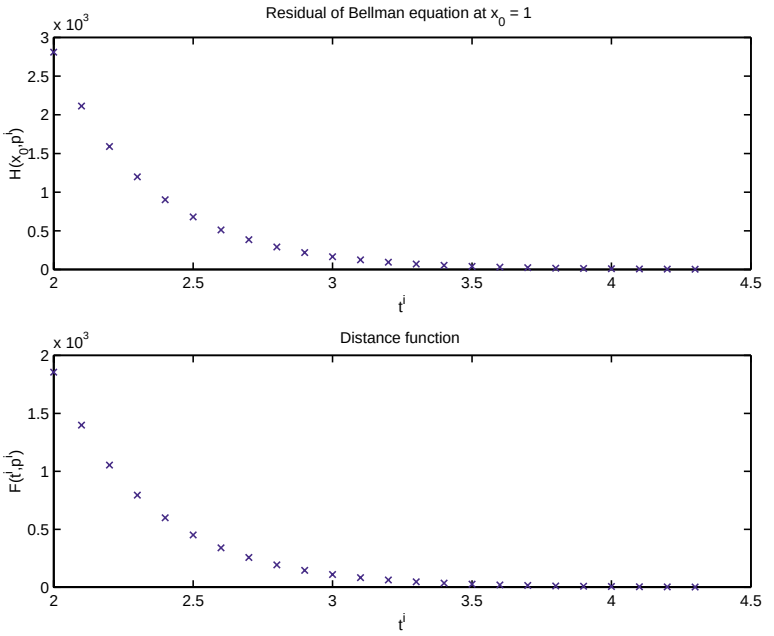


FIG. 5. *Error functions for system* (5.1).

specific initial condition by the algorithm. Let

$$A = \begin{pmatrix} 0 & 1 \\ -1 & -2 \end{pmatrix},$$
$$B = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \tag{5.5}$$
$$C = \begin{pmatrix} 1 & 0 \end{pmatrix},$$

and set $\gamma = 1.3$. The algorithm was run at the initial point $x = (1 \quad 0)^T$, with initial guess for p^* arbitrarily chosen to be $(1 \quad 1)^T$. The first time in the sequence t^i in (2.8a) used in the program was $t^0 = 5$, and the times were increased at each iteration by 0.25. The TOCA was not used.

The true solution $V_x(x) = (2.839 \quad 1.220)^T$ is found by directly solving the algebraic Ricatti equation. Figure 6 shows the convergence of each component of p^i to the true solution. The bottom two graphs show the distance function F and the residual function H^*, which indicate that the algorithm is indeed converging to a stabilizing solution of (2.9).

6. Convergence rate versus eigenvalues of a Hamiltonian. Lemma 4.4 shows that for linear systems,

$$e^i = \|p_0^i - p_0^*\| \approx Ce^{2\,\mathrm{Re}\,\lambda_1 t}, \tag{6.1}$$

where λ_1 is the eigenvalue with largest negative real part of $\bar{H}$. Therefore,

$$ln(e^i) \approx lnC + 2\,\mathrm{Re}\,\lambda_1 t^i. \tag{6.2}$$

So if the graph of the error function versus the sequence t^i is a straight line on semilog axes, then the error is truly decreasing exponentially as the times in the algorithm

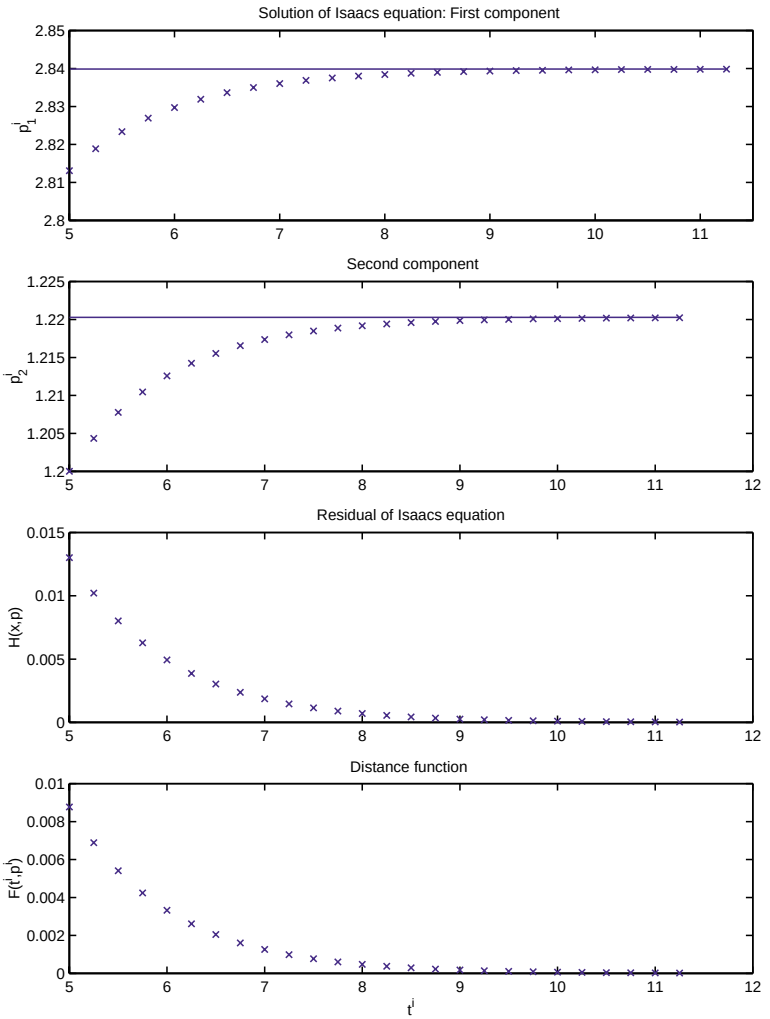


FIG. 6. *Solution and error analysis for* (2.9).

increase. Furthermore, the slope of the line will determine the time constant or the rate of decay.

The Hamiltonian matrix associated with (5.2) is

$$H = \begin{pmatrix} -a & -\frac{1}{2} \\ -2 & a \end{pmatrix}. \tag{6.3}$$

H has eigenvalues $\pm\sqrt{a^2+1}$. Running the algorithm at $x = 1$ for different values of a, Figure 7 displays the logarithmic error graph. Clearly, the curves are all linear, which imply exponential convergence. Furthermore, as a increases the slope of each line increases, indicating a faster rate of convergence. By finding the least square fit to each curve, the rate of convergence can be tested. Table 6.1 shows how the eigenvalues of the Hamiltonian and the convergence rate of the algorithm vary as a increases.

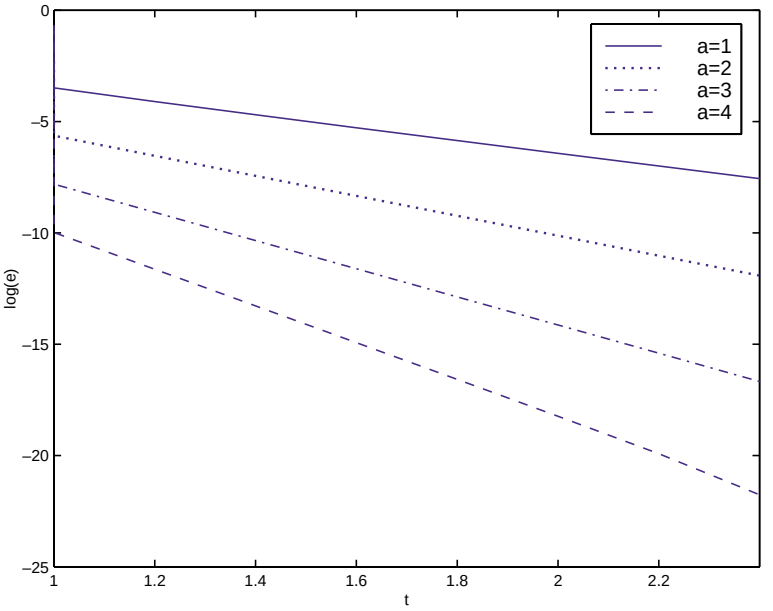


FIG. 7. *Error at each iteration for different a.*

TABLE 6.1
Eigenvalues of H versus rate of convergence.

a	λ	2λ	m_{calc}
1	$\sqrt{2}$	-2.83	-2.83
2	$\sqrt{5}$	-4.47	-4.46
3	$\sqrt{10}$	-6.32	-6.37
4	$\sqrt{17}$	-8.24	-8.25

Clearly, the rate of convergence is consistently twice the eigenvalue of the Hamiltonian matrix.

For higher dimensional systems, this convergence ration depends on the eigenvalue with largest negative real part eigenvalue of the Hamiltonian. Consider the linear H_∞ problem (5.4), (5.5). The associated Hamiltonian system of the Isaacs equation is

$$\begin{pmatrix} \dot{x} \\ \dot{p} \end{pmatrix} = \begin{pmatrix} A & \frac{1}{2\gamma^2} BB^T \\ -2C^TC & -A^T \end{pmatrix} \begin{pmatrix} x \\ p \end{pmatrix}, \tag{6.4}$$

and the eigenvalues of (6.4) are $\pm 1.31, \pm 0.48$. Figure 8 shows the graph of the error versus the time on semilog axes. The straight line indicates that the convergence is indeed exponential, and the slope confirms that the rate is approximately double the minimum eigenvalue of H.

6.1. A nonlinear second-order system. Solve the optimal control problem

$$\begin{aligned} \dot{x_1} &= x_2, \\ \dot{x_2} &= \epsilon x_1^3 + u, \end{aligned} \tag{6.5}$$

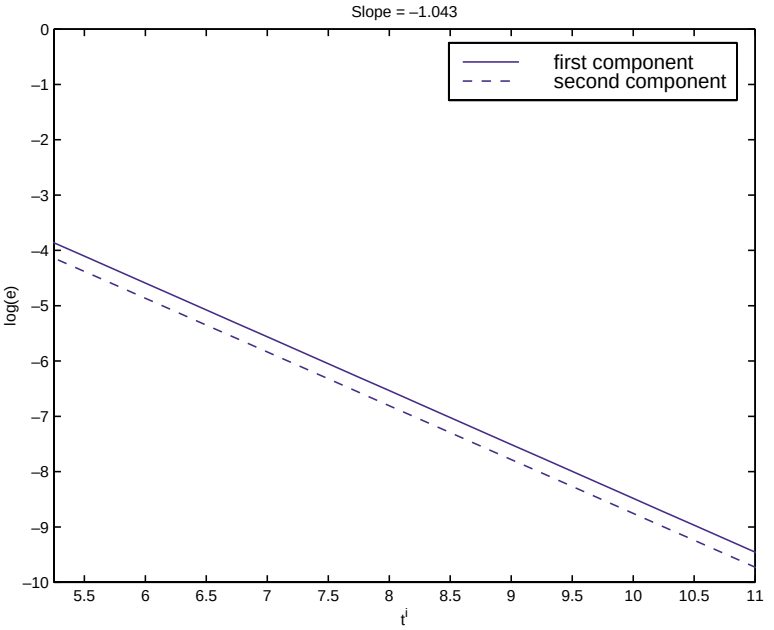


FIG. 8. *Convergence rate of algorithm for system* (5.5).

subject to to the performance index

$$J = \frac{1}{2}\int_0^\infty x_1^2 + x_2^2 + u^2 dt. \tag{6.6}$$

This problem was solved in Nishikawa [5] using a perturbation method, and by Beard [7] using a successive approximation scheme. Results using all three methods will be compared when $\epsilon = 1$.

The corresponding Bellman equation for this problem is

$$H \triangleq x_2 p_1 + x_1^3 p_2 - \frac{1}{2}p_2^2 + \frac{1}{2}x_1^2 + \frac{1}{2}x_2^2 = 0, \tag{6.7}$$

and the optimal control is simply $u^* = p_2$.

Nishikawa and Beard develop fourth-order approximations of the optimal control. They are

$$\begin{aligned} u_n(x) = &- x_1 - 1.7321x_2 - x_1^3 - 1.3235x_1^2x_2 - 0.6923x_1x_2^2 - 0.1323x_2^3, \\ u_b(x) = &- 1.0376x_1 - 1.7975x_2 - 1.3079x_1^3 - 1.3429x_1^2x_2 \\ &- 0.4664x_1x_2^2 - 0.74x_2^3. \end{aligned} \tag{6.8}$$

These controls are compared to the numerical control obtained by our algorithm for the initial conditions $x_1(0) \in (-1, 1)$ and $x_2(0) = 0$. The algorithm was run until five digits in the solution were obtained, and the results are shown in Figure 9. In the figure, J_A refers to the controls obtained by the algorithm, J_B to Beard's control, and J_N to Nishikawa's control.

The first plot shows the associated costs for all three methods. These graphs are indistinguishable, so the differences are plotted as well. It is seen that the cost

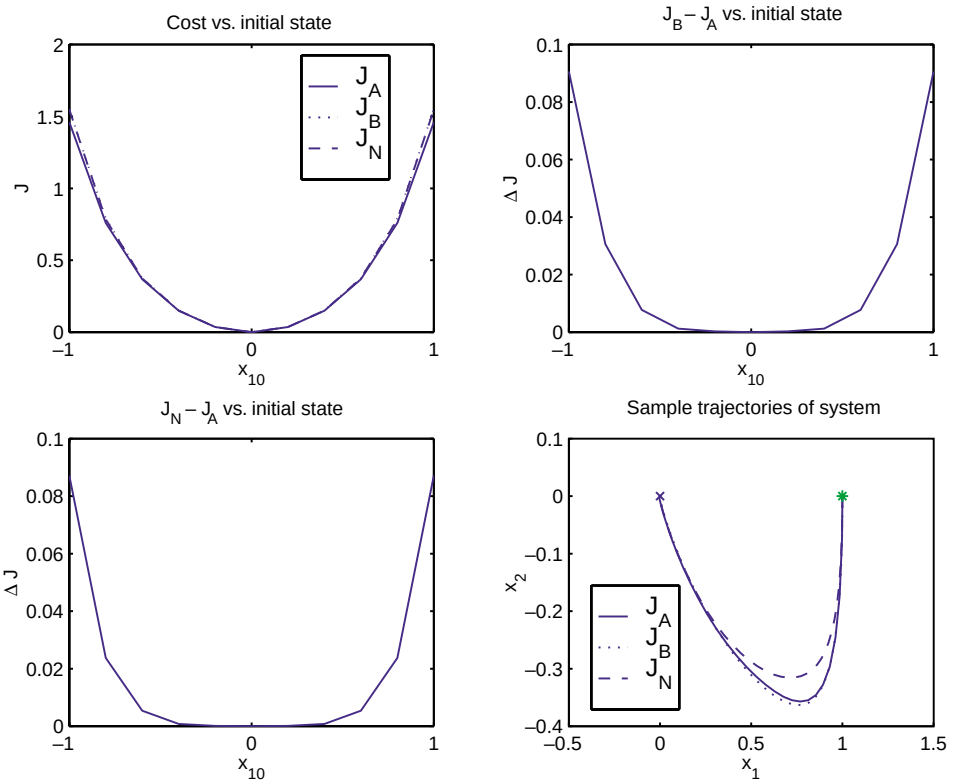


FIG. 9. *Comparison with perturbation method.*

obtained by this algorithm is comparable to the other methods. However, by strengthening the stopping criteria, the costs under our method can be decreased even further at an extra computational cost. Finally, the graph on the bottom right shows a sample trajectory of the system using all three controls.

6.2. Forming the surface numerically. Our algorithm does not find an analytic expression for the control in terms of the state x_0 like the other two methods. However, the control function can be computed numerically by choosing enough points around the origin and integrating the Hamiltonian system forward and backward in time. The forward time trajectory brings the characteristic towards the origin and the backward time trajectory propagates the solution outward. Figure 10 shows the projection of the Hamiltonian system on the state space (x_1, x_2) of the plant using various initial conditions along the line $x_2 = 0$.

It should be apparent that any section of the state space can be "covered" by characteristics given a sufficient number of initial points by the algorithm.

Recall the Hamiltonian system also solves for $V(x)$ and its derivatives as well. Therefore, the value function over the entire region can be estimated by using interpolation methods between the characteristics. Also, since in this case $u(t) = p_2(t)$, a numerical solution for the feedback control can be found as well. Figure 11 shows the graphs of both functions.

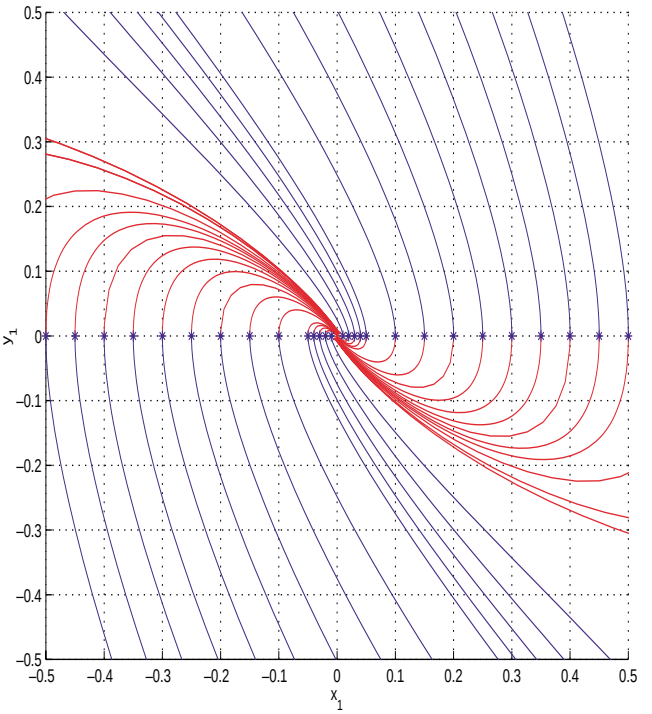


FIG. 10. *Trajectories of Hamiltonian around the origin.*

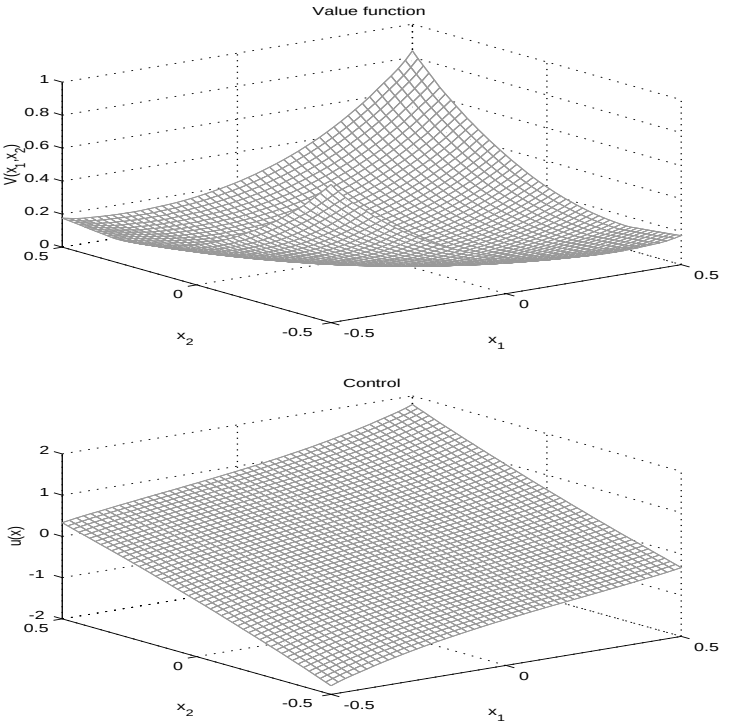


FIG. 11. *$V(x)$ and $u(x)$ for optimal control problem.*

7. Conclusions. There is a need for numerical methods which approximate solutions to the special types of equations which arise in nonlinear control theory. This paper presents an algorithm which calculates an open-loop solution to these Hamilton–Jacobi type equations for points sufficiently close to the origin.

REFERENCES

[1] A. Bryson and Y.-C. Ho, *Applied Optimal Control; Optimization, Estimation, and Control*, Blaisdell Publishing, Waltham, MA, 1969.

[2] A. Isidori, *Attenuation of disturbances in nonlinear control systems*, in Systems, Models and Feedback, A. Isidori and T. Tarn, eds., Progr. Systems Control Theory 12, Birkhäuser Boston, Boston, MA, 1992, pp. 275–300.

[3] H. Knobloch, A. Isidori, and D. Flockerzi, *Topics in control theory*, DMV Seminar 22, Birkhäuser-Verlag, Basel, Switzerland, 1993.

[4] W. L. Garrard, *Additional results on suboptimal feedback control for nonlinear systems*, Internat. J. Control, 10 (1969), pp. 657–663.

[5] Y. Nishikawa, *A method for suboptimal design of nonlinear feedback systems*, Automatica, 7 (1971), pp. 703–712.

[6] G. N. Saridis and C.-S. G. Lee, *An approximation theory of optimal control for trainable manipulators*, IEEE Trans. Systems Man. Cybernet., 9 (1979), pp. 152–159.

[7] R. Beard, G. Saridis, and J. Wen, *Improving the performance of stabilizing controls for nonlinear systems*, IEEE Control Systems Magazine, (1996), pp. 27–35.

[8] K. A. Wise and J. L. Sedwick, *Missile autopilot design using nonlinear h_∞-optimal control*, Proceedings of 13th IFAC Symposium on Automatic Control in Aerospace, Palo Alto, CA, 1994.

[9] J. Markman and I. N. Katz, *Convergence of an iterative algorithm for solving Hamilton Jacobi type equations*, Math. Comp, to appear.

[10] J. Markman, *Numerical solutions of the Hamilton-Jacobi equations arising in nonlinear H_∞ and optimal control*, D.Sc. thesis, Washington University, Department of Systems Science and Mathematics, Pullman, WA, 1998.

SIAM J. SCI. COMPUT.
Vol. 22, No. 1, pp. 330–350
© 2000 Society for Industrial and Applied Mathematics

A NEW LAGRANGIAN METHOD FOR TIME-DEPENDENT INVISCID FLOW COMPUTATION*

C. Y. LOH† AND W. H. HUI‡

Abstract. This paper presents a new Lagrangian approach for the two-dimensional (2-D) time-dependent Euler equations. It may be considered as a sequel to the authors' previous Lagrangian approaches for steady supersonic flow computations [C. Y. Loh and W. H. Hui, *J. Comput. Phys.*, 89 (1990), pp. 207–240; W. H. Hui and C. Y. Loh, *J. Comput. Phys.*, 103 (1992), pp. 450–464; W. H. Hui and C. Y. Loh, *J. Comput. Phys.*, 103 (1992), pp. 465–471; C. Y. Loh and M. S. Liou, *J. Comput. Phys.*, 104 (1993), pp. 150–161; C. Y. Loh and M. S. Liou, *SIAM J. Sci. Comput.*, 15 (1994), pp. 1038–1058; C. Y. Loh and M. S. Liou, *J. Comput. Phys.*, 113 (1994), pp. 224–248]. The theoretical background and the intrinsic flow coordinates as well as the Lagrangian conservation form are introduced based on the concept of *material functions* (or *path functions*). A TVD scheme of the Godunov type is chosen to describe the numerical procedure. Several examples are then given to justify the claimed advantages of the new methodology, namely, (a) any contact discontinuities are crisply solved and (b) grids are automatically and accurately generated following pathlines.

Key words. Lagrangian description, TVD scheme, inviscid unsteady flow

AMS subject classifications. 65C20, 65M05, 76N10, 70D10, 76L05

PII. S1064827597330856

1. Introduction.

It is well known that there exist two self-contained general formulations of studying fluid flow: the Eulerian formulation observes flow at fixed locations whereas the Lagrangian formulation does so following fluid particles.

Most of the existing work uses the Eulerian formulation in which the computational cells are fixed in space, while fluid particles move across cell interface. It is this convective flux that causes severe numerical diffusion in the numerical solutions using Eulerian formulation. Indeed, contact discontinuities or slip lines are smeared badly and shocks are also smeared, albeit somewhat better than contact discontinuities. Moreover, the smearing of contact increases with time and distance as a result of continuous mixing of fluid particles. This process exhibits the nature of Eulerian formulation. It cannot be stopped by simply reducing the artificial damping of the numerical scheme. In [1], it is estimated, for example, for a second-order scheme, the smearing is in the order of $O(n^{1/3})$, where n is the number of time steps. Higher order schemes such as piecewise parabolic method (PPM) and essentially nonoscillatory (ENO) may have better performance, but the process still exists as a result of the fundamental nature of the Eulerian approach. Special treatments or numerical fixes, such as artificial compression, steeper reconstruction, and subcell resolution [1], were then introduced in association with higher order schemes (PPM, ENO, etc.). For many problems, great improvements are achieved. However, these numerical fixes either have side-effects or are not always reliable. Since the sixties, the primary efforts of computational fluid dynamics (CFD) algorithm researchers have been concentrated

*Received by the editors December 1, 1997; accepted for publication (in revised form) May 4, 1999; published electronically June 22, 2000. Similar materials were previously presented at the 13th AIAA CFD Conference as AIAA paper 97-1974. This work was partially supported by a grant from the Research Grants Council of Hong Kong.

http://www.siam.org/journals/sisc/22-1/33085.html

†M.S. 5-11, NASA Glenn Research Center & Taitech Inc., Cleveland, OH 44135 (fsloh@turbot.grc.nasa.gov).

‡Department of Mathematics, Hong Kong University of Science and Technology, Clear Water Bay, Hong Kong (whhui@uxmail.ust.hk).

on developing better (more robust, accurate, and efficient) ways to deal with this convective flux. Although great progresses have been made, numerical diffusion still exists, causing inaccuracy, and is even more difficult to handle in multidimensional flow problems.

Computational cells in the Lagrangian formulation, on the other hand, are literally fluid particles. Consequently, there is no convective flux across cell boundaries and the numerical diffusion is thus minimized. Fundamentally, Lagrangian formulation is ideal for crisp contact resolution. In addition, flow physics is closely followed in a Lagrangian approach and the numerical scheme may benefit from it. The most famous Lagrangian method in use at the present time is the arbitrary Lagrangian–Eulerian method (ALE) [2]. The ALE begins with the first principles of flow physical laws, and the grid nodes are advanced in space and time according to their corresponding velocities. Most modern Lagrangian methods follow the same idea of using staggered grids with velocities at nodes and other quantities at cell centers. For any Lagrangian approach, however, in some cases (e.g., vortical flow with roll-up) the very essence of allowing computational cells to exactly follow the fluid particles can result in cell distortion so severe that the grid lines cross one another and cause the failure of the numerical process. To prevent this from happening, the ALE makes a step back to the Eulerian phase by using continuous rezoning.

Along the line of Lagrangian approach, recently, Hui and van Roessel [3] and van Roessel and Hui [4] developed a new idea in reformulating the Euler equations of gas dynamics and Navier–Stokes equations and thus laid the foundation of a new Lagrangian description for fluid dynamics. They utilize three material functions, M^1, M^2, M^3, to replace the conventional particle labels x_0, y_0, z_0 (typically, the initial position of the particle) as independent variables; then the fourth independent variable τ turns out to have the same dimension as time, which they term the *Lagrangian time*. A *material function* or *path function* [5] is defined as a function that remains constant along a fluid particle path. In the case of steady flow, the material functions become identical to stream functions [4] and their number reduces from three to two. The following is a brief account of their derivation for two-dimensional (2-D) unsteady inviscid flows. The purpose is to show the existence of the new coordinate system. Tensor notations are used for precise and compact presentation.

The derivation begins with an attempt to eliminate the continuity equation. In Cartesian coordinates $x^1 - x^2$ (standing for the usual $x - y$), let $x^0 = t$ be the time of motion, v^0=1, v^i, $i = 1, 2$, the x^1 and x^2 components of velocity (namely, u, v), and ρ the density. Then, the continuity equation can be written as

$$(\rho v^i)_{,i} = 0, \quad i = 0, 1, 2.$$

The material functions $M^i(x^0, x^1, x^2),\ i = 1, 2$, are defined as functions satisfying the following equations:

$$\epsilon^{ijk} K(M^1, M^2) \frac{\partial M^1}{\partial x^j} \frac{\partial M^2}{\partial x^k} = \rho v^i, \quad i, j, k = 0, 1, 2, \tag{1}$$

where ϵ^{ijk} is the permutation symbol and $K(M^1, M^2)$ is a function related to the Jacobian by

$$\frac{\rho}{K} = \frac{\partial(M^1, M^2)}{\partial(x^1, x^2)}. \tag{2}$$

Equation (2) is obtained by setting $i = 0$ in (1). By substituting (1) in the continuity equation, it is observed that the continuity equation is identically satisfied and thus can be eliminated:

$$(\rho v^i)_{,i} = \epsilon^{ijk} \frac{\partial}{\partial x^i} \left[K \frac{\partial M^1}{\partial x^j} \frac{\partial M^2}{\partial x^k} \right] = 0, \quad i, j, k = 0, 1, 2.$$

Rewrite (1) using $I, J, K = 0, 1, 2$ in place of i, j, k:

$$v^I = \frac{K}{\rho} \epsilon^{IJK} \frac{\partial M^1}{\partial x^J} \frac{\partial M^2}{\partial x^K},$$

$M^i, i = 1, 2$, are readily verified to be material functions since

$$\frac{DM^i}{Dt} = v^I \frac{\partial M^i}{\partial x^I} = \frac{K}{\rho} \epsilon^{IJK} \frac{\partial M^i}{\partial x^I} \frac{\partial M^1}{\partial x^J} \frac{\partial M^2}{\partial x^K} = 0. \tag{3}$$

Now, consider $M^i, i = 1, 2$, as new independent variables and choose a third one, M^0, in such a way that the new Jacobian remains the same as before,

$$\frac{\partial(M^0, M^1, M^2)}{\partial(x^0, x^1, x^2)} = \frac{\partial(M^1, M^2)}{\partial(x^1, x^2)} = \frac{\rho}{K}.$$

Then, there are totally three new independent variables $M^i, i = 0, 1, 2$. The inverse transformation provides

$$\frac{\partial x^i}{\partial M^0} = \frac{K}{\rho} \epsilon^{ijk} \frac{\partial M^1}{\partial x^j} \frac{\partial M^2}{\partial x^k}, \quad i, j, k = 0, 1, 2.$$

From (1), the above relations result in

$$v^i = \frac{\partial x^i}{\partial M^0}, \quad i = 1, 2; \qquad v^0 = 1 = \frac{\partial x^0}{\partial M^0} = \frac{\partial t}{\partial M^0}.$$

In other words, M^0 plays a role as time. It is thus termed the Lagrangian time and denoted by τ. τ is indeed a physical time. Following a fluid particle, τ differs from t only by a constant and may be considered as the time individually specified for this fluid particle. The material derivative now reduces to

$$\frac{D}{Dt} = v^i \frac{\partial}{\partial x^i} = \frac{\partial x^i}{\partial \tau} \frac{\partial}{\partial x^i} = \frac{\partial}{\partial \tau}. \tag{4}$$

Equation (4) shows that the new coordinate system (τ, M^1, M^2) is truly Lagrangian. By now, the existence of the new Lagrangian coordinate system is justified.

The major difference between the new and conventional Lagrangian formulations lies in the choice of fluid particle labels. In the conventional Lagrangian description of fluid motion, fluid particle labels x_0, y_0, and z_0 being the independent variables are often chosen from the initial fluid particle positions (e.g., in ALE [2]). In contrast, the new Lagrangian description is characterized by employing material functions (or stream functions in the steady flow case) as fluid particle labels. It turns out that by so doing, an intrinsic flow coordinate is established and a direct and exact relation between flow physics (velocity) and flow geometry—the compatibility equation is found, while such relation is only expressed approximately and implicitly in the conventional Lagrangian formulation (such as the Lagrangian phase of ALE [2]). When

written in conservation form, this relation becomes a geometrical conservation law. As a consequence of the new Lagrangian formulation, we are able to use an unsplit, nonstaggered grid, and to utilize the well-established modern high resolution shock-capturing schemes.

In general, it is observed that the new Lagrangian approach possesses the following features compared to its Eulerian counterpart:

- A computational cell is literally a fluid particle; consequently physics is closely followed.
- Contact discontinuity resolution is crisp and never becomes smeared.
- Grids are automatically generated following particle pathlines or streamlines. The geometrical conservation law assures the correct deformation of each grid cell.
- On the other hand, as in any (pure, nonrezoned) Lagrangian approach, the numerical procdure may fail when the computational cells are highly distorted and grid line crossing occurs (e.g., when vortical roll-up occurs).

Currently our research on the new Lagrangian approach is mostly concentrated in applying finite difference approach in the steady supersonic flow regime. The present authors [6, 7, 8] first put forward a Lagrangian conservation form and applied a Godunov/TVD scheme in 2-D steady supersonic flow computation. In the meantime, they also successfully apply a random choice method [9] for the 2-D supersonic flow problems. In [8], they developed a shock cell splitting technique to exactly locate the shock, which was recently further improved by LePage and Hui [10]. Loh and Liou [11, 12] applied the finte difference and random choice methods for 2-D real gas problems. Recently, they [13, 14] extended the TVD scheme to solve three-dimensional (3-D) supersonic flow problems and obtained interesting results.

In this paper, we report our research on the Lagrangian approach for 2-D unsteady inviscid flow, which includes the new Lagrangian formulation, the corresponding conservation form, and the TVD numerical scheme. The contents are organized in the following way. In section 2, the new conservation forms for 2-D unsteady flows are derived. Section 3 is devoted to the numerical method applied in the new Lagrangian approach. In section 4, several typical numerical results are presented to show the advantages of the new methodology. Finally the paper is concluded in section 5.

2. Lagrangian conservation form for 2-D unsteady flows. The new Lagrangian coordinates may be considered as the *intrinsic* flow coordinates, which is analogous to the intrinsic (or natural) coordinates for a spatial curve in differential geometry. From the viewpoint of numerical computation, we are more interested in the conservation form of the governing equations. Although such a Lagrangian conservation form can be derived via the first principles of physics, we prefer a derivation based on applying coordinate transformation to the Euler equations (5). By so doing, the compatibility equations, which provide geometrical conservation laws for grid deformation, are directly available. The coordinate transformations from Cartesian variables (t, x, y) to Lagrangian variables (τ, ξ, η) (hereafter, M^1, M^2 are replaced by ξ, η, respectively) lead to a *compact* form of the Euler equations under the new system. The Eulerian conservation laws for 2-D unsteady flows are

$$\frac{\partial \mathbf{E}}{\partial t} + \frac{\partial \mathbf{F}}{\partial x} + \frac{\partial \mathbf{G}}{\partial y} = 0, \tag{5}$$

where

$$\mathbf{E} = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ \rho e \end{pmatrix}, \qquad \mathbf{F} = \begin{pmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ (\rho e + p)u \end{pmatrix}, \qquad \mathbf{G} = \begin{pmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ (\rho e + p)v \end{pmatrix}.$$

As usual, t, u, v, ρ, and p are, respectively, time, the Cartesian components of flow velocity, density, and pressure of the fluid obeying the γ-law; the special total energy

$$e = \frac{1}{2}(u^2 + v^2) + \frac{1}{(\gamma - 1)}\frac{p}{\rho}.$$

Note that the γ-law is not a condition for converting (5) into a Lagrangian form but is a convenient choice since it is set in our Riemann solver. Let $\mathbf{r} = (x, y)^T$, $\mathbf{V} = (u, v)^T$ denote the Cartesian coordinates and the fluid velocity, respectively. Then,

$$\mathbf{V} = \frac{D\mathbf{r}}{Dt} = \frac{\partial \mathbf{r}}{\partial \tau}. \tag{6}$$

Furthermore, we define the following Lagrangian quantities:

$$\mathbf{T} = (U, V)^T = \frac{\partial \mathbf{r}}{\partial \xi}, \qquad \mathbf{S} = (X, Y)^T = \frac{\partial \mathbf{r}}{\partial \eta}. \tag{7}$$

These quantities represent the lateral displacement gradients of a fluid particle (or a quadrilateral computational cell). Note that they should not be confused with the lower case flow velocities (u, v) and coordinates (x, y). We also define

$$W = \frac{\partial t}{\partial \xi}, \qquad Z = \frac{\partial t}{\partial \eta}. \tag{8}$$

Then the Jacobian matrix of the transformation is

$$J = \frac{\partial(t, x, y)}{\partial(\tau, \xi, \eta)} = \begin{pmatrix} \partial t/\partial \tau & \partial t/\partial \xi & \partial t/\partial \eta \\ \partial x/\partial \tau & \partial x/\partial \xi & \partial x/\partial \eta \\ \partial y/\partial \tau & \partial y/\partial \xi & \partial y/\partial \eta \end{pmatrix} = \begin{pmatrix} 1 & W & Z \\ u & U & X \\ v & V & Y \end{pmatrix}. \tag{9}$$

Notice that

$$\frac{\partial W}{\partial \tau} = \frac{\partial}{\partial \tau}\left(\frac{\partial t}{\partial \xi}\right) = \frac{\partial}{\partial \xi}\left(\frac{\partial t}{\partial \tau}\right) = 0, \qquad \frac{\partial Z}{\partial \tau} = \frac{\partial}{\partial \tau}\left(\frac{\partial t}{\partial \eta}\right) = \frac{\partial}{\partial \eta}\left(\frac{\partial t}{\partial \tau}\right) = 0.$$

In other words, W and Z are independent of τ during marching forward in time. For simplicity, we let $W = Z = 0$ identically. The Jacobian matrix J is now reduced to

$$J = \begin{pmatrix} 1 & 0 & 0 \\ u & U & X \\ v & V & Y \end{pmatrix}, \tag{10}$$

and its determinant becomes

$$|J| = \begin{vmatrix} U & X \\ V & Y \end{vmatrix}. \tag{11}$$

Equation (11) denotes the volume ratio during the change of independent variables. Subsequently,

$$K = \rho|J| = \rho \begin{vmatrix} U & X \\ V & Y \end{vmatrix} \tag{12}$$

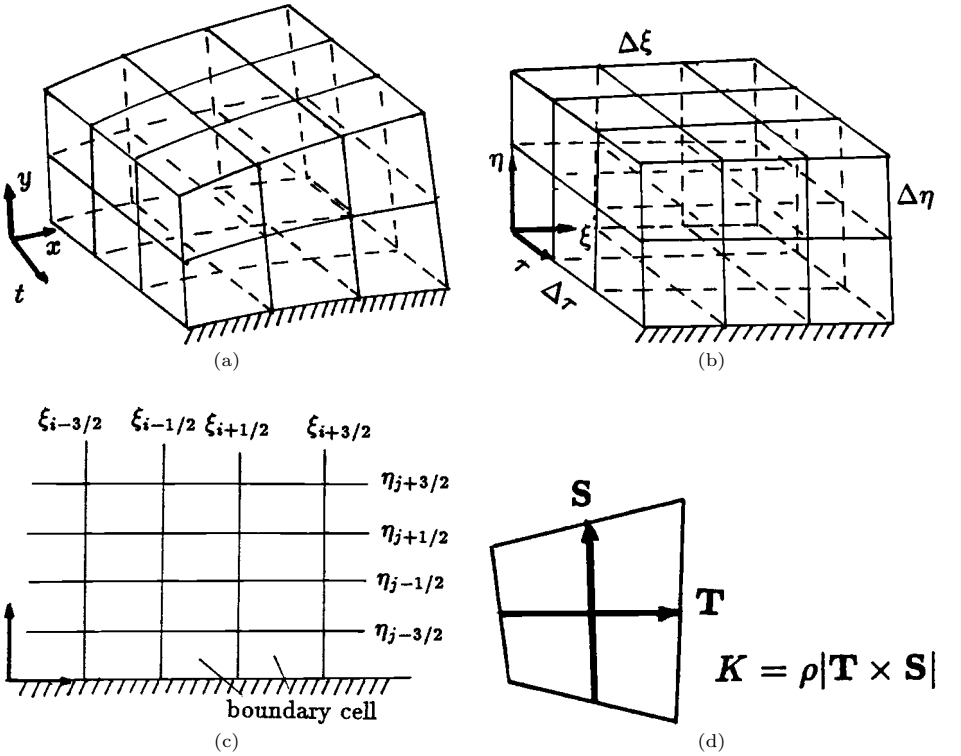


FIG. 1. *Physical and computational space-time domains.* (a) *Physical domain,* (b) *computational domain,* (c) *cell layout on a typical* $\xi - \eta$ *plane,* (d) *mass flux of a cell.*

is the mass flux (Figure 1(d)). The function K thus derived is identical to the 2-D definition of (2). From (12), $|J| = K/\rho$, and the Jacobian matrix of the inverse transformation

$$J^{-1} = \frac{\partial(\tau,\xi,\eta)}{\partial(t,x,y)} = \begin{pmatrix} \partial\tau/\partial t & \partial\tau/\partial x & \partial\tau/\partial y \\ \partial\xi/\partial t & \partial\xi/\partial x & \partial\xi/\partial y \\ \partial\eta/\partial t & \partial\eta/\partial x & \partial\eta/\partial y \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ u & U & X \\ v & V & Y \end{pmatrix}^{-1} \tag{13}$$

$$= \frac{\rho}{K} \begin{pmatrix} \frac{K}{\rho} & 0 & 0 \\ J_{21} & Y & -X \\ J_{31} & -V & U \end{pmatrix},$$

where

$$J_{21} = -\begin{vmatrix} u & X \\ v & Y \end{vmatrix}, \qquad J_{31} = -\begin{vmatrix} u & U \\ v & V \end{vmatrix}.$$

Equation (13) provides all the partial derivatives $\frac{\partial}{\partial t}$, $\frac{\partial}{\partial x}$, and $\frac{\partial}{\partial y}$ that are needed in converting the Eulerian conservation form (5) into a new Lagrangian conservation form with (τ, ξ, η) as the independent variables:

$$\frac{\partial}{\partial t} = \frac{\partial}{\partial \tau} + \frac{\rho}{K} J_{21} \frac{\partial}{\partial \xi} + \frac{\rho}{K} J_{31} \frac{\partial}{\partial \eta},$$

$$\frac{\partial}{\partial x} = \frac{\rho}{K} Y \frac{\partial}{\partial \xi} - \frac{\rho}{K} V \frac{\partial}{\partial \eta},$$

$$\frac{\partial}{\partial y} = -\frac{\rho}{K} X \frac{\partial}{\partial \xi} + \frac{\rho}{K} U \frac{\partial}{\partial \eta}.$$

Now, corresponding to (3), it can easily be confirmed that the material derivatives of ξ and η vanish,

$$\frac{D\xi}{Dt} = \frac{D\eta}{Dt} = 0,$$

and that ξ and η are truly material functions or path functions.

After some algerbraic manipulation, (5) is transformed into a new Lagrangian conservation form with independent variables (τ, ξ, η)

$$\frac{\partial \mathbf{E_l}}{\partial \tau} + \frac{\partial \mathbf{F_l}}{\partial \xi} + \frac{\partial \mathbf{G_l}}{\partial \eta} = 0, \tag{14}$$

where the subscript l stands for Lagrangian form and

$$\mathbf{E_l} = \begin{pmatrix} K \\ Ku \\ Kv \\ Ke \end{pmatrix}, \qquad \mathbf{F_l} = \begin{pmatrix} 0 \\ Yp \\ -Xp \\ -J_{21}p \end{pmatrix}, \qquad \mathbf{G_l} = \begin{pmatrix} 0 \\ -Vp \\ Up \\ -J_{31}p \end{pmatrix}.$$

Moreover, from (7), we have the following compatibility conditions between the τ-derivatives, ξ-derivatives, and η-derivatives of $\mathbf{r}$:

$$\frac{\partial \mathbf{T}}{\partial \tau} = \frac{\partial \mathbf{V}}{\partial \xi}, \qquad \frac{\partial \mathbf{S}}{\partial \tau} = \frac{\partial \mathbf{V}}{\partial \eta}. \tag{15}$$

Equation (15) represents the geometrical deformation of the computational cells (fluid particles) and is hence called the geometrical conservation law. It provides the exact relations between flow physics (velocity) and geometry and marks the major difference between the new Lagrangian approach and a conventional one. A conventional Lagrangian approach (such as ALE) employs interpolation to march forward and to achieve grid deformation. Such relations between flow velocity and cell geometry, of course, exist but are not required explicitly.

By combining the Lagrangian physical conservation laws (14) and the geometrical conservation laws (15), we achieve a complete set of the new Lagrangian conservation form (for brevity the subscript l is dropped off):

$$\frac{\partial \mathbf{E}}{\partial \tau} + \frac{\partial \mathbf{F}}{\partial \xi} + \frac{\partial \mathbf{G}}{\partial \eta} = 0, \tag{16}$$

where

$$\mathbf{E} = \begin{pmatrix} e_1 \\ e_2 \\ e_3 \\ e_4 \\ e_5 \\ e_6 \\ e_7 \\ e_8 \end{pmatrix} \equiv \begin{pmatrix} K \\ Ku \\ Kv \\ Ke \\ U \\ V \\ X \\ Y \end{pmatrix}, \quad \mathbf{F} = \begin{pmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ f_6 \\ f_7 \\ f_8 \end{pmatrix} \equiv \begin{pmatrix} 0 \\ Yp \\ -Xp \\ -J_{21}p \\ -u \\ -v \\ 0 \\ 0 \end{pmatrix}, \quad \mathbf{G} = \begin{pmatrix} g_1 \\ g_2 \\ g_3 \\ g_4 \\ g_5 \\ g_6 \\ g_7 \\ g_8 \end{pmatrix} \equiv \begin{pmatrix} 0 \\ -Vp \\ Up \\ -J_{31}p \\ 0 \\ 0 \\ -u \\ -v \end{pmatrix}.$$

We note here that in the conventional Lagrangian formulation, including ALE, the Euler equations of gas dynamics have never been written in conservation form (divergence form), except in the special case of one-dimensional (1-D) flow.

At first glance, the Lagrangian equation system (16) seems to have four more equations to solve than (5). A closer investigation shows that there is no need to solve the first equation, it only states that mass flux K is constant all the time along a pathline. Only seven equations need to be solved. The last four equations stand for a dynamic flow-adapting grid generation-deformation procedure. Their counterpart in an Eulerian approach (such as the finite volume method) is the grid generation in the presence of general solid bodies and walls. The solution turns out to be a straight forward step in the numerical approach as described in the next section and costs little overhead (generally, less than 5%) in the CPU time consumption.

Regarding the hyperbolicity of the system (16), after careful investigation, it turns out that the system (16) is only *weakly* hyperbolic [22], in contrast to the hyperbolicity of the original Euler equations (5). In other words, in system (16), all 8 eigenvalues are real but only 7 (instead of 8) linearly independent eigenvectors can be found. Let α and β be the directional cosines of any directions in the $\xi-\eta$ plane, $\alpha^2+\beta^2=1$, and define $\alpha'=\alpha\xi_x+\beta\xi_y$, $\beta'=\alpha\eta_x+\beta\eta_y$, where, from (13), $\xi_x=Y/|J|$, $\xi_y=-X/|J|$, $\eta_x=-V/|J|$, $\eta_y=U/|J|$. Then, the eigenvalues of the system (16) are $\lambda_1=0$, $\lambda_\pm=\pm a(\alpha'^2+\beta'^2)^{1/2}$, where λ_1 has a multiplicity of 6 and a is the local speed of sound. With these eigenvalues, from (13), the local CFL number ν for marching in the computational space is

$$\nu=\frac{(\gamma p\rho)^{1/2}}{K}\max\left[\frac{(V^2+Y^2)^{1/2}}{\Delta\xi},\frac{(U^2+X^2)^{1/2}}{\Delta\eta}\right]\Delta\tau\le 1,$$

when the flow is continuous. In the presence of discontinuity, the above eigenvalue analysis is no longer valid, local shock speeds from the Riemann solvers are required instead, and the expression for a local CFL number becomes very complicated.

3. Numerical approach. The Lagrangian conservation form derived in the above section provides a formulation for applying various numerical methods in the flow computations. In principle, all the existing numerical methods that are appropriate for the Eulerian formulation are appropriate for the new Lagrangian one as well. We shall choose to give brief descriptions to the Godunov/TVD schemes, since it is one of the most popular schemes in gas dynamics computation and is easy to apply.

The space-time physical domain and computational domain in the $\tau-\xi-\eta$ space are respectively illustrated in Figures 1(a) and 1(b). A physical cell in the $x-y$ plane marching along its pathline corresponds to a rectangle cell in the $\xi-\eta$ plane marching in the τ direction in the computational space $\tau-\xi-\eta$. The superscript k refers to the marching time step number and the subscripts i and j refer to the cell index number on a time plane $\tau=$ const. The time step $\Delta\tau^k=\tau^{k+1}-\tau^k$ is uniform for all i and j but is always chosen to satisfy the CFL stability condition. The mesh divides the computational domain into cuboid control volumes or cells which in ξ and η directions are centered at $(\tau^k,\,\xi_i,\,\eta_j)$ and have heights $\Delta\xi_i=\xi_{i+1/2}-\xi_{i-1/2}$ and $\Delta\eta_j=\eta_{j+1/2}-\eta_{j-1/2}$ (for all k). Unless otherwise stated we shall use uniform cell width $\Delta\xi_i$ for all i and $\Delta\eta_j$ for all j (Figure 1(c)).

In the physical space (t,x,y) a cuboid cell marching in (τ,ξ,η) space corresponds to a fluid particle marching along its path tube with step Δt ($\Delta t=\Delta\tau$). The fluid particle is bounded by four path surfaces $\xi=\xi_{i\pm1/2}$ and $\eta=\eta_{j\pm1/2}$ around it. Initially, any curvilinear coordinate mesh on the $x-y$ plane surface may be used as

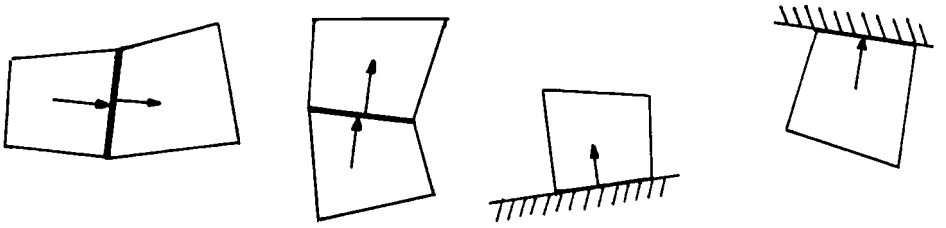


FIG. 2. (a) *Left and right states for* 1*-D local Riemann solver;* (b) *right or left state for a local boundary Riemann solver.*

the $\xi - \eta$ coordinate mesh and the initial $\mathbf{T}$ and $\mathbf{S}$ can be determined from (7) as part of the initial condition. A solid wall is always a path surface (material surface) and hence a coordinate surface.

The Godunov scheme [15, 16] for (16) is derived by applying the divergence theorem to the cuboid cell (i, j, k). The result is

$$\mathbf{E}_{i,j}^{k+1} = \mathbf{E}_{i,j}^{k} - \frac{\Delta\tau^k}{\Delta\xi_i}\,(\mathbf{F}_{i+1/2,j}^{k+1/2} - \mathbf{F}_{i-1/2,j}^{k+1/2}) - \frac{\Delta\tau^k}{\Delta\eta_j}\,(\mathbf{G}_{i,j+1/2}^{k+1/2} - \mathbf{G}_{i,j-1/2}^{k+1/2}), \tag{17}$$

$$i = 1, 2, \ldots, m, \qquad j = 1, 2, \ldots, n,$$

where the notation for the cell average of any quantity f is

$$f_{i,j}^{k} = \frac{1}{\Delta\xi_i \Delta\eta_j} \int_{\xi_{i-1/2}}^{\xi_{i+1/2}} \int_{\eta_{j-1/2}}^{\eta_{j+1/2}} f(\tau^k, \xi, \eta)\, d\xi\, d\eta \tag{18}$$

and the notation for time τ average of f is

(19)

$$f_{i+1/2,j}^{k+1/2} = \frac{1}{\Delta\tau^k} \int_{\tau^k}^{\tau^{k+1}} f(\tau, \xi_{i+1/2}, \eta_j)\, d\tau, \qquad f_{i,j+1/2}^{k+1/2} = \frac{1}{\Delta\tau^k} \int_{\tau^k}^{\tau^{k+1}} f(\tau, \xi_i, \eta_{j+1/2})\, d\tau.$$

In (17), ideally, the cell-interface fluxes $\mathbf{F}_{i+1/2,j}^{k+1/2}$ and $\mathbf{G}_{i,j+1/2}^{k+1/2}$ for the cell (i, j) would be obtained from the self-similar solution of a local 2-D Riemann problem formed by the average constant state $\mathbf{Q}_{i,j} = (u, v, p, \rho)_{i,j}^T$ of the cell (i, j) and those of its adjacent cells. Unfortunately, due to its complexity, the exact solution to a general 2-D Riemann problem is not yet available. On the other hand, it is known that a monotone difference scheme to a general conservation form converges to the physically relevant entropy-satisfying solution. In particular, Crandall and Majda [17] establish the rigorous convergence for dimensional splitting algorithms when each step is approximated by a monotone difference scheme (such as the Godunov scheme) for a single conservation law of multidimension. We shall extend and apply the dimensional splitting in the Godunov scheme for the hyperbolic system of conservation laws (16).

When applying the dimension-splitting technique in the Godunov scheme (17), one only needs to solve a regular 1-D Riemann problem formed by two adjacent constant states, say, $\mathbf{Q}_{i,j}$ and $\mathbf{Q}_{i+1,j}$ in the normal direction of the interface (Figure 2(a)). For a high resolution TVD scheme, nonlinear interpolation is employed across the interface line to reconstruct the flow variables at both sides of the interface before applying the local 1-D Riemann solver. At a solid wall, a boundary Riemann problem

is solved, which consists of Riemann data that are a mirror image with respect to the wall (Figure 2(b)).

The numerical procedure of the Godunov/TVD scheme can now be summarized as follows:

(i) Initiation. Assume the initial conditions of a flow problem are given at $t = 0$ ($\tau = 0$) in the $x - y$ plane. Then an appropriate $\xi - \eta$ coordinate mesh is laid on the $x-y$ plane (for instance, we take ξ and η equal to the arc length of their corresponding coordinate lines on $x - y$ plane), with $\xi = \xi_0, \xi_1, \xi_2, \ldots, \xi_m$, $\eta = \eta_0, \eta_1, \eta_2, \ldots, \eta_n$, and the curve $\xi = \xi_0$ (or $\eta = \eta_0$) coinciding with the body surface if there is a solid wall (Figure 1(a)). Hence $\mathbf{T}^0$, $\mathbf{S}^0$ as well as the flow variable $\mathbf{Q}^0 = (u^0, v^0, p^0, \rho^0)^T$ are known on $x - y$ plane as initial conditions. Subsequently, $\mathbf{E}^0_{i,j}$, $i = 1, 2, \ldots, m$, $j = 1, 2, \ldots, n$, are available. For example, if we choose ξ, η to be the respective arc lengths of x- and y-coordinate lines then, from (7), $\mathbf{T}^0 = (1, 0)^T$, $\mathbf{S}^0 = (0, 1)^T$, and $\mathbf{E}^0$ follows from its expressions in (16).

(ii) Evaluate cell interface fluxes, and update e_5, e_6, e_7, e_8. With all $\mathbf{E}^k_{i,j}$ and $\mathbf{Q}^k_{i,j}$ known at step k ($k = 0, 1, 2, \ldots$), the $\mathbf{Q}$ data is first upgraded or reconstructed via a monotonic upstream-centered scheme for conservation laws (MUSCL) TVD procedure by nonlinear interpolation. This upgrading is performed in a dimension-by-dimension way. For example, in the ξ direction, let f be any of the above physical variables u, v, p, or ρ, then, instead of assuming a uniform state in the cells (i, j) and $(i + 1, j)$, we assume linearly distributed states and use linear extrapolation to determine cell interface flow variables $f_r = f_{i+1,j} - .5(f_{i+2,j} - f_{i+1,j})\phi(r^+)$ with $r^+ = (f_{i+1,j} - f_{i,j})/(f_{i+2,j} - f_{i+1,j})$ and $f_l = f_{i,j} + .5(f_{i,j} - f_{i-1,j})\phi(r^-)$ with $r^- = (f_{i+1,j} - f_{i,j})/(f_{i,j} - f_{i-1,j})$, where $\phi(r) = \max(0, \min(1, r))$ is the minmod flux limiter and subscripts r and l correspond to right and left states, respectively. They are then projected on the normal direction of the cell interface (Figure 2(a)) and used as the input to the the local Riemann solvers. To obtain the normal direction of an interface, say, between cells $(i + 1, j)$ and (i, j), the directional vector of this interface is first obtained by taking the average of the unit vectors of $\mathbf{S}_{i+1,j}$ and $\mathbf{S}_{i,j}$ and its normal vector follows in a straightforward way. At a solid wall, of course, its normal direction is chosen as the normal to the wall. New cell interface flow variables $\mathbf{V}^{k+1/2}_{i\pm1/2,j}$ or $\mathbf{V}^{k+1/2}_{i,j\pm1/2}$ and $p^{k+1/2}_{i\pm1/2,j}$ or $p^{k+1/2}_{i,j\pm1/2}$ are obtained as the results of these Riemann solvers. Then fluxes $\mathbf{F}$ and $\mathbf{G}$ are calculated from their expressions in (16). In order to do so, we first update $\mathbf{T}^k_{i,j}$ and $\mathbf{S}^k_{i,j}$ to $\mathbf{T}^{k+1}_{i,j}$ and $\mathbf{S}^{k+1}_{i,j}$:

$$\begin{pmatrix} U^{k+1}_{i,j} \\ V^{k+1}_{i,j} \end{pmatrix} = \mathbf{T}^{k+1}_{i,j} = \mathbf{T}^k_{i,j} + \frac{\Delta\tau^k}{\Delta\xi_i}\,(\mathbf{V}^{k+1/2}_{i+1/2,j} - \mathbf{V}^{k+1/2}_{i-1/2,j}), \tag{20}$$

$$\begin{pmatrix} X^{k+1}_{i,j} \\ Y^{k+1}_{i,j} \end{pmatrix} = \mathbf{S}^{k+1}_{i,j} = \mathbf{S}^k_{i,j} + \frac{\Delta\tau^k}{\Delta\eta_j}\,(\mathbf{V}^{k+1/2}_{i,j+1/2} - \mathbf{V}^{k+1/2}_{i,j-1/2}). \tag{21}$$

Thus the updating of U, V, X, and Y (i.e., e_5, e_6, e_7, e_8) is completed. With these newly updated U, V, X, Y and the pressures $p^{k+1/2}_{i\pm1/2,j}$ or $p^{k+1/2}_{i,j\pm1/2}$ from the local Riemann solvers, flux components f_2, f_3, f_4 and g_2, g_3, g_4 are readily available according to their expressions in (16). For example, for the cell (i, j), f_4, the fourth component of $\mathbf{F}$ at the interface between cells $(i + 1, j)$ and (i, j), is evaluated as

$$-p^{k+1/2}_{i+1/2,j} \begin{vmatrix} u^{k+1/2}_{i+1/2,j} & Y^{k+1}_{i,j} \\ v^{k+1/2}_{i+1/2,j} & X^{k+1}_{i,j} \end{vmatrix}.$$

(iii) Use (17) to update e_2, e_3, e_4 and advance by one step (e_1 requires no updating).

(iv) Decode $\mathbf{E}_{i,j}^{k+1}$ to get $\mathbf{Q}_{i,j}^{k+1}$. For simplicity, all the i, j, k superscripts and subscripts are dropped. From (12) and the expression of $\mathbf{E}$ in (16),

$$\rho = \frac{K}{(e_5 e_8 - e_6 e_7)}, \qquad u = \frac{e_2}{K}, \qquad v = \frac{e_3}{K}, \qquad p = (\gamma - 1)\rho \left[\frac{e_4}{K} - .5(u^2 + v^2)\right].$$

Now the numerical procedure for advancing one time step is completed. To march forward further, one goes back to (ii) and repeats (ii)–(iv). The loop from (ii) to (iv) is self-sustained when marching in the $\tau - \xi - \eta$ computational space. However, if a grid in the physical $x - y$ plane is required for plotting a curve or contours, one may

(v) Generate grid points (coordinates of cell centers) along pathlines

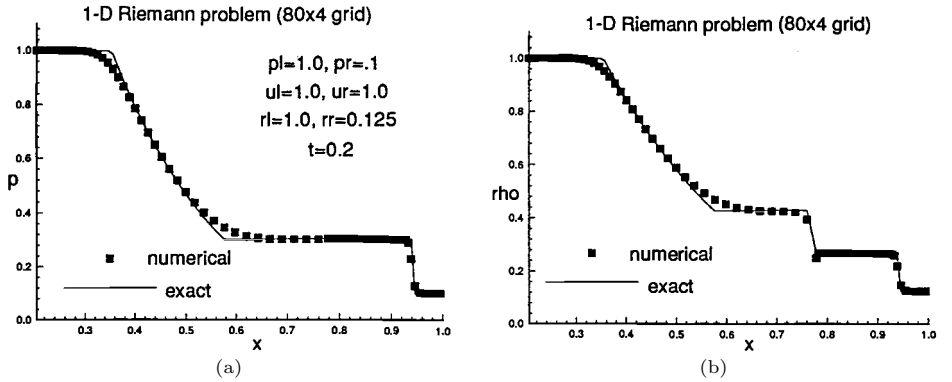
$$x_{i,j}^{k+1} = x_{i,j}^{k} + \frac{1}{2}\Delta\tau^k (u_{i,j}^{k} + u_{i,j}^{k+1}), \qquad y_{i,j}^{k+1} = y_{i,j}^{k} + \frac{1}{2}\Delta\tau^k (v_{i,j}^{k} + v_{i,j}^{k+1}).$$

Each grid point represents the center of a fluid particle (a computational cell). Note that unlike the conventional Lagrangian approach (e.g., ALE), these new x, y coordinates are not fed back to, as they are not used in, the computational process. Again, unlike any conventional Lagrangian methods using staggered grids, our approach does not need a staggered grid and the information of the positions of cell corners is never required.

An interesting feature of the time-dependent Lagrangian approach is that the entire physical domain of computation may move along with the main flow and keep deforming its shape. It plays a role as a moving viewing window. One can always adjust the viewing window so that it becomes relatively motionless. For this purpose, only a few more lines in the Fortran code are needed. For instance, assume the main flow directs from left to right, we wait until the leftmost column of cells moves over a distance of about the average cell size, then a new column of cells is injected as the new leftmost column. To avoid increasing the number of cells, the rightmost column of cells, which is exiting the viewing window, is removed. In section 4, this technique is applied to examples 2 and 3.

4. Typical numerical results. In this section, the new Lagrangian approach is tested for accuracy and robustness on several numerical examples. $\gamma = 1.4$ is used for all the cases. The numerical results are then compared to exact solutions or experimental results whenever they are available, or to the numerical results by other Eulerian approaches. Example 4 is also chosen for convergence test or validation. In all the examples, a modest grid is used to demonstrate that the Lagrangian approach is capable of catching all the flow features even with a coarse grid, except when convergence test is carried out.

The first example is indeed a 1-D Riemann problem but is numerically solved as a 2-D problem. The initial data is chosen from the well-known Sod's shock tube problem, namely, $(u_l, v_l, p_l, \rho_l) = (1.0, 0, 1.0, 1.0)$, $(u_r, v_r, p_r, \rho_r) = (1.0, 0, 0.1, 0.125)$, where the subscripts l and r denote left and right states, respectively. A grid of 80×4 with $\Delta x = \Delta y = 0.01$ is employed in the computation. Since the flow is uniform in y direction, only 4 grid points are used in this direction. The computed pressure p and density ρ at time $t = 0.2$ are shown in Figure 3, where the corresponding exact solutions are also plotted for comparison. It is observed that our numerical results agree well with the exact solutions. While a shock is resolved in about two points, the contact discontinuity remains almost absolutely sharp, as claimed in section 1. Note



(a) (b)

FIG. 3. *Comparison of numerical solutions and exact solutions for a* 1*-D Riemann problem,* (a) *pressure and* (b) *density distributions along* x *direction.*

that in the density plot, there is a little dip-down (glitch) after the contact, which is a common phenomenon for Lagrangian approach (see, e.g., [23]).

The next example is a genuine 2-D problem—a steady supersonic flow of Mach numer 2.9 past a ramp channel. The steady state is achieved by time marching until the flow variables do not change with time. At the bottom, a ramp of 10.94° locates at $x = 0.5$ (Figure 4), while the top wall is a horizontal straight line. The height of the physical domain is 1.0 at the inlet, and initially, the domain length is chosen to be 3.0. As mentioned in section 3, we apply the motionless viewing window technique, the size and shape of the window change slightly with time-marching. A grid of 90×40 with $\Delta x = 0.032$ and $\Delta y = 0.025$ is employed. A $\Delta t = .001$ is chosen as the marching time step size. For this oblique shock and shock reflection problem, there exists a steady state exact solution. The free stream state is given as $(u, v, p, \rho) = (2.9, 0, 1/1.4, 1.0)$; the state downstream of the oblique shock is evaluated to be $(u, v, p, \rho) = (2.6193, 0.50632, 1.5282, 1.7)$, with a shock angle of 29°; while the state after shock reflection is $(u, v, p, \rho) = (2.4015, 0, 2.934, 2.6872)$. Since the flow is supersonic, all the flow variables are specified at the inlet as boundary conditions. At the outlet, boundary condition is simply the extrapolation condition.

In Figure 4(a) the Lagrangian grid is seen deformed with the flow and becomes condensed when the fluid is compressed. In Figure 4(b) the isobars and the shock reflection is clearly displayed. Figures 4(c) and 4(d) show, respectively, pressure plots along the bottom wall and the top wall; along with the corresponding exact solutions. The numerical solutions agree well with the exact ones. Figure 4(e) displays the density distribution along y direction at the domain exit. The plot is seen in good agreement with the exact solution except there are "glitches" at the top and bottom walls, since solid walls represent contacts.

In the third example, we consider a supersonic flow with a subsonic pocket—the problem of supersonic flow past a channel with a ramp segment. The total length of the physical domain is 3.3 with height at inlet 1.0. As shown in Figure 5, a ramp of 15° is located at the bottom wall between $x = 0.5$ and $x = 1$, the rest of the top and bottom walls are horizontal flat lines [18, 19]. When a Mach 1.8 flow passes through the channel, an oblique shock, a Mach stem, shear layers, and reflected shocks are generated as shown in Figure 5. Again a modest grid of 90×40 cells is used in the computation. To test if the steady state is reached, the computation was first carried to $t = 8$ and then to $t = 18$. The computed flow is found only slightly changed from $t = 8$ to $t = 18$. The contours and plots in Figures 5 and 6 are based on the

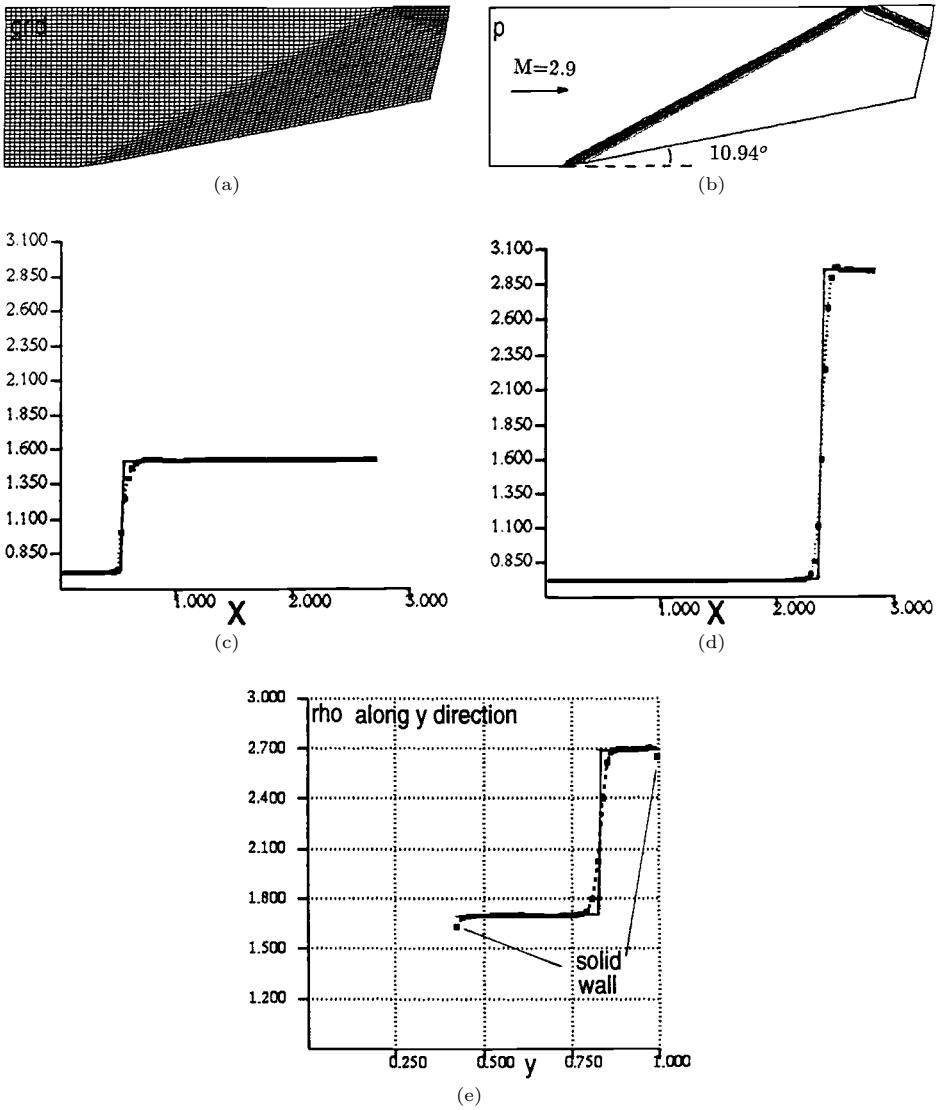


FIG. 4. *Supersonic flow of Mach* 2.9 *past a ramp channel. The ramp angle is* 10.94°. (a) *The deformed Lagrangian grid (when flow becomes steady),* (b) *isobars (*40 *levels between* max. *and* min. *pressure values),* (c) *pressure distribution along bottom wall,* (d) *pressure distribution along top wall, and* (e) *density distribution at domain exit; solid lines denote the exact solutions. A* 90×40 *grid is used.*

numerical results at $t = 18$. In Figure 5, the grid formed according to the flow, the isobars and the isomachs are illustrated. A notch at the upper right corner of the physical domain is due to the slow-down of the flow downstream of the Mach stem, where the flow becomes subsonic with a Mach number as low as 0.65. Although the grid is relatively crude, all the flow features are clearly displayed: the Mach stem is about 20% of the domain height, the oblique and reflected shocks, and particularly the shear layers stemming out from the triple point. The quality of the shear layers is seen comparable to that of a finer grid in Eulerian formulation (see, e.g., [18, 19]). In Figure 6, pressure and Mach number distributions along the top and the bottom

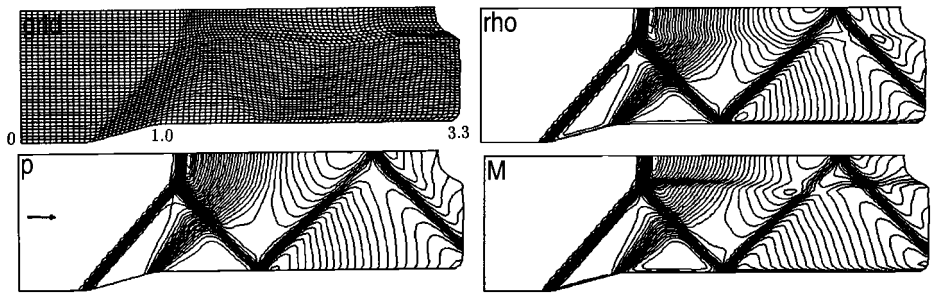


FIG. 5. *The grid, isobars, isopycnics, and isomachs for a* $M = 1.8$ *flow past a* 15° *ramp channel at* $t = 18$. *The Mach stem is about* 20% *of the max. domain height. Also note the shear layer issued from the triple point.*

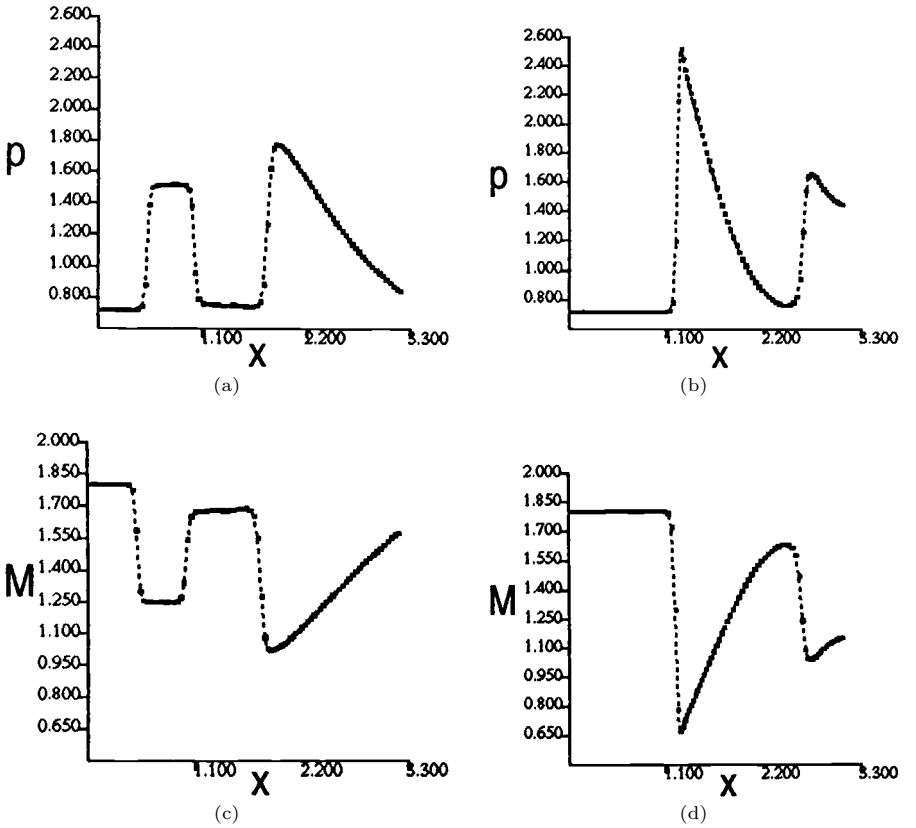


FIG. 6. (a) *and* (b), *pressure distributions along bottom and top solid walls;* (c) *and* (d), *Mach number distributions along bottom and top solid walls.*

walls are plotted. When passing across a shock, these plots show crisp resolutions of 2 to 3 points. They agree well with the results by the Eulerian method (e.g., [19]).

The fourth example is the Mach reflection of a shock wave from a wedge [20]. A vertical plane shock of Mach 1.3 travels over a wedge of 25° and generates Mach reflection as shown in Figure 7. The isobars, isomachs, and grids at $t = 1.2$ are displayed. They agree well with the experimental result [20]. In particular, the

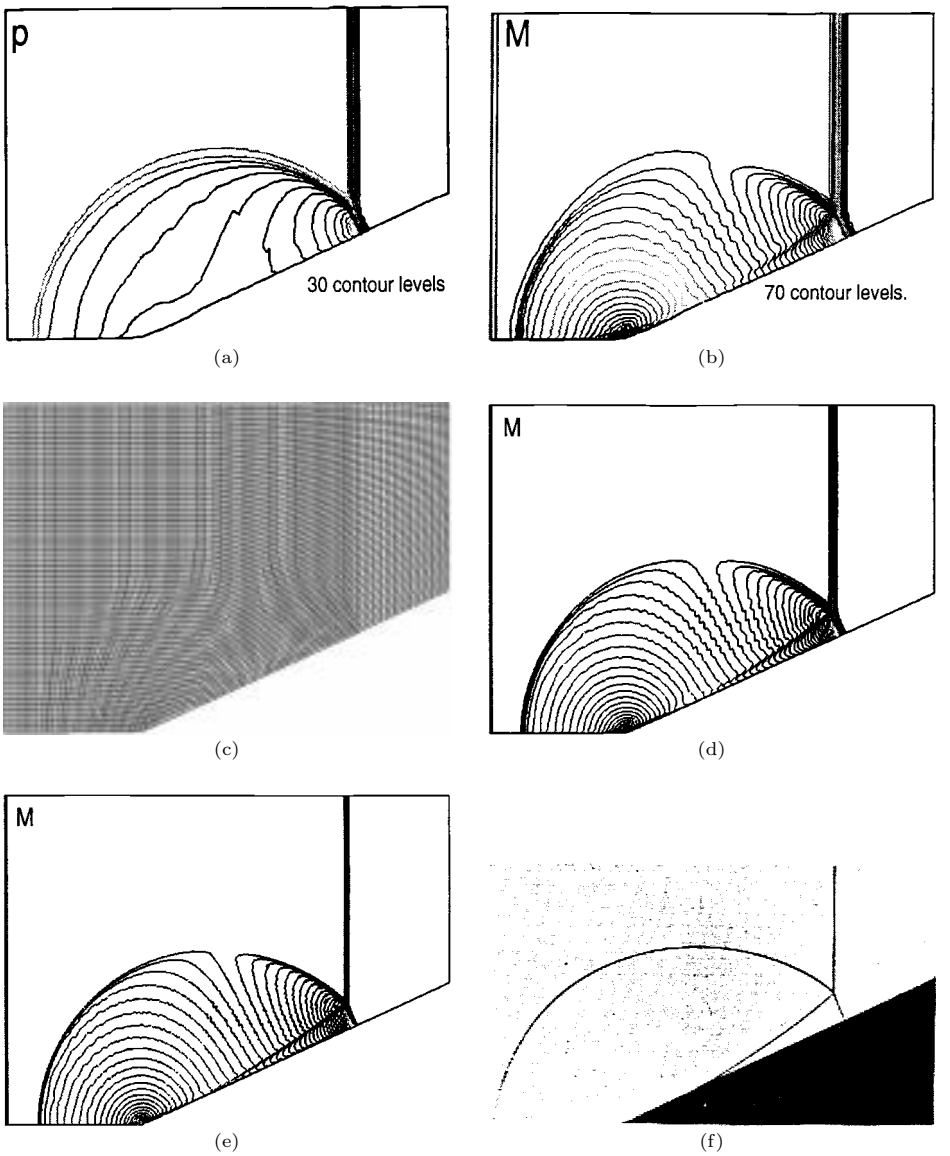
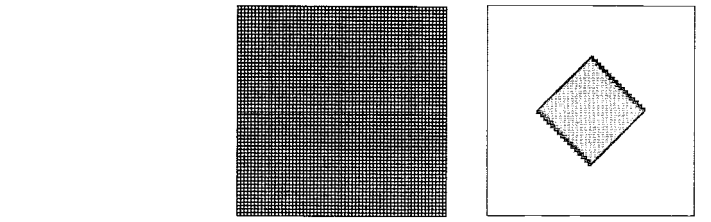
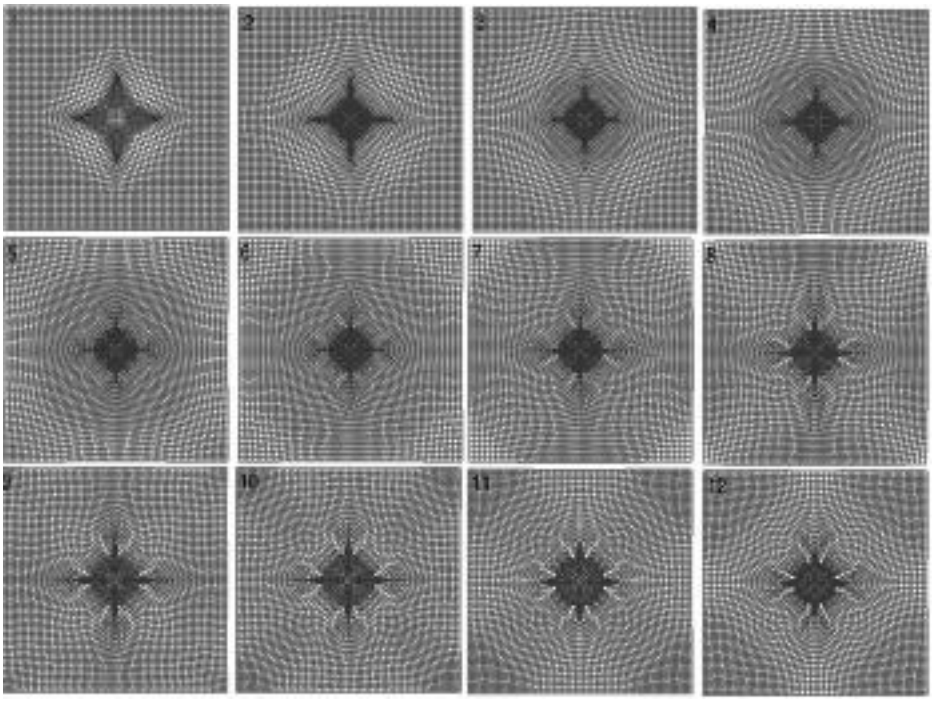


FIG. 7. *An $M = 1.3$ plane shock past a wedge of 25° at $t = 1.2$.* (a) *Isobars,* (b) *isomachs,* (c) *grid* (150×60), (d) *isomachs with double grid* (300×120), (e) *isomachs with quadruple grid* (600×240), (f) *experimental shadowgraph* [20].

contactline (shear layers) stemming from the triple point is clearly seen from the isomach contours. In this case, a grid of 150×60 is employed with $\Delta x = \Delta y = 0.01$, $\Delta t = 0.002$. The initial state is $\mathbf{Q^0} = (p^0, u^0, v^0, \rho^0)^T = (1/1.4, 0, 0, 1)^T$ and the flow state at the left end is $\mathbf{Q}_i = (1.2893, 0.44231, 0, 1.5157)^T$. We also run the problem with a given CFL number ν according to the expression at the end of section 2, instead of a fixed Δt. Although the numerical process does not blow up for ν up to 0.9, the output isomachs are noisy with wiggles. The results are improved and clean when ν is reduced to 0.5 or less. At $\nu = 0.5$, 360 time steps are marched to reach $t = 1.2$ and CPU time is considerably saved as compared to the case with fixed $\Delta t = 0.002$.



(a)



(b)

FIG. 8. (a) *Initial grid* (60×60) *and initial conditions for the implosion/explosion problem.* (b) *Lagrangian grids at different time steps,* 1–12 *correspond to* $t = 0.05$ *to* $t = 0.06$ *with a uniform increment of* $\Delta t = 0.05$.

The present example is also chosen to go through a convergence test (validation) for the new Lagrangian approach. For this purpose, double grid (300×120) and quadruple grid (600×240) are used and the same code is run with appropriate half and quarter time step sizes, respectively, to $t = 1.2$. Figures 7(d) and 7(e) demonstrate the corresponding isomachs of the two runs, they look almost identical except that the denser grid yields a sharper picture. In addition, they all agree well with the isomachs in Figure 7(b) and the convergence of the numerical approach is justified.

The last example is an interesting implosion/explosion problem. A numerical study of this problem has been carried out in [21]. A typical implosion/explosion problem is depicted in Figure 8(a). Consider a square duct of infinite length. Inside the duct, initially separated by a diaphragm, an inner square contains a gas of different state from the outer one. The centers of these two squares coincide (Figure 8(a)). At $t = 0$, the diaphragm is ruptured, and the inner and outer gases begin to interact with

each other. The implosion/explosion problem may be regarded as a 2-D version of the shock tube problem with four surrounding walls. However, due to higher dimension, the flow pattern is much more complicated. In our test, we choose the inner state $\mathbf{Q}_i$ and outer state $\mathbf{Q}_o$ as follows: $(u_i, v_i, p_i, \rho_i) = (0, 0, 0.14, 0.125)$, $(u_o, v_o, p_o, \rho_o) = (0, 0, 1.0, 1.0)$. Initially, a uniform grid of 60×60 is laid as shown in Figure 8(a) with grid size $\Delta x = \Delta y = 0.01$. Two computation runs are carried out to $t = 0.6$. The first one runs with a fixed CFL number $\nu = 0.7$ and it takes 1060 time steps to reach $t = 0.6$. The second one runs with a fixed time step $\Delta t = 0.0005$, taking only 600 steps and less CPU time. In this case, it is more advantageous to march with a constant time step size. Figure 8(b) illustrates how the grid is deformed with the flow from $t = 0.05$ to $t = 0.6$ with an increment of $\Delta = 0.05$. Soon after time $t = 0.6$ (step 12) the computation stops due to severe cell deformation and the consequent gridline crossing. An indication of the failure is that the determinant of transformation $|J|$ in (11) passes through 0 and becomes negative, rendering the transformation invalid. Figure 9(a) and 9(b) display, respectively, the isobars and isopycnics at different time steps. Steps 1–12 correspond to the same steps in Figure 8(b). Since the pressure of $\mathbf{Q}_i$ is lower at the beginning, a shock appears in the inner region. It propagates and contracts to the center. At the four corners of the shock, triple points and embedded shocks are generated as observed from step 1, Figure 9(a). Later on, very complicated flow patterns are developed. From Figure 9(b), it is seen that the contact stays in the center area but keeps changing its shape and reducing its encircled area slowly and steadily. The contact can be located by comparing the contours at the same step in Figures 9(a) and 9(b). The Mach numbers in the domain range from 0 to about 0.3. The present problem is a typical one of low Mach number flow.

In Figure 10, pressure and density distributions along the centerline in x-direction are plotted. In general, shocks are resolved in 2–4 points and the contacts are almost absolutely sharp, which is a well-known major advantage of the Lagrangian approach. Qualitatively the contours in Figure 9(a) and 9(b) are comparable to those from [21], except the 2-D shocks look thicker due to the crude grid. In contrast, a grid of 359×359 points is employed in [21] to achieve high resolution. In Figure 10(b), the numerical density value dips down slightly across a contact, as other examples do in the present paper (see also, e.g., [23]).

Finally, regarding the CPU time consumption, it somewhat slightly depends on the newton iterations in the local Riemann solvers (i.e., flow complexity). Typically on a CRAY-YMP without vectorization, CPU time consumption per cell per time step is $120\mu s$. With appropriate vectorization, it is expected to be a few times faster.

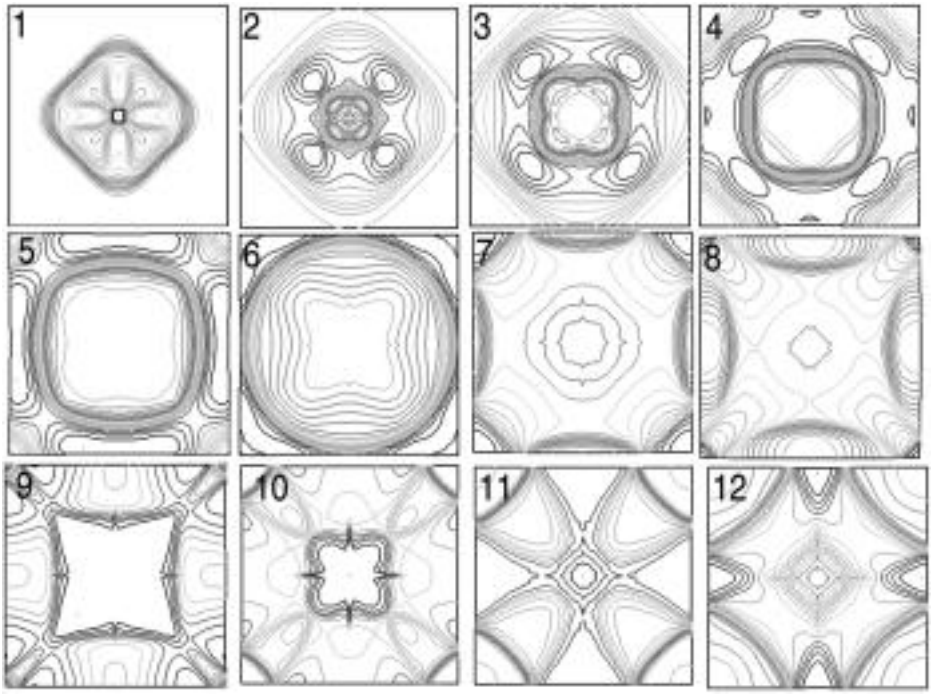
5. Concluding remarks. In this paper, with the concept of material functions (path functions), we successfully introduce the new Lagrangian formulation in conservation form for time-dependent inviscid flows and its numerical procedure based on the standard Godunov/TVD numerical schemes. Various numerical examples are presented to demonstrate the capability and robustness of the new Lagrangian approach in handling unsteady or steady, supersonic or subsonic inviscid flow computations.

Compared to the Eulerian approach, the new Lagrangian approach possesses the following features:

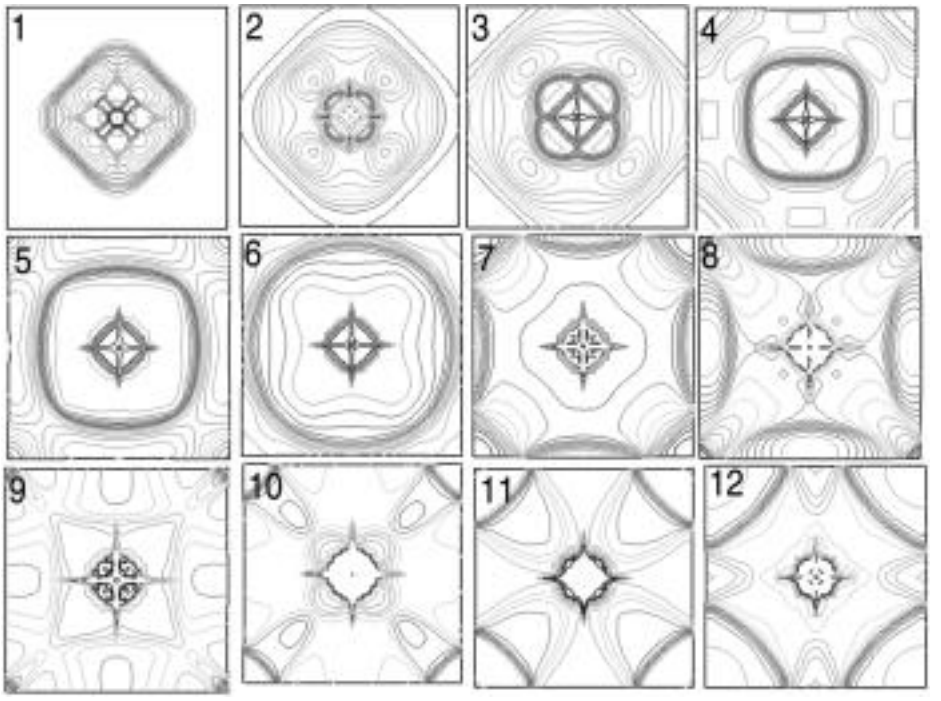
(i) Flow physics is closely followed, and contact discontinuity resolution is crisp and never becomes smeared.

(ii) Grids are automatically generated following particle pathlines. The geometrical conservation law assures the correct deformation of each grid cell.

A possible extra advantage is some enhancement of shock resolution since the

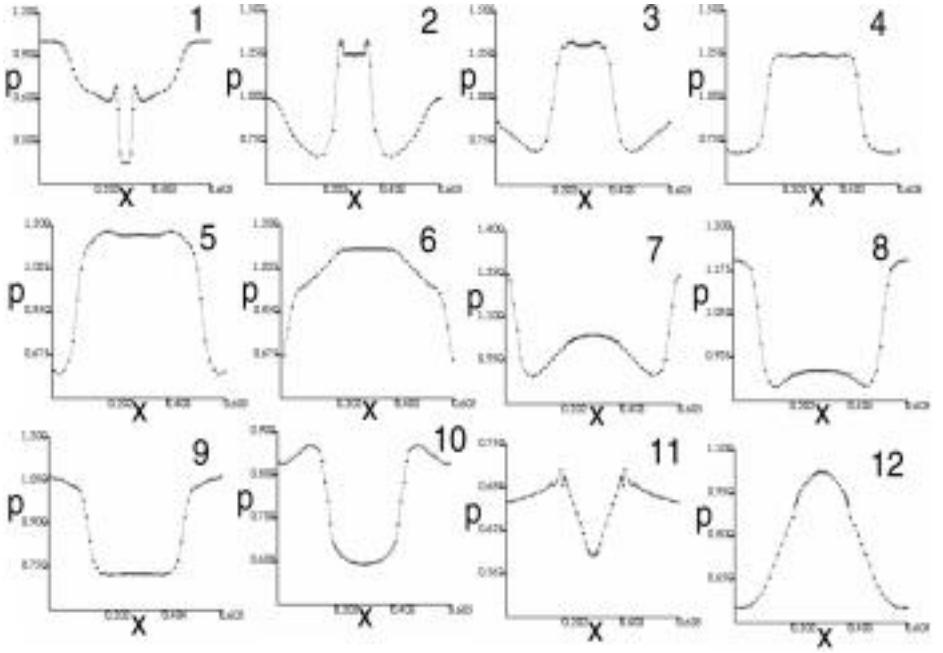


(a)

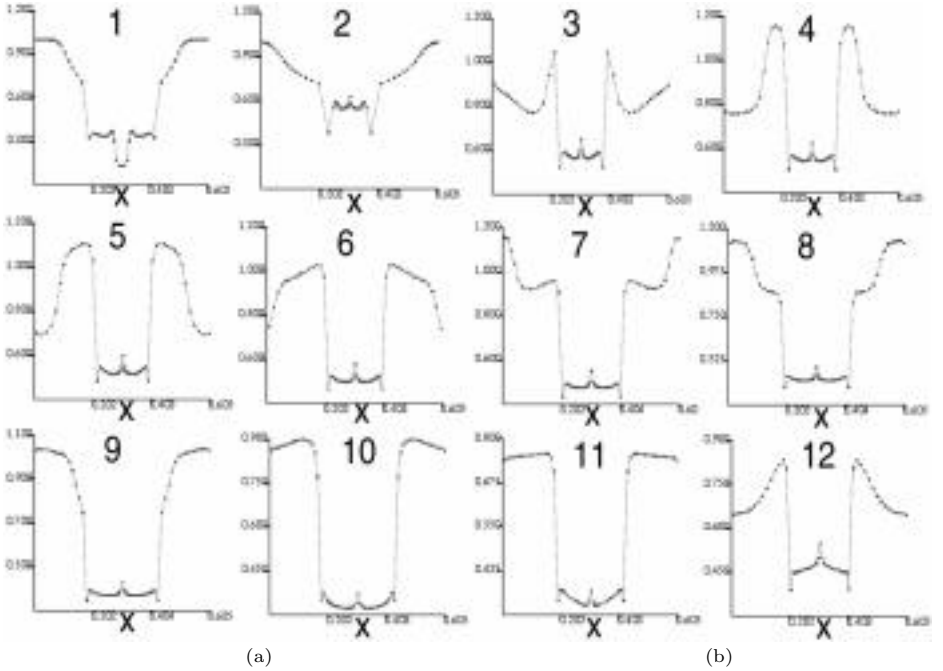


(b)

FIG. 9. (a) *Isobars,* (b) *isopycnics of the implosion/explosion problem.* 1–12 *correspond to* $t = 0.05$ *to* $t = 0.6$ *with a uniform increment of* $\Delta t = 0.05$.



(a) (b)



(a) (b)

FIG. 10. (a) *Pressure,* (b) *density along centerline in x direction, presented at every other stage,* 2–12 *correspond to the same stages as in Figure* 9. *Shocks are resolved in* 2–4 *points. Contacts are resolved almost absolutely sharp.*

Lagrangian TVD scheme changes its upwind direction automatically according to the flow, playing a role as a rotating upwind scheme.

Compared to the conventional Lagrangian approach (such as the Lagrangian phase of ALE), the new Lagrangian formulation puts the Euler equations in divergence form, making it possible to use an unsplit, nonstaggered grid. In other words, it provides an intrinsic coordinate system and a direct and exact relation between flow physics and grid geometry—the geometrical conservation laws, which allows proper analysis and easy application of modern high resolution shock capturing schemes and subsequently may yield better results.

On the other hand, it shares with any conventional Lagrangian method the same difficulty for flows where fluid particles experience servere distortion, resulting in gridline crossing and failure. In ALE [2], the gridline curdling is remedied by rezoning—a step back to the Eulerian phase. In the new Lagrangian approach, a remedy to gridline crossing is under investigation and will be reported in the future.

Acknowledgments. The authors are thankful to the reviewers for their valuable comments.

REFERENCES

[1] A. Harten, *ENO schemes with subcell resolution*, J. Comput. Phys., 83 (1989), pp. 148–184.
[2] C. W. Hirt, A. A. Amsden, and J. L. Cook, *An arbitrary Lagrangian-Eulerian computing method for all flow speeds*, J. Comput. Phys., 14 (1974), pp. 227–253.
[3] W. H. Hui and H. J. Van Roessel, *Unsteady three dimensional flow theory via material functions*, in NATO AGARD Symposium on Unsteady Aerodynamics-Fundamentals and Application to Aircraft Dynamics, Götingen, West Germany, 1985, CP-386, Paper S1.
[4] H. J. van Roessel and W. H. Hui, *A new Lagrangian formulation for steady* 3*-D hypersonic flow*, Z. Angew. Math. Phys., 40 (1989), pp. 677–710.
[5] C. S. Yih, *Stream functions in three dimensional flows*, La Houille Blanche, 12 (1957), pp. 445–450.
[6] C. Y. Loh and W. H. Hui, *A new Lagrangian method for steady supersonic flow computation part* I: *Godunov scheme*, J. Comput. Phys., 89 (1990), pp. 207–240.
[7] W. H. Hui and C. Y. Loh, *A new Lagrangian method for steady supersonic flow computation, part* II: *Slipline resolution*, J. Comput. Phys., 103 (1992), pp. 450–464.
[8] W. H. Hui and C. Y. Loh, *A new Lagrangian method for steady supersonic flow computation, part* III: *Strong shocks*, J. Comput. Phys., 103 (1992), pp. 465–471.
[9] C. Y. Loh and W. H. Hui, *A new Lagrangian random choice method for solving the steady Euler equations*, Comput. Fluid Dynamics J., 2 (1993), pp. 247–268.
[10] C. Y. LePage and W. H. Hui, *A shock adaptive Godunov scheme based on the generalized Lagrangian formulation*, J. Comput. Phys., 122 (1995), pp. 291–299.
[11] C. Y. Loh and M. S. Liou, *Lagrangian solution of supersonic real gas flow*, J. Comput. Phys., 104 (1993), pp. 150–161.
[12] C. Y. Loh and M. S. Liou, *A Lagrangian random choice approach for supersonic real gas flows*, SIAM J. Sci. Comput., 15 (1994), pp. 1038–1058.
[13] C. Y. Loh and M. S. Liou, *A new Lagrangian method for three dimensional steady supersonic flows*, J. Comput. Phys., 113 (1994), pp. 224–248.
[14] C. Y. Loh and M. S. Liou, *Three dimensional steady supersonic duct flow using Lagrangian formulation*, Shock Wave, 3 (1994), pp. 239–248.
[15] S. K. Godunov, *Difference method of numerical computations of discontinuous solutions in hydrodynamic equations*, Math. Sbornik, 47 (1959), pp. 271–306.
[16] B. Van Leer, *Towards the ultimate conservative difference scheme* V. *A second order sequel to Godunov's method*, J. Comput. Phys., 32 (1979), pp. 234–245.
[17] M. G. Crandall and A. Majda, *Monotone difference approximations for scalar conservation laws*, Math. Comp., 34 (1980), pp. 1–21.
[18] M. S. Liou, *An extended Lagrangian method*, J. Comput. Phys, 118 (1995), pp. 294–309.
[19] K. H. Kao and M. S. Liou, *Grid adaption using chimera composite overlapping scheme*, AIAA J., 32 (1994), pp. 942–949.
[20] M. Van Dyke, *An Album of Fluid Motion*, Parabolic Press, Stanford, CA, 1982, p. 143.

[21] T. Aki and F. Higashino, *A numerical study on implosion of polygonally interacting shocks and consecutive explosions in a box*, in American Institute of Physics Conference Proceedings 208, 1990, pp. 167–172.
[22] W. H. Hui, P. Y. Li, and Z. W. Li, *A unified coordinate system for solving the two-dimensional Euler equations*, J. Comput. Phys., 153 (1999), pp. 596–637.
[23] M. Ben-Artzi and J. Falcovitz, *A second-order Godunov-type scheme for compressible fluid dynamics*, J. Comput. Phys., 55 (1984), pp. 1–32.

SIAM J. SCI. COMPUT.
Vol. 22, No. 1, pp. 351–367
© 2000 Society for Industrial and Applied Mathematics

GRADIENT-PARTICLE SOLUTIONS OF FOKKER–PLANCK EQUATIONS FOR NOISY DELAY BIFURCATIONS*

R. KUSKE†

Abstract. A gradient particle method is used to compute probability densities for processes with delay bifurcations that are sensitive to very small noise. The method is particularly useful for computing the probability density in the transition region, where asymptotic approximations may not be valid. For a one-dimensional steady bifurcation problem and a two-dimensional FitzHugh–Nagumo model, we solve the Fokker–Planck equation (FPE) and compare the results with direct simulations and asymptotic approximations.

Key words. particle method, Fokker–Planck, probability density, delay bifurcation, FitzHugh–Nagumo

AMS subject classifications. 65P30, 60G51, 37H20, 35K15

PII. S1064827599350332

1. Introduction.

Certain dynamical systems are very sensitive to noise, such as models of chemical kinetics [1], [2], neuron firing [2], [3], chaotic wave mode interactions [4], and lasers [5]. The deterministic dynamics of these systems, described by systems of ordinary differential equations (ODEs), can change substantially when very small noise is introduced. The probability density for the process describes the sensitivity of the process to noise. It can be used to find a variety of properties of the stochastic process, including time and location of state transitions and moments. Even though the probability density function holds all of the information for the stochastic process, much of the previous work on noisy nonlinear dynamics does not attempt to compute or approximate it. This is because it can be difficult to find the probability density when the underlying dynamical system is complicated and the noise is small.

In general, the goal is to find $p(\mathbf{x}, t)$, the probability that the process takes the value $\mathbf{x}$ at a given time t. Given a system of stochastic differential equations (SDEs) describing the process,

$$d\mathbf{x} = \mathbf{a}(\mathbf{x}, \mathbf{t})dt + \sqrt{2}\epsilon d\mathbf{W} \quad (\epsilon \text{ is a constant}), \tag{1.1}$$

with $\mathbf{W}$ a vector of independent Brownian motions, the probability density $p(\mathbf{x}, t)$ satisfies the partial differential equation (PDE),

$$\frac{\partial p}{\partial t} = \epsilon^2 \nabla^2 p - \nabla \cdot (\mathbf{a}p), \tag{1.2}$$

known as the Fokker–Planck equation (FPE) [6]. Note that the white noise in the SDEs corresponds to the diffusion in (1.2), while the drift $\mathbf{a}(\mathbf{x}, \mathbf{t})$ describes the convection. In this paper, we consider problems in which a slow variation of a control parameter through a critical point results in a delay in the transition between states for the underlying noiseless dynamics ((1.1) with $\epsilon = 0$). This delay can be changed significantly by very small noise (e.g., $0 < \epsilon \leq 10^{-4}$). This is discussed in detail in section 2.

*Received by the editors January 4, 1999; accepted for publication (in revised form) July 31, 1999; published electronically June 22, 2000.

http://www.siam.org/journals/sisc/22-1/35033.html

†School of Mathematics, University of Minnesota, 127 Vincent Hall, Minneapolis, MN 55455 (rachel@math.umn.edu). The work of this author was supported in part by NSF grant DMS-9704589.

There are several issues to consider in trying to find a solution to (1.2). For small noise ($\epsilon \ll 1$) it is expected that $p(\mathbf{x}, t)$ has large gradients, as is typical of solutions of problems with small diffusion. Then one might look for an asymptotic approximation, rather than a numerical solution. However, in the problems we study in this paper, the probability density has large gradients in some regions of space or time but not in others. Then an asymptotic approach may not be uniformly applicable. Therefore a numerical solution is necessary both to validate the asymptotic approximation where it is correct, and to provide a solution when the asymptotic approximation breaks down. The asymptotic expansions for (1.2) are discussed in detail in [7]. In this paper we focus on numerical solutions for the probability density and illustrate their complementary relationship to the asymptotic results.

A variety of numerical methods can be considered for solving (1.2). For very small diffusion, grid-based numerical methods can introduce numerical diffusion which obscures the effects of the small noise. Since the main purpose of the calculation is to determine the effect of the small noise, an appropriate method must sufficiently resolve the solution. In the applications we consider in this paper the diffusion coefficient in the Fokker–Planck equation is 10^{-8} or smaller. Under these circumstances, adapting or refining the mesh could be expensive or impractical.

Another approach is to find the probability density from simulations of (1.1), which avoids the issue of numerically solving (1.2). While this is a direct and intuitive way to find $p(\mathbf{x}, t)$ given the SDE (1.1), a very large number of simulations is necessary to obtain a smooth result for the density function. The number of simulations necessary can increase with the dimension of $\mathbf{x}$, and it also depends on the concentration of the density. We discuss the various methods further in the context of the application of section 2.

In this paper we use a gradient-particle method (GPM) to find the probability densities from (1.2) for problems with noisy delay bifurcations. Even with very small noise, the concentration and the gradient of the probability density function for these processes can vary significantly. The GPM incorporates the natural idea of simulation of (1.1) into the numerical solution of (1.2) so that the effects of the noise are not obscured and the result for the density is smooth.

In section 2 we give a description of a one-dimensional problem which illustrates why GPMs are appropriate for this problem. In section 3 we outline the algorithm, which uses ideas from [8] and [9]. In section 4 we compare the GPM results for the one-dimensional problem of section 2 with results from direct simulation (DS) and the asymptotic method of [7]. In section 5 we compute the probability density function for the noisy FitzHugh–Nagumo model and compare the GPM results with DS and the asymptotic results. In this paper we focus on the computational issues in finding the probability density function, and refer the reader to [7] for a complete discussion of the complementary asymptotic methods. A brief outline of the asymptotic approach is given in the appendix.

2. Motivation for the GPM. A description of a simple example of a delay bifurcation illustrates the important issues. The deterministic problem is

$$\frac{dy}{dt} = cy - y^3. \tag{2.1}$$

This equation was studied in [10] as a canonical model of the delay of a steady bifurcation. We summarize the results. If the control or bifurcation parameter, c, is a constant, there are two steady state solutions: $y = 0$ for all c and $y = \sqrt{c}$ for $c > 0$.

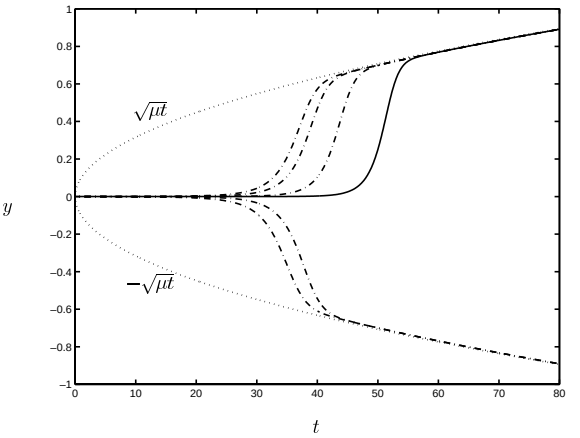


FIG. 2.1. *The solution of* (2.2) *for* $\epsilon = 0$ *and* $\epsilon = 10^{-4}$. *The solid line is the deterministic solution of* (2.2) *with* ($\epsilon = 0$), *and the dash-dotted lines are five realizations of the solution of* (2.2) *with* $\epsilon = 10^{-4}$. *In both cases the initial condition was* $y(0) = 10^{-6}$ *and* $\mu = .01$. *Note that the noisy solution makes the transition to a solution near* $\pm\sqrt{\mu t}$ *much sooner than the deterministic solution does; that is, the delay of the transition is reduced in the presence of noise.*

The solution $y = \sqrt{c}$ is stable for $c > 0$. That is, there is a steady bifurcation from the zero solution to the solution $y = \sqrt{c}$ at the critical parameter value $c = 0$.

Now consider $c = \mu t$ for $0 < \mu \ll 1$. When the system is subcritical, ($\mu t < 0$), $y(t)$ decays exponentially. As t increases so that $t > 0$, $y(t)$ remains small for a long interval of time. For $\sqrt{\mu t} = O(1)$, $y(t)$ grows rapidly, approaching $y(t) \sim \sqrt{\mu t}$. This is known as a delay bifurcation, since the solution does not immediately increase rapidly as the coefficient μt varies slowly through the critical point $\mu t = 0$. This behavior can be seen from the exact solution of (2.1).

Now we consider the effect of noise on the delay bifurcation. Previous studies have considered the reduction of the delay with the addition of sinusoidal oscillations [10]. Numerical simulations of

$$dy = (\mu t y - y^3)dt + \sqrt{2}\epsilon dW, \tag{2.2}$$

where W is standard Brownian motion, demonstrate that the delay is reduced with the addition of noise [11], even though $\epsilon = 10^{-4}$ (see Figure 2.1). The simulations of Figure 2.1 show that the noisy dynamics are close to the deterministic dynamics, except during the transition from $y \ll 1$ to $y \sim \pm\sqrt{\mu t}$. These simulations suggest that the probability density function is sharply peaked about either $y = 0$ or $y = \pm\sqrt{\mu t}$, except in the transition region. This variation suggests that the concentration of the probability density function changes significantly over time.

A quantitative description of the delay can be obtained by considering the FPE for the probability density $p(y, t)$, defined as

$$p(y, t)dy = \text{ P}(y(t) \in (y, y + dy) \text{ at time } t), \tag{2.3}$$

for the process $y(t)$ described in (2.2). For $\epsilon \neq 0$ we look for the solution of

$$p_t = \epsilon^2 p_{yy} - ((\mu t y - y^3)p)_y = 0, \quad p(0) = p_0(y). \tag{2.4}$$

An asymptotic approximation can be made for this density when the random dynamics are close to the deterministic dynamics [7], but from the few typical realizations in

Figure 2.1, that leaves a significant region (in this case between $t = 30$ and $t = 50$) in which this approximation may fail. Therefore we are especially interested in computing $p(y,t)$ in this transition region. As mentioned in the introduction, $p(y,t)$ gives the probability that the process has made the transition from $y \sim 0$ to $y \sim \pm\sqrt{\mu t}$ at time t and can be used to compute moments of the process $y(t)$ or the expected time until this transition is made [7].

In section 4 we use a GPM to compute the probability density function for this one-dimensional example of a delay bifurcation. In section 5 we extend this method to a two-dimensional model, the FitzHugh–Nagumo model. There the noise also reduces the time of a delay bifurcation. We have several reasons for using GPMs which follow from the discussion above.

1. The diffusion in the FPE is very small for the cases of interest, that is, $\epsilon \ll 1$. (In particular, $\epsilon \leq .0001$ in this paper.) Insufficiently resolved grid-based methods give numerical diffusion which swamps the diffusion due to the noise.

2. The probability density has steep gradients in some regions in space and time and not in others. This is due to the fact that the noisy dynamics follow the deterministic dynamics closely except in the transition region. Outside of the transition region the gradients are large, but within this region the gradients are no longer large. This variation in shape, and the transition of the process itself, could make adaptive methods expensive. Such a method would require frequent regridding, based on some prior knowledge given by analysis or preprocessing or on monitoring the evolution of the solution at each step.

3. A straightforward simulation method involving only direct simulation of the SDE would require a large number of realizations in order to resolve the probability density in all regions of space and time of interest. The natural idea of simulating the stochastic motion by solving the SDE (1.1) is incorporated in the GPM. It requires many fewer realizations than the DS method since both particle positions and gradients are computed. We compare the GPM results to the DS results in sections 4 and 5.

3. The method. We outline the GPM for computing the probability density function $p(\mathbf{x},\mathbf{t})$ for the process described by (1.1). Here $\mathbf{x}$ and $\mathbf{a}(\mathbf{x},t)$ are vectors (in one and two dimensions for the applications in this paper) and $d\mathbf{W}$ is a vector of independent white noise processes. Then the probability density satisfies the Fokker–Planck equation (1.2) with initial condition $p(\mathbf{x},0) = p_0(\mathbf{x})$. In one dimension $\mathbf{x} = x$ and $\mathbf{a}(\mathbf{x},t) = a(x,t)$, and the equation for $p_x(x,t)$ is

$$(p_x)_t = \epsilon^2 \frac{\partial^2 p_x}{\partial x^2} - (a p_x)_x - a_{xx} p - a_x p_x. \tag{3.1}$$

In two dimensions we use the notation of the application in section 5, taking $\mathbf{x} = (u,v)$, $\mathbf{g} = (g_1, g_2) \equiv (p_u, p_v)$, and $\mathbf{a} = (a_1, a_2)$ and differentiating (1.2),

$$\begin{aligned} \frac{\partial}{\partial t}\begin{pmatrix} g_1 \\ g_2 \end{pmatrix} &= \epsilon^2 \nabla^2 \begin{pmatrix} g_1 \\ g_2 \end{pmatrix} - \frac{\partial}{\partial u}\left[a_1 \begin{pmatrix} g_1 \\ g_2 \end{pmatrix}\right] - \frac{\partial}{\partial v}\left[a_2 \begin{pmatrix} g_1 \\ g_2 \end{pmatrix}\right] \\ &\quad - \begin{bmatrix} (a_1)_u & (a_2)_u \\ (a_1)_v & (a_2)_v \end{bmatrix} \begin{pmatrix} g_1 \\ g_2 \end{pmatrix} - \begin{pmatrix} (a_1)_{uu} + (a_2)_{vu} \\ (a_1)_{uv} + (a_2)_{vv} \end{pmatrix} p. \end{aligned} \tag{3.2}$$

These equations are similar to those in [8], where the convection-diffusion equation for a chemical concentration in an incompressible flow was solved with GPMs. The main difference between [8] and this application is that the equation for the gradient

in this paper involves both ∇p and p itself, while the equation for the gradient in [8] involves only the derivatives of p. This is because the flow was incompressible ($\nabla \cdot \mathbf{a} = 0$) for the convection-diffusion equation studied in [8]. This is not true for (1.1) in general, and, in particular, $\nabla \cdot \mathbf{a} \neq 0$ for the delay bifurcation applications which we study here. It follows that there is a reconstruction of p from ∇p in each time step. A similar reconstruction was required in [9] for the solution of a reaction-diffusion system using a gradient random walk method. We outline the steps of the algorithm, pointing out differences and similarities to [8] and [9]. Both [8] and [9] contain excellent discussions of GPMs and comparisons with previous work in deterministic and random particle methods. We do not repeat their discussion here but mention a few alternative approaches at the end of this section.

The algorithm is based on the observation that both the particle motion and the evolution of the gradient vectors contribute to changes in ∇p and thus in p as well [8]. Specifically, in (3.1) the terms $\epsilon^2(p_x)_{xx} - (ap_x)_x$ correspond to changes in concentration due to the motion of the particles as described by (1.1). The terms $-a_{xx}p - a_x p_x$ give the changes in shape due to the evolution of p_x. Similarly in the two-dimensional case, the terms $\epsilon^2\nabla^2(g_j) - \nabla \cdot (\mathbf{a}g_j)$ for $j = 1, 2$ are due to the motion of the particles, and the remaining terms on the right-hand side of (3.2) describe changes in the gradient vectors. The motion of the particles is obtained by solving (1.1).

One additional note is that the probability densities described above are defined on an infinite domain. However, the probability densities are exponentially small outside a bounded region. Therefore we compute on a rectangular region which contains all but the tails of the probability density whose mass is essentially zero and set $p(\mathbf{x}, t) = 0$ on the boundary of this region.

THE ALGORITHM:

Initial conditions: To each of N particles assign position vectors $\mathbf{x}_k$, the probability density $p_k \equiv p_0(\mathbf{x}_k)$ from the initial condition, and gradient vectors $\mathbf{g}_k \equiv \nabla p_k$ for $k = 1, \ldots, N$.

At each time step:

1. Find the new position of the particles by solving (1.1). In this paper a second order weak explicit scheme for SDEs [12] was used. Since the coefficient of the noise is a constant, the scheme simplifies considerably,

$$\mathbf{x}(t + \Delta t) = \mathbf{x}(t) + \frac{1}{2}(\mathbf{a}(\mathbf{z}(t + \Delta t), t + \Delta t) + \mathbf{a}(\mathbf{x}(t), t))\Delta t + \epsilon\sqrt{2\Delta t}v_n,$$

$$\mathbf{z}(t + \Delta t) = \mathbf{x}(t) + \mathbf{a}(\mathbf{x}(t), t)\Delta t + \epsilon\sqrt{2\Delta t}v_n, \tag{3.3}$$

where v_n is a standard normal random variable, generated at the nth time step.

2. Evolve the gradient vectors using the remaining terms from the equation. For the one-dimensional example, $g_k = (p_x)_k$ and $(g_k)_t = -a_{xx}p_k - a_x g_k$, and a_{xx} and a_x are evaluated at x_k. In two dimensions the equation is

$$\frac{\partial \mathbf{g}_k}{\partial t} = -\begin{bmatrix} (a_1)_u & (a_2)_u \\ (a_1)_v & (a_2)_v \end{bmatrix} \begin{pmatrix} g_{1k} \\ g_{2k} \end{pmatrix} - \begin{pmatrix} (a_1)_{uu} + (a_2)_{vu} \\ (a_1)_{uv} + (a_2)_{vv} \end{pmatrix} p_k. \tag{3.4}$$

This step is similar to one used in [8], but there is an extra term involving p_k that was not present in the applications studied in [8], as discussed above. A second order Runge–Kutta method was used to advance $\mathbf{g}_k$.

Note that this step is not present in the method used for reaction-diffusion equations in [9] and [13]. Instead changes in the shape of the density were obtained by either creating or destroying particles, depending on the sign of terms analogous to the right-hand side of (3.4). In this paper the number of particles is constant throughout the computation, and changes in the shape of the density enter through the evolution of the gradient computed in this step.

3. Reconstruct p_k from its gradient vector $\mathbf{g}_k$. In one dimension this is accomplished by computing $\hat{\mathbf{g}}_k = (p_k)_x \Delta x_k$, then sorting the particles and approximating the integral of $\mathbf{g}_k$ with a sum of the $\hat{\mathbf{g}}_k$'s to get p_k.

In two dimensions, using $\mathbf{x}_k = (u_k, v_k)$, we find it convenient to compute $\tilde{\mathbf{g}}_k = \nabla p_k \Delta u \Delta v$. Then $\nabla p(u, v, t)$ is determined on a lattice with points given by (u_j, v_m), for $j = 1, \ldots, N_u$, $m = 1, \ldots, N_v$, as follows:

$$\begin{pmatrix} p_u(u_j, v_m, t) \\ p_v(u_j, v_m, t) \end{pmatrix} = \sum_{k=1}^{N} \tilde{\mathbf{g}}_k(t) \delta_{\mathbf{x}}((u_k - u_j), (v_k - v_m)). \tag{3.5}$$

Here $\delta_{\mathbf{x}}$ approximates the two-dimensional Dirac delta function, using the piecewise linear hat function as in [8],

$$\begin{aligned} &\delta_{\mathbf{x}}((u - u_j), (v - v_m)) \\ &= \frac{1}{\Delta u \Delta v} \max\left(0, 1 - \frac{|u - u_j|}{\Delta u}\right) \max\left(0, 1 - \frac{|v - v_m|}{\Delta v}\right). \end{aligned} \tag{3.6}$$

Then $p(u, v, t)$ is obtained using the method outlined in [8], by solving two Poisson's equations. Note that this reconstruction must be performed at every time step since the evolution equation for the gradient, as in (3.4), depends on p. Briefly, the reconstruction of $p(u, v, t)$ is computed on the rectangle $a \le u \le b$, and $c \le v \le d$,

$$\begin{aligned} &\nabla^2 \phi = p_u, \quad \phi(a, v) = \phi(b, v) = \phi_v(u, c) = \phi_v(u, d) = 0, \\ &\nabla^2 \psi = p_v, \quad \psi_u(a, v) = \psi_u(b, v) = \psi(u, c) = \psi(u, d) = 0, \\ &p(u, v, t) = \phi_u + \psi_v + \text{ constant}. \end{aligned} \tag{3.7}$$

Here the rectangle is sufficiently large so that $p(u, v, t)$ is essentially zero on the boundaries. This reconstruction method is discussed in [8]. As noted there, we found that this reconstruction gives a nonnegative result for the density. The constant in (3.7) was chosen in order to properly normalize $p(x, y, t)$ for any t, so that the integral over space of the density is 1. We found that this constant approached zero as the number of points in the reconstruction lattice was increased and was negligible as compared to the maximum of the density function.

In [9] the reconstruction step used a method similar to random vortex methods [14]. The method suggested by Anderson [15] uses Poisson's formula to obtain a point-gradient formulation for the recovery of $p(u, v, t)$. This method is expensive, but fast multipole methods of [16] can be used to reduce the computation time. Some smoothing problems were encountered in [9] in using the fast multipole methods. Since we found the implementation straightforward and the method (3.7) sufficiently fast we used the method of [8]. As noted there, this reconstruction method does introduce some errors when the gradients are very large. In our problem these errors are apparent in the tails of the density, but they did not affect the bulk of the density. These effects are discussed further in the following sections. We used the *fishpack* subroutines [17] to solve (3.7).

In the next sections we illustrate the use of particle methods for these applications and compare the results with other methods. We have not explored all possibilities for optimizing run time. Higher order methods could be used to solve (1.1), which could introduce some savings even though these methods require that additional random variables be generated at each time step [12]. Higher order methods could also be used to evolve $\mathbf{g}_k$. We also found that adaptive initial placement of the particles, based on experience with the solution, could be used to reduce the number of particles necessary for resolution. However, we did not pursue this further than trying a few different initial placements of particles for each case. Future work could explore these time saving steps and compare this reconstruction method with that of [15] for the noisy delay bifurcation problems described here.

4. The one-dimensional model of a steady bifurcation. First we give the results for the probability density for the one-dimensional steady delay bifurcation example as discussed in section 2. The equation for $p(y,t)$ is given by

$$p_t = \epsilon^2 p_{yy} - ((\mu t y - y^3)p)_y, \tag{4.1}$$

$$p(0) = p_0(y). \tag{4.2}$$

As an initial condition, we take

$$p_0(y) = \frac{1}{\sqrt{2\pi\epsilon^2}} \exp\left(-\frac{(y-y_0)^2}{2\epsilon^2}\right). \tag{4.3}$$

As discussed above, y decays exponentially for $t < 0$. Then starting with (4.3) is the same as taking an initial condition at $t < 0$ so that $y(0) < \epsilon$ and the noise dominates the dynamics at the critical point $t = 0$. The figures compare computations of $p(y,t)$ using 10,000 and 15,000 particles in the GPM with $p(y,t)$ obtained from a DS of the SDE (2.2). The results at $t = 36$ are also compared with an asymptotic approximation which is valid through a part of the transition region, as discussed in detail in [7], and summarized in the appendix.

In Figure 4.1(a), we see good agreement between all methods, noting that the direct simulation with 60,000 realizations (particles) has statistical errors, while the GPM method gives a much smoother solution. In Figure 4.1(b) this difference is even more obvious. Figure 4.1 shows that the GPM can be used to compute the probability density through the transition region and gives the probability density before and after this transition has occurred (mean transition time is shown to be $t \approx 28$ in [7] for this case). Specifically, Figure 4.1(a) shows the probability density while the process is still concentrated around $y = 0$. Figure 4.1(b) shows the density within the time interval in which the process makes the transition from $y \ll 1$ to $|y| \sim \sqrt{\mu t}$, and Figure 4.1(c) shows the density at a time when the transition is likely to have occurred. Thus the GPM result is valid throughout the transition interval, which is a longer time range than the validity interval for the asymptotic approximation. The reconstruction of the density from its gradient provides the smoothing which gives good results even though the GPM uses only 10,000 particles, as compared to the nonsmooth results from the DS of the particle motion, using 60,000 particles. Even though the GPM requires additional computations of the evolution of the gradients and the reconstruction of $p(y,t)$ from $p_y(y,t)$, the computation times for the GPM with 10,000 particles and the DS with 60,000 particles are comparable (overnight on a Pentium II workstation to compute to $t = 40$). Note that the time step can not be too large, in order to keep computational errors below the noise level.

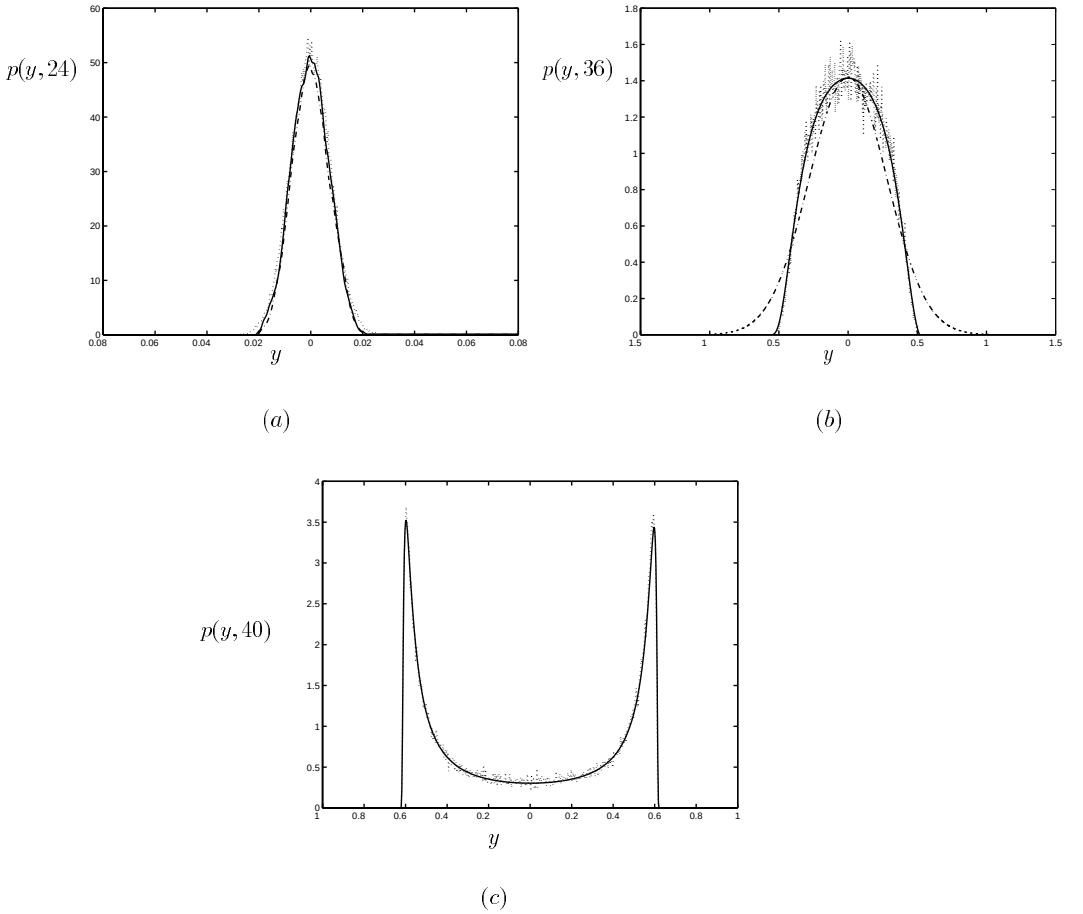


FIG. 4.1. *Comparison of results for the density at* $t = 24$, $t = 36$, *and* $t = 40$, *with* $\mu = .01$ *and* $\epsilon = 10^{-4}$ *and initial condition* (4.3). (a) *The solid line is the GPM solution with* 10,000 *particles, the dash-dotted line is the GPM solution for* 15,000 *particles, and the dotted line is the result from the DS of* (2.2) *with* 60,000 *realizations (particles). Although it is not shown, the asymptotic approximation of* [7] *is in good agreement with all of these results.* (b) *The solid line is the GPM solution with* 10,000 *particles and the dotted line is the result for DS of* (2.2) *with* 60,000 *realizations (particles). The dash-dotted line is the asymptotic (Gaussian-type) approximation, which does not capture the behavior of the tails for this value of* t, *when the density is less concentrated.* (c) *The solid line is the GPM solution with* 10,000 *particles, and the dotted line is the result from the DS of* (2.2) *with* 60,000 *particles. The numerical results are in good agreement as far as location and general shape of the density, but the DS results are not smooth.*

Recalling that the density $p(y,t)$ is reconstructed from $p_y(y,t)$ at each step in the GPM method, one might expect that errors in the reconstruction can feedback into the computation. We found that these errors could be significant, especially in the transition zone. Then the gradient is not large, except near $y = \pm\sqrt{\mu t}$, where the density function drops off sharply. The reconstruction errors for $p(y,t)$ near $y = \pm\sqrt{\mu t}$ can appear in the computation near this sharp gradient in the density function. By decreasing the time-step and using a sufficiently fine grid, we reduced this error substantially. For example, for times before the transition ($t < 28$) we used a time step of $\Delta t = .0005$, in order to keep the errors of the computations ($O(\Delta t^2)$) well below the noise level ($\epsilon = 10^{-4}$). If we computed until $t = 40$ with this same Δt

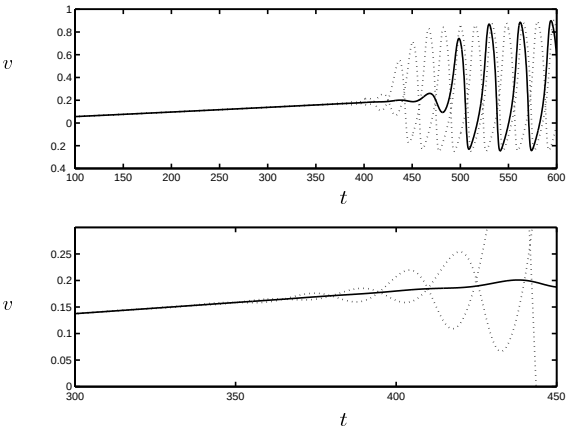


FIG. 5.1. *The solid line is the solution of v from* (5.1) *with $\epsilon = 0$, while the dotted line is the solution for v from* (5.1) *with $\epsilon = 10^{-5}$, $a = .2$, $b = .05$, $\gamma = .4$, and $I(t) = \mu t$ with $\mu = .001$. The bottom figure is an enlargement of the top figure. The bifurcation point for the problem when I is constant is $I_c \approx 273$. Then it is clear that there is a delay in the transition from steady to oscillatory behavior, but the delay is reduced when small noise is present.*

we would eventually see errors in the tails of the density which were approximately 5–10% of the peak of the density. To avoid these reconstruction errors, we reduced the size of the time step through the transition region. For $28 < t < 32$ we used $\Delta t = .00025$, but for $t > 32$ we found the computation increasingly sensitive to reconstruction errors. When we used $\Delta t = .0001$ for $32 < t < 36$ and $\Delta t = .00001$ for $36 < t < 40$, we found an error in the tails that was approximately 3–4% of the height of the peak of the density. By using $\Delta t = .00001$ for $32 < t < 36$ and $\Delta t = .000005$ for $36 < t < 40$, we reduced this error, as shown in Figure 4.1(c).

5. The two-dimensional model: FitzHugh–Nagumo. Next we consider the two-dimensional model of FitzHugh–Nagumo [2], [18],

$$du = b(v - \gamma u)dt,$$

$$dv = (-f(v) - u + I(t))dt + \sqrt{2}\epsilon dW, \tag{5.1}$$

$$f(v) = v(v-1)(v-a), \tag{5.2}$$

where a, b, and γ are physical parameters. This is a reduced model of the propagation of neural impulses in the giant axon of a squid, with v the potential difference across the membrane of the axon, and u the recovery current. $I(t)$ is an applied current and plays the role of the bifurcation parameter. When I is constant, there is a Hopf bifurcation at $I = I_c$; for $I < I_c$ the stable solution is constant, while for $I > I_c$ the stable solution is periodic. Figure 5.1 shows that there is delay in the bifurcation from constant to periodic behavior as the applied current $I(t) = \mu t$ is increased slowly ($\mu \ll 1$). In the realizations with small noise ($\epsilon = 10^{-5}$ in Figure 5.1) this delay is significantly reduced. Only the solution for v is shown in Figure 5.1, since the variable u is virtually slaved to v. That is, the behavior of u follows that of v, with a time lag. Then a transition to oscillatory behavior in v is followed by a transition to oscillatory behavior in u. Since the noise causes a reduction in the delay until the transition to the oscillatory state, we anticipate that the gradient of the probability density varies significantly in the transition region. The probability density for this process satisfies

(1.2) with $a_1 = b(v - \gamma u)$ and $a_2 = -f(v) - u + I(t)$ so that

$$\frac{\partial p}{\partial t} = \epsilon^2 \nabla^2 p - ((b(v - \gamma u)p)_u - ((-f(v) - u + I(t))p)_v. \tag{5.3}$$

We determine $p(u, v, t)$ using the GPM and compare to the results from the direct simulation of (5.1) and the asymptotic approximation of [7], summarized in the appendix. In the computations the initial condition was

$$p(u, v, t_0) = p_{asymp}(u, v, t_0), \tag{5.4}$$

where $p_{asymp}(u, v, t_0)$ is the asymptotic approximation to the density obtained in [7] and $t_0 > 0$ is some time before the transition time. This choice of initial condition is justified by the dynamics of the system. For a typical realization, as shown in Figure 5.1, the stochastic and deterministic behavior are close until t approaches the transition time.The asymptotic approximation and the numerical computations agree for t below the transition interval, which Figure 5.1 suggests is $t < 370$ for the parameter values used there. We verified this agreement by starting at different initial times $t_0 < 350$ with initial conditions of the type (5.4). As expected, the asymptotic approximation is valid until large values of t where the probability function is less concentrated around the deterministic dynamics.

The use of the initial condition (5.4) also illustrates the complementary nature of the asymptotic approximation and the computational method; the asymptotic approximation is used as an initial condition at t_0 before the transition time, and then the computational method is used to find the probability density in the transition region, where the validity of the asymptotic method is suspect. This is a very practical reduction of the amount of computation necessary, since the computational method does not perform as well as the asymptotic method for values of t_0 before the transition time, where the density function has large gradients. The computation time is significantly reduced, since the computations are done for $t > t_0 > 0$, rather than for all $t > 0$.

In Figure 5.2 we compare the results from GPM and the asymptotic method of [7] for $t = 375$ and the same parameter values as in Figure 5.1. These figures show good agreement between the two methods. This is expected, since the process has not reached the expected transition time (calculated as $t \approx 383$ in [7]), and the density is still concentrated around the deterministic behavior. In Figure 5.3 we compare the GPM and asymptotic method for $t = 390$ and $t = 410$ for the same parameter values as in Figure 5.2, and in Figure 5.4 we compare the results from the GPM and asymptotic method for a different value of $\mu = .003$. In both of these cases the results are compared for times before and after the transition to oscillatory behavior. Again good agreement is seen for the location of the concentration of the density and the shape of the peak. The shape and location of the density is consistent with the simulations and the dynamics of (5.1). The density remains centered around the deterministic dynamics, but the variable v makes the transition to the oscillatory state first, so that the density is less concentrated about the deterministic value of v. The gradients are larger in the variable u since u follows v with a time lag. That is, the gradients in u are also decreasing but not as fast as in the v direction.

Errors in the reconstruction of p can accumulate in our computations, since the reconstruction must be carried out at each time step, and the result for p is used in the evolution of the gradient vectors (3.4) at the following time step. The computations in [8] did not have this accumulation of error, since the concentration (analogous

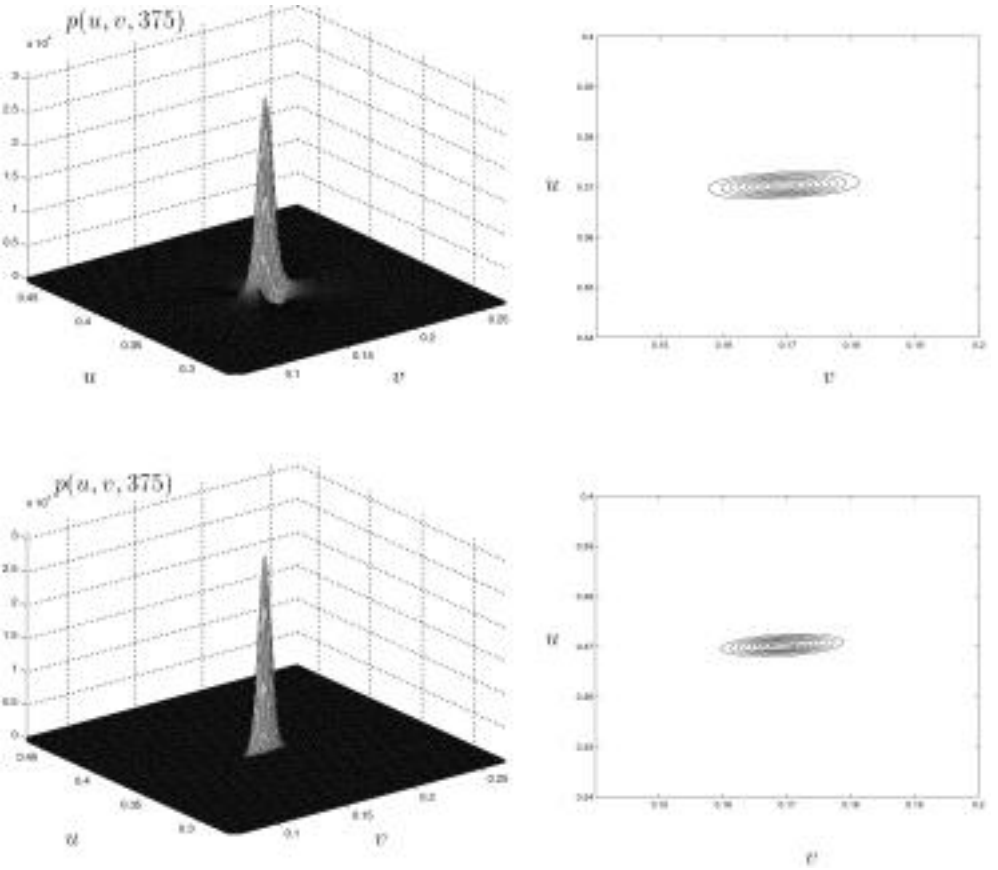


FIG. 5.2. *Comparison of the probability density $p(u, v, t)$ computed with the GPM of this paper (top row) with the asymptotic result from* [7] *(bottom row). The parameter values are the same as in Figure* 5.1*, and $\epsilon = 10^{-5}$. The left column is the surface plot, and the right column is the corresponding contour plot. Good agreement in both location and shape is obtained from the two methods. Note that this is for a time before the transition has occurred.*

to $p(u, v, t)$) did not appear in the equation for the gradient (see section 3). There is some oscillation in the tails of the numerical result which is especially evident at $t = 410$ in Figure 5.3 and $t = 145$ in Figure 5.4. There the contour line at the base of the peak is irregularly shaped (not elliptic). This is due to the reconstruction process, which has difficulty resolving the very steep gradients. We found an increased effect of these errors when $|p_u|$ is much larger than $|p_v|$, which is far into the transition zone. Since this p is reconstructed from ∇p at every time step, these errors can accumulate over long computation times, as is required for these computations. As in the one-dimensional problem of section 4, these errors can be reduced by decreasing the time step and adjusting the reconstruction lattice. We chose the lattice and time step so that the computations would not take longer than 10–20 hours on a Pentium II workstation, times which are comparable to direct simulation times. As in the one-dimensional example, we took $\Delta t = .0005$ for times before the transition. As the density began to spread out, we reduced the time step to $\Delta t = .0001$, and the errors in the tails were reduced by a few percent of the maximum height of the density. The largest reconstruction lattice that we used was 512×256, and the smallest time

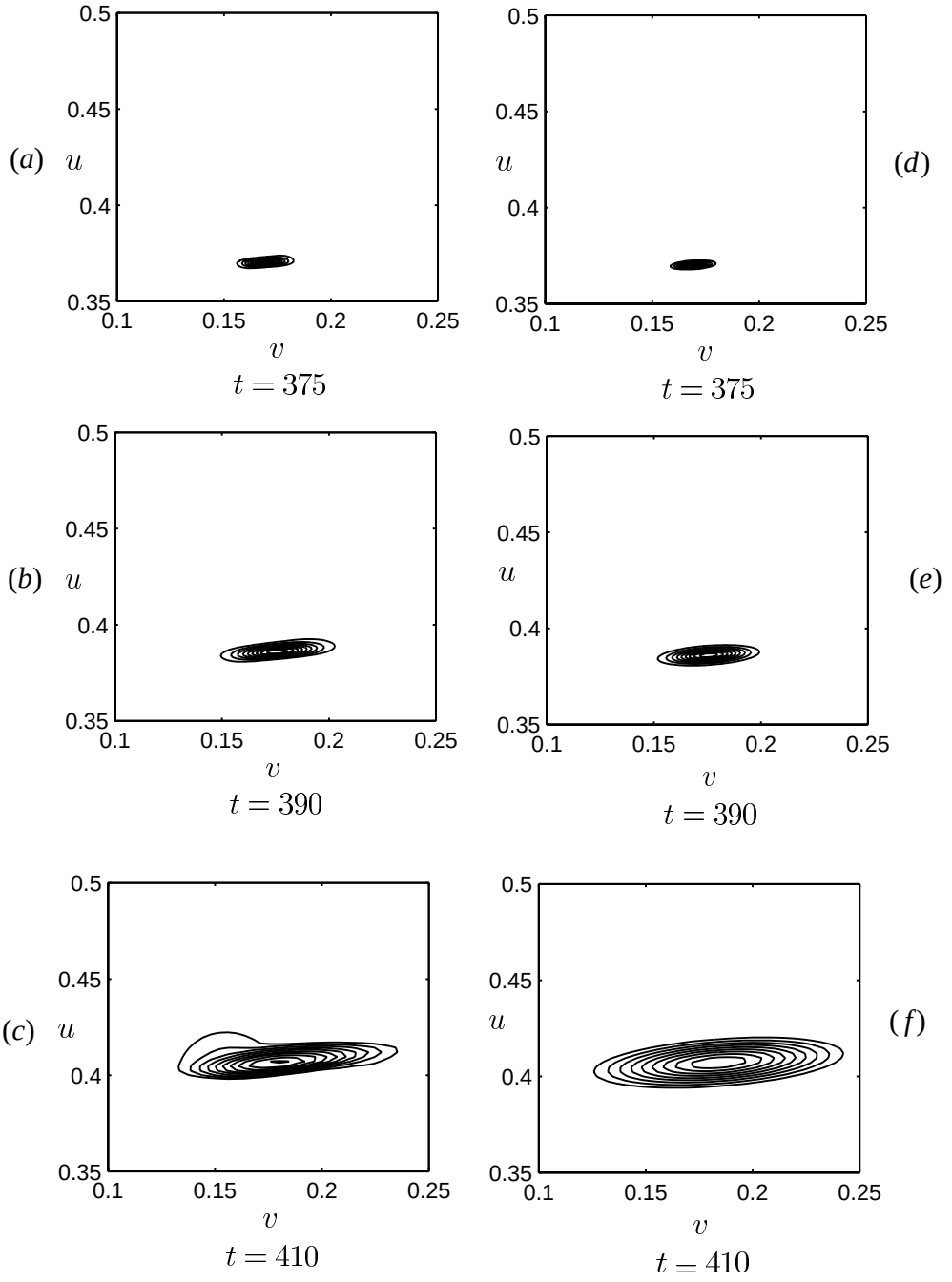


FIG. 5.3. *Comparison of the contour plots for* $p(u, v, t)$ *for the asymptotic result* [7] *with GPM results for* $t = 375, t = 390, t = 410$. *The parameters are the same as in Figure* 5.2. *The* (a)–(c) *are the results from the GPM. The* (d)–(f) *are the asymptotic results.*

step was .00001. Even with these extreme cases the error in the tails of the density (approximately 2–3% of the peak of the density) was not completely eliminated. In

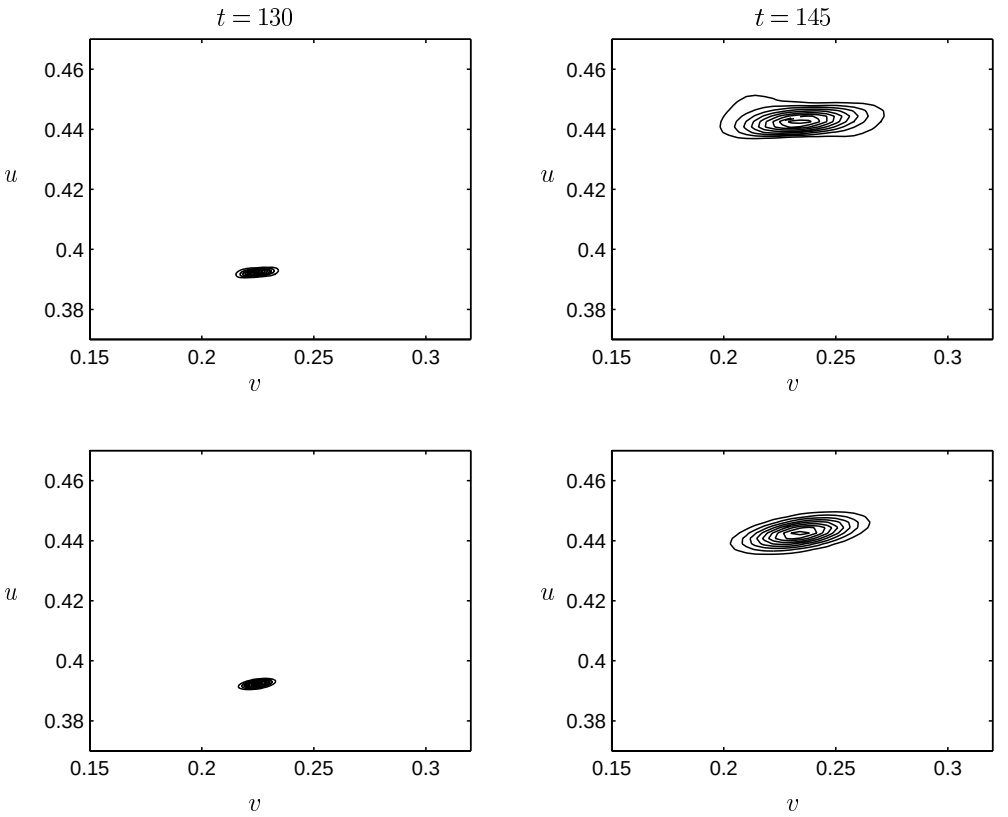


FIG. 5.4. *Comparison of the contour plots for $p(u, v, t)$ for the asymptotic result* [7] *with GPM results for the parameter values $\epsilon = 10^{-5}$, $a = .2$, $b = .05$, $\gamma = .4$, and $I(t) = \mu t$ with $\mu = .003$. The plots are compared for times before ($t = 130$) and after ($t = 145$) the transition. Again there is a large change in the shape of the density. Accumulation of reconstruction errors is reduced as described in the text. The top row are the results from the GPM. The bottom row are the asymptotic results.*

this application we found it more effective to reduce the errors by using the asymptotic approximation for an initial condition at larger values of t, where the approximation is still valid and agrees with the computation. (For Figure 5.3 we used (5.4) with $t_0 = 390$, and for Figure 5.4 we used (5.4) with $t_0 = 135$.) This adjustment reduces the computing time and the accumulation of error. The resulting reduction of the error can be seen in comparing Figures 5.3 ($\mu = .001$) and 5.4 ($\mu = .003$). In Figure 5.4, the parameter μ is larger ($\mu = .003$) and the transition occurs at a faster rate. Then the probability density after the transition is obtained from computing over a shorter time interval, and we found less accumulation of reconstruction errors for $\mu = .003$.

In Figure 5.5 we compare the results for $p(u, v, t)$ from solving (5.3) with a GPM and from the DS of (5.1). The results from the DS use 80,000 realizations, while the GPM uses 10,000 particles. The statistical errors in the DS are increasingly evident as time increases since the density is spreading out. An increasing number of realizations would be necessary to properly approximate the density for later times. The results for the GPM method are smooth, and the number of particles is sufficient for resolving the peak in the density. Other solutions were obtained using a larger number of particles in the GPM, but they gave essentially the same results.

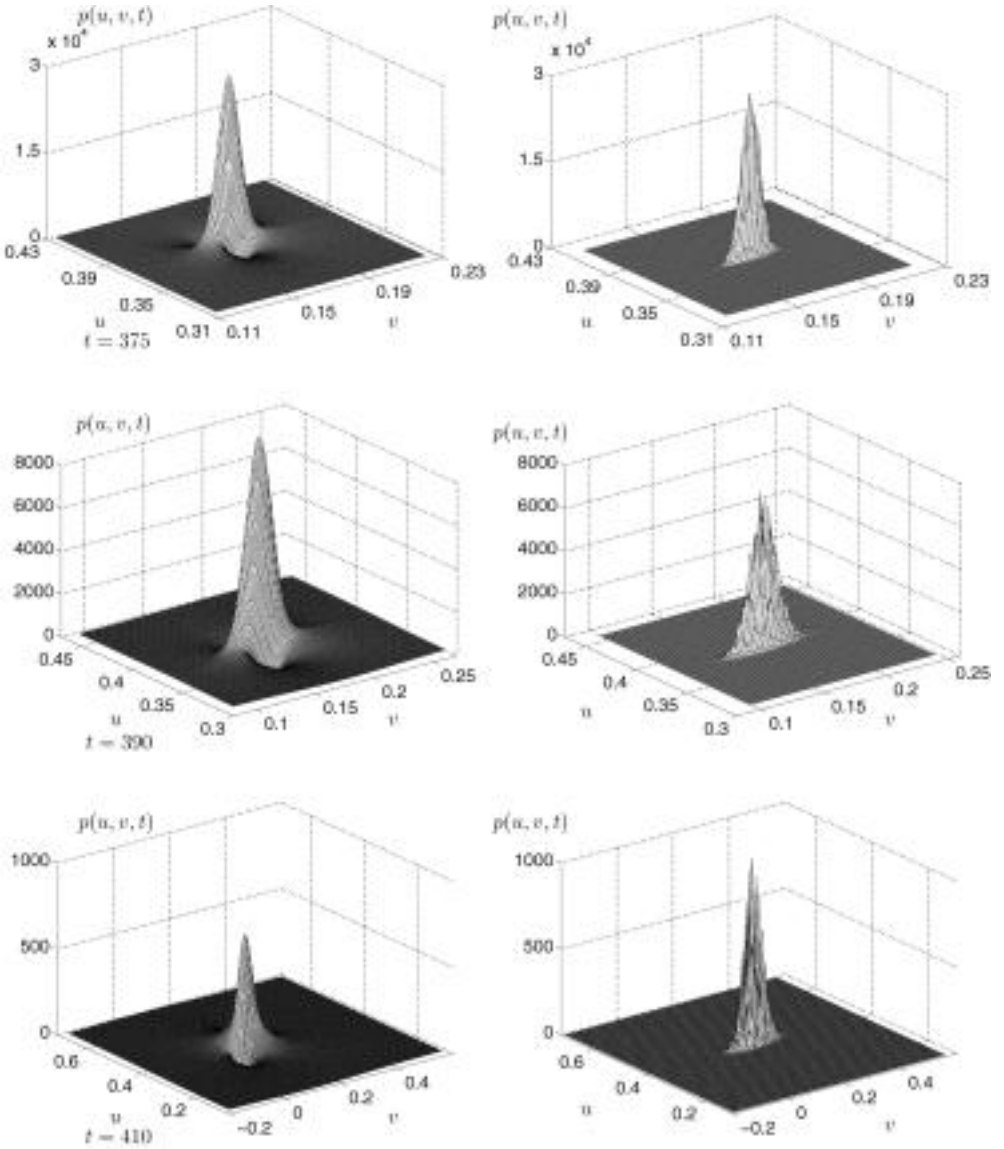


FIG. 5.5. *Comparison of* $p(u, v, t)$ *computed as the solution of* (5.3) *with the GPM (left column) using* 10,000 *particles and computed by DS of* (5.1) *(right column) using* 80,000 *realizations (particles) for* $t = 375$, $t = 390$, *and* $t = 410$. *The parameters are the same as in Figures* 5.2 *and* 5.3. *Note the different scales of the plots, which indicate that the density is less concentrated for increasing time. As time increases, the results of the DS get increasingly rough, which indicates that an increasing number of realizations is necessary to obtain an approximation for* $p(u, v, t)$. *Here* $a = .2$, $b = .05$, *and* $\gamma = .4$, $\mu = .001$, *and* $\epsilon = 10^{-5}$.

In Figure 5.6 we compare the results for the marginal density

$$p(v,t) = \int_{-\infty}^{\infty} p(u,v,t)du. \tag{5.5}$$

The dynamics of v indicate the transition time since the transition in u follows that of

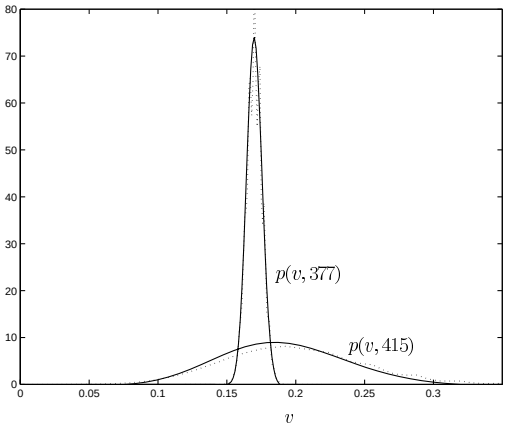


FIG. 5.6. *Comparison of the marginal density $p(v,t)$ obtained by the DS of* (5.1) (80,000 *realizations*) *and the GPM for $t = 377$ and $t = 415$. Here $a = .2$, $b = .05$, $\gamma = .4$, $\mu = .001$, and $\epsilon = 10^{-5}$.*

v. Therefore $p(v,t)$ itself may be used to determine when the transition has occurred, as in [7]. Figure 5.6 compares the results for the marginal density, using $p(u,v,t)$ for the GPM and the DS of (2.2). Note that even with the extra step of smoothing, that is, integration in u, the results from the DS do not give a smooth result for the marginal density.

6. Conclusion. A gradient particle method is used to solve FPEs for noisy delay bifurcations. The solution of the FPE gives the time-dependent probability density function for the stochastic process.

In these problems, a delay in the transition between states occurs in the deterministic dynamics when a bifurcation parameter varies slowly through the bifurcation point. This delay is reduced by the presence of noise, even if it is very small. The probability density gives a complete description of the process, and in particular it describes the transition of the process. Since there is considerable variation in the gradient of the density, numerical methods must be used to solve the FPE, yielding results which are complementary to those obtained asymptotically. The small diffusion of the cases of interest (very small noise) and the variation in the concentration of the density suggest that the GPMs are practical methods for computing the solution to the FPE. These methods incorporate simulation of the noisy process (motion of the particles) with the evolution of the gradients of the probability density. The reconstruction of the density from its gradient introduces a smoothing so that fewer realizations are necessary than in computing the density with a direct simulation.

We compute probability densities through the transition regions and compare the GPM results to asymptotic results and direct simulations. The GPMs yield satisfactory results throughout the transition regions, where the asymptotic method is no longer valid and yield results which are much smoother than those obtained with direct simulation. Some caution must be used in applying the GPMs, since the density must be reconstructed from its gradient at each time step and errors can accumulate over long time periods. These errors can be reduced by adjusting the time step and reconstruction grid, and by using the asymptotic approximation as an "initial condition," which reduces the length of the time of computation.

Appendix. The asymptotic method. We give a brief summary of the asymptotic method of [7] which was used to approximate the probability densities in sections 4 and 5. The asymptotic method is based on a Gaussian-type ansatz for the form of the probability density. With this ansatz the problem is reduced to solving a system of ODEs.

For the one-dimensional example of section 4, the ansatz for $p(y,t)$ is

$$p(y,t) \sim C\left[\frac{1}{2}\sqrt{g(t)}e^{-(y-|F(t)|-\frac{2\epsilon}{\sqrt{g_n}})^2\frac{g(t)}{2\epsilon^2}} + \frac{1}{2}\sqrt{g(t)}e^{-(y+|F(t)|-\frac{2\epsilon}{\sqrt{g_n}})^2\frac{g(t)}{2\epsilon^2}}\right]. \tag{A.1}$$

Here $F(t)$ is the solution of the deterministic problem (2.1) for $c = \mu t$, $g(t)$ is a function to be determined, and C is a normalization constant. By substituting (A.1) into the FPE (2.4) and using a perturbation expansion about $\epsilon = 0$, an ODE for $g(t)$ is found at leading order

$$-\frac{g'(t)}{2} = g^2 + (\mu t - 3F^2(t))g. \tag{A.2}$$

This equation has the solution

$$g(t) = \frac{e^{-2Q}}{2\int_0^t e^{-2Q(t')}dt' + c_g} \equiv \frac{g_n(t)}{2\int g_n(t')dt'}, \tag{A.3}$$
$$Q(t) = \int^t \mu t' - 3F^2(t')dt', \quad c_g = \text{constant},$$

and the result can be substituted in (A.1) to give an asymptotic approximation to $p(y,t)$.

The idea behind (A.1) is that the density is approximately Gaussian, centered around the deterministic dynamics given by $F(t)$. The concentration of the density varies in time, with $\epsilon^2/g(t)$ playing the role of a time-dependent variance. When the function $g(t)$ is very small, the variance is large and the Gaussian-type approximation is no longer valid. This point was discussed in section 4, where the asymptotic approach fails when the density is less concentrated in the transition region (see Figure 4.1(b)). Note that the ansatz (A.1) is actually a linear combination of two Gaussian-type densities since there are two possible deterministic solutions, $y = \pm|F(t)|$, symmetric about $y = 0$.

Similarly, for the two-dimensional example of section 5, one can make the ansatz

$$p(u,v,t) \sim C\exp\left[-\frac{g(t)}{2\epsilon^2}(v-V(t))^2 - \frac{h(t)}{\epsilon^2}(u-U(t))(v-V(t)) - \frac{k(t)}{2\epsilon^2}(u-U(t))^2\right.$$
$$\left. + \frac{q(t)}{\epsilon}(v-V(t)) + \frac{r(t)}{\epsilon}(u-U(t)) + s(t)\right]. \tag{A.4}$$

Here $(U(t),V(t))$ is the deterministic solution to (5.1) with $\epsilon = 0$. Substitution of (A.4) into (5.3) yields a system of coupled ODEs for $g(t)$, $h(t)$, $k(t)$, $q(t)$, and $r(t)$, and $s(t)$, which depend on $U(t)$ and $V(t)$ and describe the time-dependent concentration of the density. This system can be solved numerically, and the solution is substituted into (A.4) to give an asymptotic approximation for $p(u,v,t)$.

Acknowledgment. The author would like to thank Christoph Börgers for many helpful discussions.

REFERENCES

[1] R. S. Maier and D. L. Stein, *Limiting exit location distributions in the stochastic exit problem*, SIAM J. Appl. Math, 57 (1997), pp. 752–790.

[2] J. J. Collins, C. C. Chow, A. C. Capela, and T. T. Imhoff, *Aperiodic stochastic resonance*, Phys. Rev. E, 54 (1996), pp. 5575–5584.

[3] J. Rinzel and S. M. Baer, *Threshold for repetitive activity for a slow stimulus ramp: a memory effect and its dependence on fluctuations*, Biophys. J., 54 (1988), pp. 551–555.

[4] D. Hughes and M. R. E. Proctor, *Chaos and the effect of noise in a model of three-wave mode coupling*, Phys. D, 46 (1990), pp. 163–176.

[5] J. C. Celet, D. Dangoisse, P. Glorieux, G. Lythe, and T. Erneux, *Slowly passing through resonance strongly depends on noise*, Phys. Rev. Lett., 81 (1998), pp. 975–978.

[6] Z. Schuss, *Theory and Applications of Stochastic Differential Equations*, John Wiley, New York, 1980.

[7] R. Kuske, *Probability density functions for noisy delay bifurcations*, J. Statist. Phys., 96 (1999), pp. 787–816.

[8] A. L. Fogelson, *Particle-method solutions of two-dimensional convection-diffusion equations*, J. Comput. Phys., 100 (1992), pp. 1–16.

[9] A. Sherman and M. Mascagni, *A gradient random walk method for two-dimensional reaction-diffusion equations*, SIAM J. Sci. Comput., 15 (1994), pp. 1280–1293.

[10] P. Mandel and T. Erneux, *The slow passage through a steady bifurcation: Delay and memory effects*, J. Statist. Phys., 48 (1987), pp. 1059–1070.

[11] G. D. Lythe, *Domain formation in transitions with noise and a time dependent bifurcation parameter*, Phys. Rev. E, 53 (1996), pp. R4271–R4274.

[12] P. Kloeden and E. Platen, *Numerical Solution of Stochastic Differential Equations*, Springer-Verlag, Berlin, 1992.

[13] A. S. Sherman and C. S. Peskin, *Solving the Hodgkin–Huxley equations by a random walk method*, SIAM J. Sci. Statist. Comput., 9 (1988), pp. 170–190.

[14] A. J. Chorin, *Numerical study of slightly viscous flow*, J. Fluid Mech., 57 (1973), pp. 785–796.

[15] C. R. Anderson, *A vortex method for flows with slight density variation*, J. Comput. Phys., 61 (1985), pp. 417–444.

[16] L. Greengard and V. Rokhlin, *A fast algorithm for particle simulations*, J. Comput. Phys., 73 (1987), pp. 325–348.

[17] *FISHPACK* is a set of FORTRAN subprograms for solving elliptic boundary value problems with finite differences; available online at http://www.netlib.org.

[18] S. M. Baer, T. Erneux, and J. Rinzel, *The slow passage through a Hopf bifurcation: Delay, memory effects, and resonance*, SIAM J. Appl. Math, 49 (1989), pp. 55–71.

SIAM J. SCI. COMPUT.
Vol. 22, No. 1, pp. 368–385
© 2000 Society for Industrial and Applied Mathematics

ASYMPTOTIC-NUMERICAL STUDY OF SUPERSENSITIVITY FOR GENERALIZED BURGERS' EQUATIONS*

MARC GARBEY† AND HANS G. KAPER‡

Abstract. This article addresses some asymptotic and numerical issues related to the solution of Burgers' equation, $-\varepsilon u_{xx}+u_t+uu_x = 0$ on $(-1,1)$, subject to the boundary conditions $u(-1) = 1+\delta$, $u(1) = -1$, and its generalization to two dimensions, $-\varepsilon\Delta u+u_t+uu_x+uu_y = 0$ on $(-1,1)\times(-\pi,\pi)$, subject to the boundary conditions $u|_{x=1} = 1+\delta$, $u|_{x=-1} = -1$, with 2π periodicity in y. The perturbation parameters δ and ε are arbitrarily small positive and independent; when they approach 0, they satisfy the asymptotic order relation $\delta = O_s(\mathrm{e}^{-a/\varepsilon})$ for some constant $a \in (0,1)$.

The solutions of these convection-dominated viscous conservation laws exhibit a transition layer in the interior of the domain, whose position as $t \to \infty$ is supersensitive to the boundary perturbation. Algorithms are presented for the computation of the position of the transition layer at steady state. The algorithms generalize to viscous conservation laws with a convex nonlinearity and are scalable in a parallel computing environment.

AMS subject classifications. Primary, 35B25, 35B30; Secondary, 35Q53, 65M55

Key words. asymptotic analysis, domain decomposition, Burgers' equation, viscous conservation laws, transition layers, supersensitivity

PII. S1064827598342201

1. Introduction.

In this article we address some asymptotic and numerical issues related to the solution of Burgers' equation,

$$-\varepsilon u_{xx} + u_t + uu_x = 0 \quad \text{on } (-1,1), \quad u(-1) = 1+\delta, \quad u(1) = -1, \tag{1.1}$$

and its generalization to two dimensions,

$$-\varepsilon\Delta u + u_t + uu_x + \beta uu_y = 0 \quad \text{on } (-1,1)\times(-\pi,\pi), \quad u|_{x=-1} = 1+\delta, \quad u|_{x=1} = -1. \tag{1.2}$$

In the latter case, we assume periodicity (period 2π) in the second coordinate (y). The perturbation parameters δ and ε are arbitrarily small positive; they are independent, but when they approach 0, they satisfy the asymptotic order relation

$$\delta = O_s(\mathrm{e}^{-a/\varepsilon}) \quad \text{as } \delta, \varepsilon \downarrow 0 \tag{1.3}$$

for some constant $a \in (0,1)$ which does not depend on δ or ε. This asymptotic relation implies that δ is transcendentally small (in the sense of asymptotic analysis) compared

*Received by the editors July 20, 1998; accepted for publication (in revised form) August 17, 1999; published electronically June 22, 2000. The U.S. Government retains a nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or allow others to do so, for U.S. Government purposes. Copyright is owned by SIAM to the extent not limited by these rights.

http://www.siam.org/journals/sisc/22-1/34220.html

†CDCSP-ISTIL, Université Claude Bernard Lyon 1, 69622 Villeurbanne cedex, France (garbey@cdcsp.univ-lyon1.fr). This author was supported by the Fondation Cromey-le-Bas under contract BS-95-2.

‡Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL 60439-4844 (kaper@mcs.anl.gov). This author was supported by the Mathematical, Information, and Computational Sciences Division subprogram of the Office of Advanced Scientific Computing Research, U.S. Department of Energy, Contract W-31-109-Eng-38.

with ε, but δ dominates $e^{-1/\varepsilon}$ as $\varepsilon \downarrow 0$. (See [1, 2, 3, 4, 5] for definitions and basic concepts of asymptotic analysis.)

If $\varepsilon = 0$, the solution of (1.1) develops a shock (discontinuity) in finite time, even when the initial data are smooth [6, 7]. The perturbation introduced by a nonzero ε models the presence of viscosity, which tends to smooth the discontinuity [8, 9]. Instead of a shock, one has a transition layer—a region of rapid variation, which extends over a distance $O(\varepsilon)$ as $\varepsilon \downarrow 0$. The position of the transition layer varies with time, and its eventual location at steady state is extremely sensitive to the boundary data. In fact, even the transcendentally small perturbation δ leads to a measurable (that is, order one) effect on the eventual location of the transition layer. This phenomenon, known as *supersensitivity*, was first observed by Lorentz [10]. It has been studied extensively for Burgers' equation and more general viscous conservation laws in one dimension by Kreiss and Kreiss [11], Kreiss [12], Laforgue and O'Malley [13, 14, 15, 16, 17, 18], and Reyna and Ward [19, 20, 21].

An example from combustion theory shows that supersensitivity is of more than mathematical significance. A simple model of flame propagation in gaseous fuels involves a system of two coupled convection-diffusion equations, one for the temperature of the mixture, another for the concentration of the reaction-limiting component in the mixture [22, section 3.2]. If one ignores exponentially small perturbations in the data, one finds that the Lewis number $\mathcal{L}$, which is a measure for the ratio of heat and mass transfer in the mixture, has no effect on the location of the combustion front. Yet, numerical computations show that this location is very sensitive—in fact, supersensitive—to the value of $\mathcal{L}$.

Although the phenomenon of supersensitivity is fairly well understood theoretically, at least for one-dimensional problems, the numerical solution of such problems still poses formidable challenges, especially in more than one dimension. The methods that have been proposed in the numerical literature for singularly perturbed boundary-value problems (see, for example, [23]) tend to focus on uniform approximations or finite elements with special features, not on the supersensitive dependence of the transition layer on the boundary data. On the other hand, the algorithms we propose are designed specifically to capture the phenomenon of supersensitivity. They use the fact that the solution approaches a certain profile as the perturbation parameters approach zero and focus on the computation of the corrections.

Our ultimate goal is to develop algorithms for multidimensional problems that are, first of all, suitable for long-time integration, so stable steady states can be computed with confidence; second, extremely accurate in space, so the eventual location of transition layers can be predicted with accuracy; and third, scalable in a multiprocessing environment, so large-scale problems can be solved in a reasonable length of time. Although we discuss only Burgers' equation and its generalization to two dimensions, the algorithms are not restricted by the special form of the nonlinearity.

In section 2 we consider Burgers' equation. We propose a simple algorithm that effectively captures the supersensitive location of the transition layer at steady state. We stress the importance of the regions outside the transition layer, where the solution does not yet deviate appreciably from the boundary values. In section 3 we address the generalized Burgers' equation in two dimensions. We show through a formal asymptotic analysis that the location of the transition layer may vary in the direction of periodicity (y), but the transition layer is essentially flat, and only its average position (averaged over y) depends supersensitively on the small parameters. We then develop an algorithm that effectively approximates the transition layer.

2. One-dimensional case. We begin by considering (1.1),

$$-\varepsilon u_{xx} + u_t + uu_x = 0 \quad \text{on } (-1,1), \quad u(-1) = 1+\delta, \quad u(1) = -1. \tag{2.1}$$

As shown by Laforgue and O'Malley [16], the solution approaches a certain profile function as $\varepsilon \downarrow 0$,

$$u(x,t) = -\tanh\eta + \mathrm{e}^{-a/\varepsilon} u_1(\eta,\sigma) + \cdots, \tag{2.2}$$

where

$$\eta = \frac{x - x^*(\sigma)}{\varepsilon}, \quad \sigma = t\mathrm{e}^{-a/\varepsilon}. \tag{2.3}$$

The hyperbolic tangent incorporates a transition layer centered at x^*, which connects the limiting values ± 1 at $\mp\infty$. Note that these limiting values are transcendentally close to the prescribed boundary values $1+\delta$ and -1 of u at -1 and 1. The position of the transition layer varies on a transcendentally slow time scale; if $\delta = 2b\mathrm{e}^{-a/\varepsilon}$, its limit as $\sigma \to \infty$ is

$$x^*_{\mathrm{as}} = 1 - a + \varepsilon \ln b. \tag{2.4}$$

The important points to observe are that, first, the solution u approaches a certain profile function as $\varepsilon \downarrow 0$; second, the accurate determination of the steady-state position of the transition layer requires long-term integration; and, third, a transcendentally small perturbation of the boundary data has a measurable effect on the location of the transition layer.

The asymptotic analysis has been generalized to more general nonlinearities by Laforgue and O'Malley [17, 18] and Reyna and Ward [19], with similar results. One finds a limiting profile, which generalizes the hyperbolic tangent function, and a transition layer which moves on a transcendentally slow time scale to a steady-state position. This position depends supersensitively on the boundary perturbation. Whenever this situation arises, appropriate variants of the following algorithms can be developed.

2.1. Spatial approximation. To approximate the solution in space, we use a domain decomposition method with two nonoverlapping subdomains, where the interface is located approximately at the center of the transition layer; an adaptive pseudospectral method on each subdomain; and collocation based on Tchebychev polynomials, where the collocation points are concentrated in the transition layer. The algorithm is standard and has been described elsewhere [24, 25, 26, 27, 28]; we summarize it here only for completeness.

Let $x^* \in (-1,1)$ denote the (approximate) position of the center of the transition layer; x^* varies in time (t), but since t enters only as a parameter in the discussion of the spatial approximation, we do not write it explicitly. We decompose,

$$\Omega_1 = (-1, x^*), \quad \Omega_2 = (x^*, 1), \tag{2.5}$$

and map each of the subdomains Ω_1 and Ω_2 linearly onto $(-1,1)$,

$$g_1 : y \in (-1,1) \mapsto x = g_1(y) = -1 + \tfrac{1}{2}(x^*+1)(y+1) \in \Omega_1,$$

$$g_2 : y \in (-1,1) \mapsto x = g_2(y) = 1 - \tfrac{1}{2}(1-x^*)(1-y) \in \Omega_2.$$

The restrictions of u to Ω_1 and Ω_2 exhibit boundary layer behavior near x^*. The point $x = x^*$ corresponds to $y = 1$ under g_1 and to $y = -1$ under g_2. To concentrate the collocation points near x^*, we define a one-parameter family of nonlinear mappings of the interval $(-1, 1)$ onto itself,

$$f_1(\cdot\,,\alpha) : s \in (-1,1) \mapsto y = f_1(s,\alpha) = 1 - (4/\pi)\arctan\left(\alpha \tan \tfrac{1}{4}(1-s)\pi\right) \in (-1,1),$$

$$f_2(\cdot\,,\alpha) : s \in (-1,1) \mapsto y = f_2(s,\alpha) = -1 + (4/\pi)\arctan\left(\alpha \tan \tfrac{1}{4}(s+1)\pi\right) \in (-1,1).$$

If the parameter α is small, f_1 concentrates points near 1 and f_2 concentrates points near -1. Concentrating points near critical points is the computational analog of coordinate stretching in asymptotic analysis. The choice of α can be optimized by means of a priori estimates [24, 25]; we usually take $\alpha = \varepsilon^{1/2}$ [27]. The composite maps,

$$h_i(\cdot\,;\alpha) = g_i(\cdot\,) \circ f_i(\cdot\,,\alpha), \quad i = 1, 2,$$

are one-to-one from $(-1, 1)$ onto Ω_i; we denote their inverses by $h_i^{-1}(\cdot\,;\alpha)$, $i = 1, 2$.

We look for the solution of (2.1) by approximating locally on each of the subdomains Ω_1 and Ω_2 and imposing C^1 continuity at x^*. If U denotes the global approximation, then

$$U(x) = U_i(x) \quad \text{for } x \in \Omega_i, \quad U \in C^1([-1,1]).$$

The local approximations consist of finite sums of Tchebychev polynomials,

$$U_i(x) = \sum_{j=0}^{N-1} a_{ij} T_j(h_i^{-1}(x;\alpha)), \quad x \in \Omega_i,\ i = 1,2; \quad T_j(\cos\theta) = \cos(j\theta), \quad \theta = \pi/N.$$

2.2. Integration for times of order one. For short-time integration it suffices to combine a one-step forward Euler approximation with an implicit treatment of the second-order spatial derivative and an explicit treatment of the nonlinear term.

Starting with an approximate solution $U^0 = U(\cdot\,, t_0)$ at time t_0, we identify the point $x^* = x^*(t_0)$ with the location of the zero of U^0, partition the domain in two subdomains, and select the collocation points. Fixing this configuration temporarily, we compute a sequence $\{U^n : n = 1, 2 \ldots\}$ of successive approximations U^n using the algorithm

$$-\varepsilon D^2 U^n + \frac{U^n - U^{n-1}}{\Delta t} + U^{n-1} D U^{n-1} = 0, \quad n = 1, 2, \ldots. \tag{2.6}$$

The symbol D represents the pseudospectral differentiation operator in physical space [29]. The time step Δt is constant, so U^n is the approximate solution of the boundary-value problem (1.1) at $t_0 + n\Delta t$. The algortithm (2.6) is nonconservative.

When the location of the zero of U^n has moved over a distance ε, we suspend the algorithm (2.6). We shift x^* to the current location of the zero, reconfigure the partition, update the collocation points, replace U^0 by the values at the new collocation points (using interpolation if necessary), and continue the algorithm (2.6). We repeat this process until the steady state is reached. The change in the location of the zero of the computed approximation becomes smaller as time progresses, so the same collocation configuration serves for longer and longer time intervals.

TABLE 1
Location of the transition layer at steady state.

	$\varepsilon = 0.1$		$\varepsilon = 0.05$	
δ	x^*_∞	$x^*_{\rm as}$	x^*_∞	$x^*_{\rm as}$
$1.0 \cdot 10^{-1}$	0.72464	0.700427	0.86237	0.850213
$1.0 \cdot 10^{-2}$	0.47486	0.470176	0.73755	0.735084
$1.0 \cdot 10^{-3}$	0.24133	0.240724	0.62055	0.619955
$1.0 \cdot 10^{-4}$	0.05265	0.052606	0.50485	0.504826
$1.0 \cdot 10^{-5}$	0.00537	0.005504	0.38962	0.389696

TABLE 2
Effect of grid refinement (N) on x^*_∞; $\varepsilon = 0.1$, $\delta = 1.0 \cdot 10^{-3}$.

N	15	19	29	39	49	59
x^*_∞	0.25576	0.24115	0.24166	0.24133	0.24140	0.24143

The algorithm (2.6) requires the solution of U^n from the equation

$$AU^n = U^{n-1} + (\Delta t)U^{n-1}DU^{n-1}. \tag{2.7}$$

The matrix A, which is order $2N-1$, has a block structure,

$$A = \begin{pmatrix} A_1 & b_1 & \\ a_1^{\rm t} & c & b_2^{\rm t} \\ & a_2 & A_2 \end{pmatrix}.$$

A_1 and A_2 are square matrices of order $N-1$; a_1, a_2, b_1, and b_2 vectors of length $N-1$; c is a constant. The center row accounts for the C^1 continuity at the interface. This block structure allows a solution of the system (2.7) in two parallel processes from opposite ends. The matrix A does not change as long as the collocation configuration is frozen. However, its condition number increases with the number of collocation points. This increase puts a lower limit on the values of δ one can handle in practice.

Table 1 gives the location of the transition layer at steady state, x^*_∞, computed with the algorithm (2.6) with $N = 39$ collocation points in each subdomain and a time step $\Delta t = 0.02$. The number $x^*_{\rm as} = 1 - \varepsilon \ln(2/\delta)$, which is an asymptotic estimate of x^* (see (2.4)), is given for comparison. The initial conditions were usually obtained by linear interpolation from the boundary data, but variations were made to test the answers. The lower limit on ε is determined by the fact that the computation time for the algorithm (2.6) increases as ε decreases. In subsection 2.4 we discuss an algorithm suitable for long-time integration.

Table 2 shows the impact of grid refinement (the number of collocation points, N) on the value of x^*_∞.

2.3. Neglecting viscosity. In supersensitive boundary-value problems, the solution in the "tails" on either side of the transition layer is exponentially close to the prescribed boundary values, so the viscous term is exponentially small there. It is therefore tempting to assume that one can sacrifice some accuracy in the computation of the viscous term during the transient phase and still find the position of the transition layer at steady state with a high degree of accuracy.

An extreme form of this assumption underlies the approach where one constructs a first approximation by ignoring the viscous term altogether. Using a conservative

finite-difference scheme, such as Godunov, one constructs the entropy solution of the inviscid conservation law ($\varepsilon = 0$) and takes this as a first approximation. One then constructs higher order uniform approximations, for example, by means of a heterogeneous domain-decomposition method, using either a different numerical scheme to solve the full viscid problem in the interior of the transition layer or some other approximation of the viscous equation.

The χ method introduced by Brezzi, Canuto, and Russo [30] is a more sophisticated nonlinear adaptive scheme based on the same assumption. Here, one replaces the boundary-value problem by

$$-\varepsilon\chi(u_{xx}) + u_t + (f(u))_x = 0 \text{ on } (-1,1), \quad u(-1,t) = 1+\delta,\ u(1,t) = -1, \tag{2.8}$$

where $\chi \equiv \chi_{\sigma,\tau}$ is a smooth monotone function, $\chi(s) = 0$ if $|s| \le \sigma$, and $\chi(s) = s$ if $|s| \ge \sigma + \tau$ for some positive numbers σ and τ. This method has been applied to Burgers' equation [31] and the incompressible Navier–Stokes equations [32]. However, we claim that the χ method cannot accurately predict the ultimate position of the transition layer, at least for Burgers' equation on a finite interval with Dirichlet data. This claim is supported by the following observations.

Consider the results quoted in [31, Table II]. With few exceptions, they involve relatively large values of σ (σ is called δ in [31]); in fact, σ is typically greater than $\varepsilon^{-1/2}$ (ε is called ν in [31]) by one or two orders of magnitude. The viscous term is therefore always neglected, unless u_{xx} is of the same order as $\varepsilon^{-1/2}$; that is, the viscous term is neglected everywhere except in the transition layer. If the χ method gave the correct position of the transition layer at steady state, then the same would certainly be the case when we simply multiply the viscous term by a smooth function of position, whose support is of order one and includes the transition layer. After all, in the latter case we account for the viscous term over a much broader region. These arguments lead us to consider the boundary-value problem

$$-\varepsilon H(x)u_{xx} + u_t + uu_x = 0 \text{ on } (-1,1), \quad u(-1,t) = 1+\delta,\ u(1,t) = -1, \tag{2.9}$$

instead of the boundary-value problem (2.8). Here, H is a smooth cutoff function,

$$H(x) = \begin{cases} \frac{1}{2}\left(1 + \tanh(\alpha(x - x^*(t) + \beta))\right), & x < x^*(t), \\ \frac{1}{2}\left(1 + \tanh(\alpha(-x + x^*(t) + \beta))\right), & x > x^*(t). \end{cases} \tag{2.10}$$

We use a numerical approximation of $x^*(t)$ and choose the parameters α and β so

$$H(x) = 1 \text{ if } |x - x^*(t)| < \tfrac{1}{2}\beta, \quad H(x) = 0 \text{ if } |x - x^*(t)| > \tfrac{3}{2}\beta$$

to within machine accuracy ($1 \cdot 10^{-15}$). The algorithm (2.6) leads to the solution of U^n from the equation

$$-\varepsilon H D^2 U^n + \frac{U^n - U^{n-1}}{\Delta t} + U^{n-1} D u^{n-1} = 0, \quad n = 1, 2, \ldots. \tag{2.11}$$

The results given in Table 3 (computed for $\varepsilon = 0.1$ and $\delta = 1.0 \cdot 10^{-2}$, with $\alpha = 200$) show that the algorithm (2.11) can give incorrect results for the position of the transition layer at steady state. (The correct value is $x^*_\infty = 0.47486$; see Table 1.) The position of the shock freezes too early during the transient phase. The actual moment of freezing depends on the size of the zone to the left of the transition layer (where $H \equiv 0$). The result improves as β increases, but for $\beta = 0.8$ the algorithm

TABLE 3
Location of the transition layer at steady state computed with the χ method.

β	0.1	0.2	0.3	0.4	0.5	0.6	0.7
x^*_∞	0.0056	0.0110	0.0174	0.0384	0.0923	0.1067	0.1565

fails to converge. (The position of the transition layer keeps oscillating between the regions where $H \equiv 1$ and $H \equiv 0$.) On the other hand, if we include the missing part of the viscous term explicitly and use the algorithm

(2.12)
$$-\varepsilon H D^2 U^n - \varepsilon(1-H)D^2 U^{n-1} + \frac{U^n - U^{n-1}}{\Delta t} + U^{n-1} D u^{n-1} = 0, \quad n = 1, 2, \ldots,$$

instead of (2.11), we retrieve the correct position of the transition layer at steady state. This result demonstrates clearly that, when the problem is supersensitive, it is not advisable to neglect the viscous term, even when that term is exponentially small.

Note that the modified algorithm (2.13) treats the second-order derivative explicitly in the region where $H \equiv 0$ and implicitly in the region where $H \equiv 1$. The idea of using a cutoff function to construct a composite algorithm is described in detail in our article [28]. The procedure offers a very general tool for the design of heterogeneous domain decompositions in the framework of a finite-difference approximation. However, since the algorithm (2.13) is based on a Tchebyshev pseudospectral approximation, the partially explicit treatment of the viscous term forces a severe constraint on the time step that cannot be circumvented. For example, with $\varepsilon = 0.1$ and $N = 49$ collocation points per subdomain, the time step must be 10 times smaller than the time step for the algorithm (2.6). The algorithm (2.13) is therefore certainly not practical for long-time integration.

2.4. Long-time integration. The explicit treatment of the nonlinear term in the algorithm (2.6) imposes a severe constraint on the time step (CFL condition). The algorithm is therefore not suitable for long-time integration. The alternative approach commonly taken is to use a fully implicit scheme in combination with a Newton algorithm [33]. However, an implicit scheme can be expensive and is certain to increase interprocessor communication in a multiprocessing environment. If the solution of the viscous conservation law is close to a certain profile function, as is the case for Burgers' equation, at least after an initial transient, the following algorithm offers a more efficient alternative.

We start the integration of (2.1) at $t = t_0$, say, when we have an approximate profile with a transition layer centered at x^*. We construct the function u_0,

$$u_0(x) = -\tanh \frac{x - x^*}{2\varepsilon}, \tag{2.13}$$

which satifies Burgers' equation exactly, and look for a solution u of the form

$$u(x,t) = u_0(x) + \delta v(x,t). \tag{2.14}$$

Then v must satisfy the nonlinear boundary-value problem

$$-\varepsilon v_{xx} + v_t + u_0 v_x + u_0' v + \delta v v_x = 0 \quad \text{on } (-1,1),$$

$$v(-1,t) = \delta^{-1}(1 + \delta - u_0(-1)), \quad v(1,t) = \delta^{-1}(-1 - u_0(1)).$$

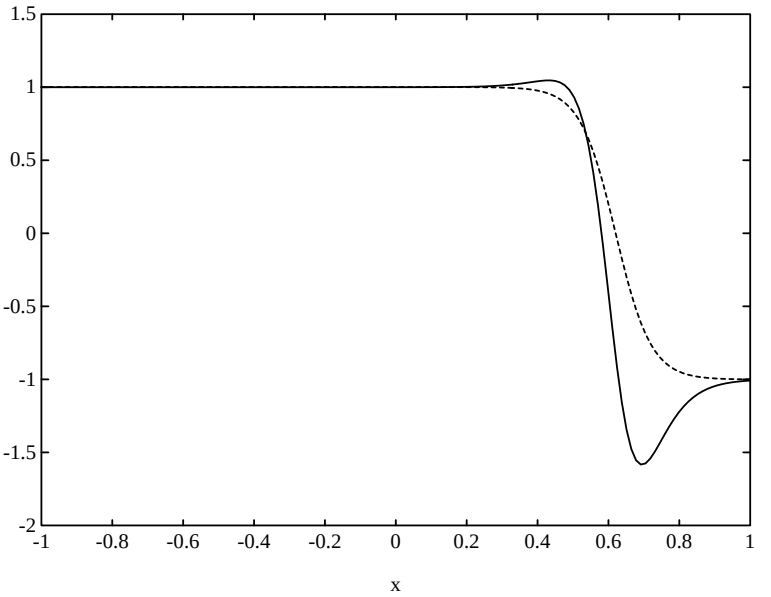


FIG. 1. *The profile function u_0 (dashed line) and the computed solution $U(\cdot\,,t)$ at some time t (solid line); $\varepsilon = 0.05$, $\delta = 1 \cdot 10^{-3}$.*

We integrate this boundary-value problem for $t > t_0$, using the algorithm

(2.15)
$$-\varepsilon D^2 V^n + \frac{V^n - V^{n-1}}{\Delta t} + u_0 DV^n + u_0' V^n = -\delta V^{n-1} DV^{n-1}, \quad n = 1, 2, \ldots,$$

and define an approximation U of u for $t > t_0$,

(2.16)
$$U(x,t) = u_0(x) + \delta V(x,t), \quad t > t_0.$$

We proceed with the integration as long as the supremum of $U(\cdot\,,t) - u_0$ remains of the order of δ. When this criterion is no longer met, at $t = t_1$, say, we suspend the integration, identify the point x^* with the location of the center of the transition layer at t_1, update the profile function u_0, and continue the integration beyond t_1. We repeat the procedure until the steady state is reached. Figure 1 shows a profile function u_0 and the computed solution $U(\cdot\,,t)$ at some time t. Note that the former is monotone, the latter is not.

The algorithm (2.16) is very similar to (2.6). The spatial approximation is handled with the same domain-decomposition method, so the structure of the resulting linear system is the same as in (2.7). But the constraint on the time step is relaxed by a factor δ. With the algorithm (2.16) we can integrate the boundary-value problem for values of ε as small as 0.01, down to δs of the order of 10^{-6}, using time steps that are typically 50 times larger than with the algorithm (2.6). Table 4 gives some results for x_∞^*, computed with the algorithm (2.16), with $N = 39$ collocation points in each subdomain.

3. Two-dimensional case. Next, we consider Burgers' equation generalized to two dimensions,

(3.1)

TABLE 4
Location of the transition layer at steady state.

	$\varepsilon = 0.1$	$\varepsilon = 0.05$	$\varepsilon = 0.02$	$\varepsilon = 0.01$
δ	x^*_∞	x^*_∞	x^*_∞	x^*_∞
$1.0 \cdot 10^{-1}$	0.72346	0.86175		
$1.0 \cdot 10^{-2}$	0.47508	0.73774	0.89518	
$1.0 \cdot 10^{-3}$	0.24140	0.62057	0.84827	0.92429
$1.0 \cdot 10^{-4}$	0.05275	0.50485	0.80210	0.90104
$1.0 \cdot 10^{-5}$	0.00561	0.38964	0.75609	0.87800
$1.0 \cdot 10^{-6}$	0.00066	0.27452	0.70996	0.85482
$1.0 \cdot 10^{-7}$				0.83084

$$-\varepsilon\Delta u + u_t + uu_x + \beta u u_y = 0 \quad \text{on } (-1,1)\times(-\pi,\pi), \quad u|_{x=-1} = 1+\delta, \quad u|_{x=1} = -1.$$

We assume periodicity in y (period 2π). The perturbation δ may vary with y; its Fourier expansion is

$$\delta \equiv \delta(y) = \delta_0 \sum_{k\in\mathbf{Z}} \eta_k \mathrm{e}^{iky}. \tag{3.2}$$

The coefficients η_k, as well as the prefactor δ_0, are independent of y. The prefactor δ_0 is arbitrarily small positive and defined in such a way that $\eta_0 = 1$ and $\eta_k = O(1)$ as $\delta_0 \downarrow 0$ for $k = \pm 1, \pm 2, \ldots$. The order relation (1.3) between δ and ε, which must hold uniformly in y, implies that

$$\delta_0 = O_s(\mathrm{e}^{-a/\varepsilon}) \quad \text{as } \delta_0, \varepsilon \downarrow 0 \tag{3.3}$$

for some $a \in (0,1)$. We show in section 3.1 that, under these conditions, the *average* position of the transition layer (averaged over y) depends supersensitively on the small parameters ε and δ_0. In section 3.2 we briefly review the spatial approximation. Section 3.3 is devoted to the integration procedure.

3.1. Profile function. The algorithm we propose for the solution of the boundary-value problem (3.1) is again based on the assumption that the solution is close to a known profile function. Our goal in this section is to show that, under the conditions given above, the profile function is asymptotically independent of y and again given to leading order by the hyperbolic tangent, as in (2.2).

We introduce the constant $x^* \in (0,1)$ such that $x^* \sim 1 - \varepsilon \ln(2/\delta_0)$ as $\varepsilon \downarrow 0$. We define the function u_0,

$$u_0(x) = -\tanh \frac{x - x^*}{2\varepsilon}, \tag{3.4}$$

and look for a profile function u of the form

$$u(x,y) = u_0(x) + v(x,y). \tag{3.5}$$

Because $-\varepsilon u_0'' + u_0 u_0'$, v must satisfy the differential equation

$$-\varepsilon\Delta v + u_0 v_x + u_0' v + \beta u_0 v_y + v v_x + \beta v v_y = 0, \tag{3.6}$$

together with the boundary conditions

$$v|_{x=-1} = \delta - \delta_0, \quad v|_{x=1} = 0. \tag{3.7}$$

The linearized equation,

$$\ell(v) \equiv -\varepsilon\Delta v + u_0 v_x + u_0' v + \beta u_0 v_y = 0, \tag{3.8}$$

has a solution, $\ell(u_0') = 0$, so we reduce the order by substituting

$$v = u_0' w. \tag{3.9}$$

Then w must satisfy the equation

$$-\varepsilon\Delta w - u_0 w_x + \beta u_0 w_y = 0, \tag{3.10}$$

together with the boundary conditions

$$w|_{x=-1} = (\delta - \delta_0)/u_0'(-1), \quad w|_{x=1} = 0. \tag{3.11}$$

We estimate w by means of the coefficients in its Fourier expansion,

$$w(x,y) = \sum_{k\in\mathbf{Z}} w_k(x)\mathrm{e}^{iky}. \tag{3.12}$$

The leading coefficient w_0 is the solution of the boundary-value problem

$$-\varepsilon w_0'' - u_0 w_0' = 0 \quad \text{on } (-1,1), \quad w_0(-1) = 0, \quad w_0(1) = 0, \tag{3.13}$$

so $w_0 = 0$. Note that this result is a direct consequence of the fact that we have defined x^* in terms of the average δ_0 in (3.4); any other definition leads to an inhomogeneous boundary-value problem, whose solution w_0 does not vanish.

The remaining coefficients w_k, $k = \pm1, \pm2, \ldots$, are found from the boundary-value problem

$$-\varepsilon w_k'' - u_0 w_k' + (\varepsilon k^2 + i\beta k u_0) w_k = 0, \quad w_k(-1) = \delta_0\eta_k, \quad w_k(1) = 0. \tag{3.14}$$

This is a classical turning-point problem, as u_0 changes sign in the interval $(-1,1)$. The asymptotic behavior of w_k as $\varepsilon \downarrow 0$ can be found by the method described in [3, section 3.E],

$$w_k(x) \sim \left[c_k\mathrm{e}^{i\beta kx} + \left(1 - c_k\mathrm{e}^{-i\beta k}\right)\mathrm{e}^{-(1+x)/\varepsilon} + \left(0 - c_k\mathrm{e}^{i\beta k}\right)\mathrm{e}^{-(1-x)/\varepsilon}\right] w_k(-1). \tag{3.15}$$

The coefficient c_k is such that the functional

$$\mathcal{L}[w] = \frac{1}{2}\int_{-1}^{1} \left[\varepsilon w'^2 + (\varepsilon k^2 + i\beta k u_0) w^2\right] \exp\left(\frac{1}{\varepsilon}\int_0^x u_0(\xi)\,\mathrm{d}\xi\right)\,\mathrm{d}x \tag{3.16}$$

has a critical point at $w = w_k$,

$$\frac{\partial\mathcal{L}[w_k]}{\partial c_k} = 0. \tag{3.17}$$

Notice that the differential equation (3.14) is the Euler equation of the functional $\mathcal{L}$; the Neumann boundary conditions $w_k'(\pm1) = 0$ are the natural boundary conditions associated with $\mathcal{L}$.

Obviously, finding an explicit expression for c_k is out of the question. The best we can aim for is an asymptotic expansion as $\varepsilon \downarrow 0$, and even here we must resort to computational assistance. Using the symbolic manipulation language MAPLE, we find

$$c_k \sim \frac{\mathrm{e}^{-(x^*+1)/\varepsilon}}{(1+\beta^2)\varepsilon^2 k^2}\mathrm{e}^{-i\beta k(1+2x^*)} \quad \text{as } \varepsilon \downarrow 0. \tag{3.18}$$

The first term in the brackets in (3.16) represents the regular part of the asymptotic behavior of w_k, which dominates in the interior; the remaining two terms represent the singular part, which dominates near the endpoints of the interval. Since we are interested in the transition layer, which is located in the interior, we ignore the singular part and take only the regular part, $w_k(x) \sim c_k w_k(-1)\mathrm{e}^{i\beta kx}$. That is, we take

$$v_k(x) \sim \frac{\delta_0\eta_k \mathrm{e}^{-(1+x^*)/\varepsilon}}{(1+\beta^2)\varepsilon^2 k^2}\,\frac{u_0'(x)}{u_0'(-1)}\,\mathrm{e}^{i\beta k(1-2x^*+x)} \quad \text{as } \varepsilon \downarrow 0. \tag{3.19}$$

If we use the asymptotic approximation

$$\frac{u_0'(x)}{u_0'(-1)} = \frac{1-\tanh^2((x-x^*)/(2\varepsilon))}{1-\tanh^2((-1-x^*)/(2\varepsilon))} \sim \frac{1-\tanh^2((x-x^*)/(2\varepsilon))}{4\mathrm{e}^{-(1+x^*)/\varepsilon}},$$

we obtain the asymptotic expression

$$v_k(x) \sim \frac{\delta_0\eta_k}{4(1+\beta^2)\varepsilon^2 k^2}\,\mathrm{e}^{i\beta k(1-2x^*+x)}\left(1-\tanh^2\frac{x-x^*}{2\varepsilon}\right). \tag{3.20}$$

This result implies that the Fourier series of v, as well as those of v_x and v_y, converge. Furthermore, $\|v\|_\infty$ and $\|v_y\|_\infty$ are $O(\delta_0\varepsilon^{-2})$.

Finding the asymptotic behavior of $\|v_x\|_\infty$ is less obvious. It follows from (3.16) that $w_k'(x) \sim c_k\mathrm{e}^{i\beta kx}$ as $\varepsilon \downarrow 0$, at least for x in the interior of the domain. Therefore, $v_k'(x) = (u_0'w_k)'(x) \sim (u_0''w_k)(x) \sim \varepsilon^{-1}(u_0'w_k)(x) = \varepsilon^{-1}v_k(x)$. Hence, $\|v_x\|_\infty = O(\delta_0\varepsilon^{-3})$ as $\delta_0, \varepsilon \downarrow 0$.

Since $\delta_0 = O_s(\mathrm{e}^{-a/\varepsilon})$ for some $a \in (0,1)$, we have $\delta_0\varepsilon^{-p} = O_s(\mathrm{e}^{-a'/\varepsilon})$ $(p = 2,3)$ for any $a' \in (0,a)$, so any solution v of (3.8) which satisfies the boundary conditions (3.7) is transcendentally small. The residue $vv_x + \beta v v_y$, which was ignored in the transition from the nonlinear equation (3.6) to the linear equation (3.8) is likewise transcendentally small and, in fact, $O(\delta_0^2\varepsilon^{-5})$, so we also have an a posteriori justification for the linearization.

These arguments motivate the choice of u_0, which depends only on x, as the profile function in the design of the numerical algorithms for (3.2).

3.2. Spatial approximation. The spatial approximation is again based on a domain decomposition with two nonoverlapping subdomains on either side of the y-averaged location of the center of the transition layer. On each subdomain we use an adaptive pseudospectral method in the x direction and a finite-difference method in the y direction. The pseudospectral method is the same as in the one-dimensional case; it uses Tchebychev polynomial collocation with N_x collocation points.

Since the transition layer is close to a plane parallel to the x axis, there is no need to resort to an adaptive grid in the y direction. For our numerical experiments we chose a regular grid with mesh width $h = 2\pi/N_y$ and a sixth-order central finite-difference approximation of u_{yy} and u_y. The choice may seem inconsistent with the

spectral approximation in the x direction; a more obvious choice would be a Fourier approximation in the y direction. Theoretically, the finite-difference approximation in the y direction restricts the accuracy of the approximation for a regular problem ($\varepsilon = O_s(1)$) to sixth order, less than the accuracy guaranteed by the pseudospectral approximation in the x direction. However, as the transition layer is close to a plane parallel to the x axis, it is relatively easy to keep the numerical error in the finite-difference approximation of the term εu_{yy} smaller than the numerical error in the pseudospectral approximation of the term εu_{xx} with a moderate number of discretization points N_y. There is, therefore, no need to use a better approximation, like the Fourier approximation, for the term εu_{yy}. Furthermore, the spectral radius of D_y^2 is smaller with sixth-order finite differences than with Fourier differentiation. This difference implies an additional advantage for a finite-difference approximation when the y derivatives are treated explicitly [34].

3.3. Integration for times of order one. We extend the Euler scheme (2.6) to two dimensions as follows:

(3.21)
$$-\varepsilon D_x^2 U^n + \frac{U^n - U^{n-1}}{\Delta t} = \varepsilon D_y^2 U^{n-1} + U^{n-1} D_x U^{n-1} + \beta U^{n-1} D_y U^{n-1}, \quad n = 1, 2, \ldots.$$

Here, D_x is the pseudospectral differential operator with Tchebychev polynomials, D_y the finite-difference operator with sixth-order central finite differences.

The algorithm (3.21) has several features that make it readily parallelizable. First, the approximations $D_y U^{n-1}$ and $D_y^2 U^{n-1}$ are taken explicitly, so the variable y is only a parameter. Second, because we are using finite-difference approximations, we have only local data dependencies. This latter point especially offers a significant advantage over a spectral method, which uses global interpolation.

Table 5 shows the results for the boundary-value problem (3.1) with $\beta = 1$, $\varepsilon = 0.1$ and piecewise constant boundary data with $\delta_0 = 1.0 \cdot 10^{-2}$,

(3.22)
$$u(-1, y) = \begin{cases} 1.01 - \Delta\delta & \text{if } -\pi \le y < -\frac{1}{2}\pi, \\ 1.01 + \Delta\delta & \text{if } -\frac{1}{2}\pi \le y < \frac{1}{2}\pi, \\ 1.01 - \Delta\delta & \text{if } \frac{1}{2}\pi \le y < \pi. \end{cases}$$

The algorithm (3.21) was applied with $N_x = 39$ collocation points per subdomain in the x direction and $N_y = 32$ interpolation points in the y direction. The table gives the y-averaged location of the transition layer at steady state, $\langle x_\infty^* \rangle$, as well as the maximum deviation of the center of the transition layer from its y-averaged location, Δx^*; that is, $x_\infty^*(y)$ varies between $\langle x_\infty^* \rangle - \Delta x^*$ and $\langle x_\infty^* \rangle + \Delta x^*$. These results show that the algorithm (3.21) is extremely effective for the boundary-value problem. As $\Delta\delta$ increases, Δx^* grows approximately linearly with $\Delta\delta$. The graph of $U - u_0$ maintains its overall shape, so its width (which measures Δx^*) varies in proportion to its height (which measures $\|U - u_0\|_\infty$). The numerical results therefore indicate also that $\|U - u_0\|_\infty$ grows approximately linearly with $\Delta\delta$. This conclusion matches the results of the asymptotic analysis in section 3.1, in particular (3.20), where it was shown that v_k is proportional to η_k.

Results for a much harder case are presented in Table 6. The parameters β and ε are fixed as before, $\beta = 1$ and $\varepsilon = 0.1$, but this time the data at the left boundary are sharply peaked at the midpoint,

(3.23)
$$u(-1, y) = 1.005 + (\Delta\delta) \mathrm{e}^{-20(1-\cos y)}, \quad -\pi < y < \pi.$$

TABLE 5
Location of the transition layer at steady state; boundary data (3.22).

$\Delta\delta$	$\langle x^*_\infty \rangle$	Δx^*
$0.25 \cdot 10^{-2}$	0.4758	$1.3114 \cdot 10^{-2}$
$0.50 \cdot 10^{-2}$	0.4759	$2.3584 \cdot 10^{-2}$
$1.0 \cdot 10^{-2}$	0.4758	$4.8865 \cdot 10^{-2}$
$1.5 \cdot 10^{-2}$	0.4750	$7.8632 \cdot 10^{-2}$
$2.0 \cdot 10^{-2}$	0.4738	$9.3250 \cdot 10^{-2}$
$3.0 \cdot 10^{-2}$	0.4700	$15.688 \cdot 10^{-2}$

TABLE 6
Location of the transition layer at steady state; boundary data (3.23).

$\Delta\delta$	$\langle x^*_\infty \rangle$	x^*_{as}	Δx^*
$0.25 \cdot 10^{-2}$	0.40808	0.40811	$0.27278 \cdot 10^{-2}$
$0.50 \cdot 10^{-2}$	0.41277	0.41241	$0.37231 \cdot 10^{-2}$
$1.0 \cdot 10^{-2}$	0.42096	0.42052	$0.92263 \cdot 10^{-2}$
$2.0 \cdot 10^{-2}$	0.43573	0.43506	$1.4589 \cdot 10^{-2}$
$3.0 \cdot 10^{-2}$	0.44854	0.44782	$2.3924 \cdot 10^{-2}$

TABLE 7
Effect of grid refinement (N_x, N_y) on $\langle x^*_\infty \rangle$ (upper entries) and Δx^* (lower entries); boundary data (3.22) *with* $\Delta\delta = 0.01$.

N_y	$N_x = 19$	$N_x = 29$	$N_x = 39$	$N_x = 49$	$N_x = 59$
8	0.47848	0.47653	0.47523	0.47557	0.47547
	$4.7396\ 10^{-2}$	$4.8375\ 10^{-2}$	$4.8902\ 10^{-2}$	$5.2615\ 10^{-2}$	$5.2251\ 10^{-2}$
16	0.47807	0.47614	0.47610	0.47478	0.47516
	$4.7396\ 10^{-2}$	$5.8062\ 10^{-2}$	$4.8849\ 10^{-2}$	$4.9273\ 10^{-2}$	$4.9518\ 10^{-2}$
32	0.47883	0.47611	0.47578	0.47536	0.47481
	$4.7396\ 10^{-2}$	$4.8374\ 10^{-2}$	$4.8864\ 10^{-2}$	$4.9267\ 10^{-2}$	$4.9516\ 10^{-2}$
64	0.47847	0.47610	0.47561	0.47513	0.47524
	$4.7396\ 10^{-2}$	$4.8372\ 10^{-2}$	$4.8867\ 10^{-2}$	$4.9266\ 10^{-2}$	$5.2280\ 10^{-2}$

The algorithm (3.21) was again applied with $N_x = 39$ collocation points per subdomain in the x direction and $N_y = 32$ interpolation points in the y direction. The table gives, in addition to the values of $\langle x^*_\infty \rangle$ and Δx^*, the asymptotic value $x^*_{\text{as}} = 1 - \varepsilon \ln(2/\delta_0)$. The average location of the transition layer is predicted very well by the asymptotics. Again, the maximum deviation Δx^* grows with $\Delta\delta$, although not linearly as in the case of the step boundary data (3.22).

Table 7 shows the impact of grid refinement (the number of collocation points per subdomain, N_x, and the number of discretization points, N_y) on the computed value of $x^*_\infty(y)$ for the problem with boundary data (3.22), $\Delta\delta = 0.01$.

An obvious way to parallelize the algorithm (3.21) is to partition the interval $(-\pi, \pi)$ into subintervals of equal length $2\pi/N_y$ and map this partition onto a ring of processors. Thus, one can achieve high speedups on a Paragon using nonblocking communications. If each processor covers at least four mesh points in the y direction, only nearest-neighbor communication is needed. Table 8 gives ample evidence that the algorithm (3.21) is highly scalable; doubling the number of processors with the problem size results in almost identical CPU times.

Additional parallelism can be introduced by decomposing the domain in the x

TABLE 8
CPU time for $1,000$ *time steps on the Paragon XP/S as a function of the number of processors* (P) *and the size of the problem (measured by* N_y*);* $N_x = 49$.

N_y	$P=1$	$P=2$	$P=4$	$P=8$	$P=16$
32	129.16	65.88	33.91	17.43	
64	252.85	130.02	65.85	33.85	17.45
128	519.05	257.85	129.68	65.85	33.92
256		515.01	257.49	129.77	65.93

direction. However, our experience with a similar algorithm for combustion problems indicates a potentially significant decrease (as much as 70%) in the efficiency of the algorithm [35].

In general, the algorithm (3.21) is very well adapted to the quasi-one-dimensional structure of the transition layer. The algorithm predicts the location of the transition layer at steady state with a significant accuracy. The time step is of the same order of magnitude as for the one-dimensional analogue (2.6).

3.4. Long-time integration. The algorithm (3.21) needs to be modified for long-time integration. We distinguish between the cases $\beta = 0$ and $\beta \neq 0$.

If $\beta = 0$, we use an algorithm similar to the one described in section 2.4. We start the integration of (3.1) at $t = t_0$, say. We identify the point x^* with the location of the zero of the approximation U of u, averaged over y, at $t = t_0$ and define the profile function u_0 as in (3.4),

$$u_0(x) = -\tanh \frac{x - x^*}{2\varepsilon}. \tag{3.24}$$

Then we integrate the nonlinear boundary-value problem

$$-\varepsilon \Delta v + v_t + u_0 v_x + u_0' v + \delta_0 \varepsilon^{-2} v v_x = 0 \quad \text{on } (-1,1) \times (-\pi, \pi)$$

forward in time, subject to the boundary conditions

$$v|_{x=-1} = \delta_0^{-1} \varepsilon^2 (1 + \delta - u_0(-1)), \quad v|_{x=1} = \delta_0^{-1} \varepsilon^2 (-1 - u_0(1)),$$

using the algorithm

$$-\varepsilon D_x^2 V^n + \frac{V^n - V^{n-1}}{\Delta t} + u_0 D_x V^n + u_0' V^n = \varepsilon D_y^2 V^{n-1} - \delta_0 \varepsilon^{-2} V^{n-1} D_x V^{n-1} \tag{3.25}$$

for $n = 1, 2, \ldots$. We define the approximation U of u,

$$U(x, y, t) = u_0(x) + \delta_0 \varepsilon^{-2} V(x, y, t), \tag{3.26}$$

and integrate as long as the supremum of $U(\cdot\,, \cdot\,, t) - u_0$ remains of the order of $\delta_0 \varepsilon^{-2}$. When this criterion is no longer met, at $t = t_1$, say, we suspend the integration, identify the point x^* with the location of the center of the transition layer (averaged over y), and update the profile function u_0. We repeat the procedure until the steady state is reached.

The time step for the algorithm (3.25) is limited by the (explicit) term $\varepsilon D_y^2 V^{n-1}$, $\Delta t < c(2\pi/N_y)^2/\varepsilon$, for some constant $c < \frac{1}{2}$. This limitation is not too severe, as ε is

TABLE 9
Location of the transition layer at steady state; boundary data (3.28).

$\Delta\delta$	$\langle x^*_\infty \rangle$	Δx^*
$1.0 \cdot 10^{-6}$	0.7099	$< 1.0 \cdot 10^{-4}$
$1.0 \cdot 10^{-5}$	0.7099	$< 1.0 \cdot 10^{-4}$
$1.0 \cdot 10^{-4}$	0.7101	$3.9 \cdot 10^{-3}$
$0.5 \cdot 10^{-3}$	0.7122	$1.9 \cdot 10^{-2}$
$1.0 \cdot 10^{-3}$	0.7162	$3.0 \cdot 10^{-2}$

very small and the variation of the solution in the y direction is exponentially small, so N_y need not be large.

If $\beta \neq 0$, the situation becomes more complicated. One can, of course, extend the algorithm (3.25) trivially by incorporating the convective term in the right member,

$$-\varepsilon D_x^2 V^n + \frac{V^n - V^{n-1}}{\Delta t} + u_0 D_x V^n + u_0' V^n \tag{3.27}$$
$$= \varepsilon D_y^2 V^{n-1} - \delta_0 \varepsilon^{-2} V^{n-1} D_x V^{n-1} - \beta u_0 D_y V^{n-1} - \beta \delta_0 \varepsilon^{-2} V^{n-1} D_y V^{n-1}$$

for $n = 1, 2, \ldots$. Representative results obtained in this way for the boundary-value problem (3.1) with $\beta = 1$ and $\varepsilon = 0.02$ are given in Table 9. The boundary data are again piecewise constant, as in Table 5, but with $\delta_0 = 1.0 \cdot 10^{-6}$,

$$u(-1, y) = \begin{cases} 1 + 1.0 \cdot 10^{-6} - \Delta\delta & \text{if } -\pi \leq y < -\frac{1}{2}\pi, \\ 1 + 1.0 \cdot 10^{-6} + \Delta\delta & \text{if } -\frac{1}{2}\pi \leq y < \frac{1}{2}\pi, \\ 1 + 1.0 \cdot 10^{-6} - \Delta\delta & \text{if } \frac{1}{2}\pi \leq y < \pi. \end{cases} \tag{3.28}$$

The algorithm (3.27) was applied with $N_x = 39$ collocation points per subdomain in the x direction and only $N_y = 16$ interpolation points in the y direction. We observe that the average location of the transition layer does not change appreciably as long as $\Delta\delta$ is of the same order as the average perturbation δ_0. As $\Delta\delta$ increases, the perturbation is no longer small compared with δ_0, and the asymptotic results of section 3.1 do not necessarily apply. Indeed, Δx^* does not appear to vary linearly with $\Delta\delta$, as was the case in Table 6.

A graphical representation of the computed solution U and its deviation from the profile function, $U - u_0$, for the case $\Delta\delta = 1.0 \cdot 10^{-3}$ are given in Figure 2 (contour lines of U at steady state) and Figures 3 and 4 (perspective drawings of U and the difference $U - u_0$ at steady state). The data for these figures were taken after 2,500,000 time steps ($\Delta t = 0.4$).

The results are quite good, but for long-time integration one would do better by looking for v in terms of its Fourier coefficients v_k and adopting an implicit scheme in the y direction. Thus, the constraint on the time step becomes the same as in the one-dimensional case. The algorithm would be of the following type:

$$-\varepsilon D_x^2 V_k^n + \frac{V_k^n - V_k^{n-1}}{\Delta t} + u_0 D_x V_k^n + (i\beta k u_0 + \varepsilon k^2) V_k^n + u_0' V^n = -\delta_0 \varepsilon^{-2} W_k^{n-1} \tag{3.29}$$

for $n = 1, 2, \ldots$; D_y is the matrix of differentiation with respect to y in Fourier space, and W_k is some approximation to the kth Fourier coefficient of $vv_x + \beta v v_y$.

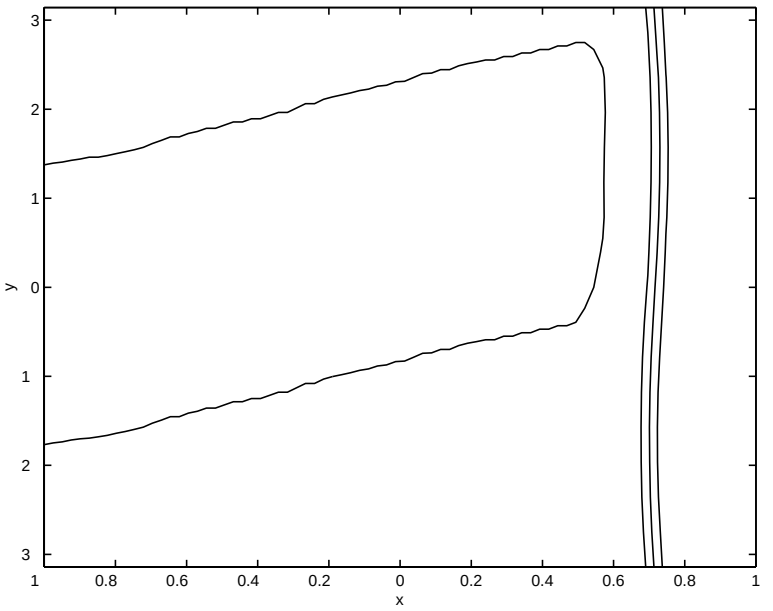


FIG. 2. *Contour lines of the solution U at steady state.*

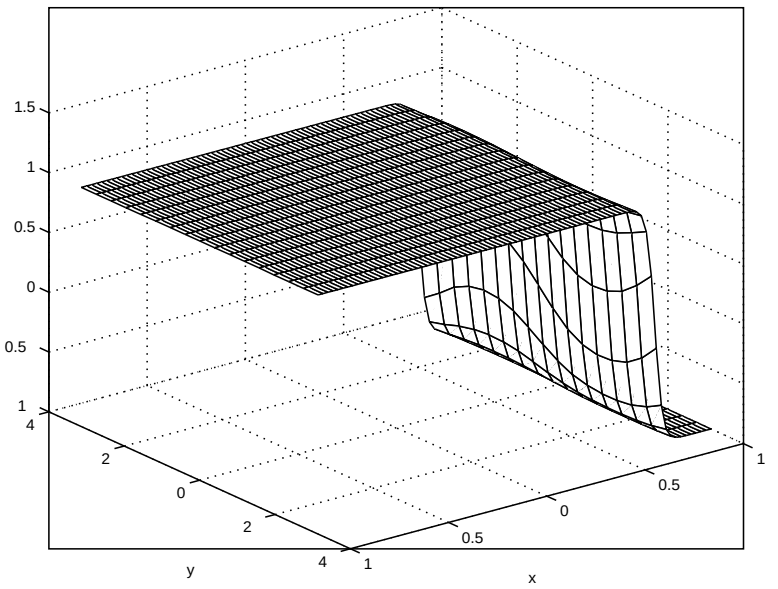


FIG. 3. *The solution U at steady state.*

This algorithm parallelizes with respect to the Fourier modes. It has been applied to a problem involving a propagating combustion front in a moving fluid [35, 36].

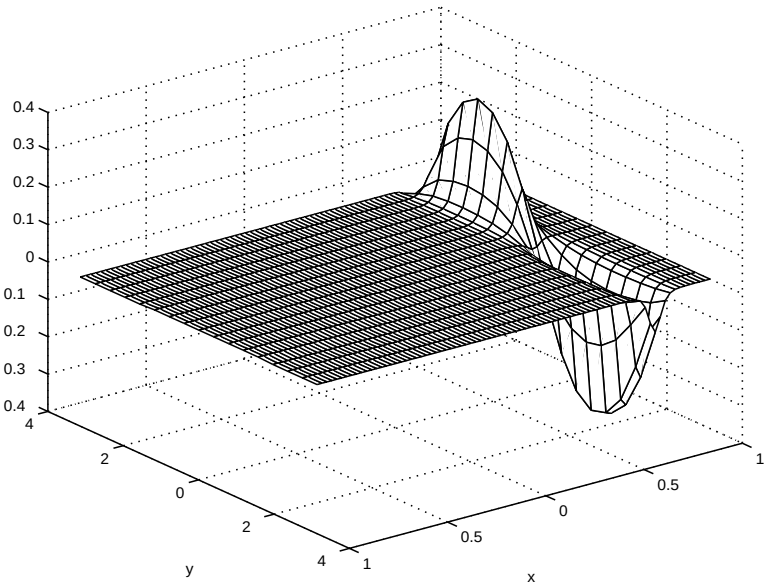


FIG. 4. *The difference $U - u_0$ at steady state.*

REFERENCES

[1] W. ECKHAUS, *Asymptotic Analysis of Singular Perturbations*, North-Holland, Amsterdam, 1979.

[2] D. R. SMITH, *Singular-Perturbation Theory*, Cambridge University Press, Cambridge, New York, 1985.

[3] R. E. O'MALLEY, JR., *Singular Perturbation Methods for Ordinary Differential Equations*, Springer-Verlag, New York, 1991.

[4] J. KEVORKIAN AND J. D. COLE, *Multiple Scale and Singular Perturbation Problems*, Springer-Verlag, New York, 1996.

[5] E. M. DE JAGER AND J. FURU, *The Theory of Singular Perturbations*, Elsevier, Amsterdam, 1996.

[6] P. D. LAX, *Hyperbolic Systems of Conservation Laws and the Mathematical Theory of Shock Waves*, CBMS-NSF Regional Conf. Ser. Appl. Math. 11, SIAM, Philadelphia, 1973.

[7] K. TANUNA, *Asymptotic formulas for the shock wave of the scalar conservation law with smooth initial data*, Quart. Appl. Math., 1 (1992), pp. 109–128.

[8] E. HOPF, *The partial differential equation $u_t + uu_x = \mu u_{xx}$*, Comm. Pure Appl. Math., 3 (1950), pp. 201–230.

[9] P. C. FIFE, *Dynamics of Internal Layers and Diffusive Interfaces*, CBMS-NSF Regional Conf. Ser. Appl. Math. 53, SIAM, Philadelphia, 1988.

[10] J. LORENTZ, *Nonliner Singular Perturbation Problems and the Engquist-Osher Difference Scheme*, Report 8115, University of Nijmegen, Nijmegen, The Netherlands, 1981.

[11] G. KREISS AND H.-O. KREISS, *Convergence to steady state of solution of Burgers' equation*, Appl. Num. Math., 2 (1986) pp. 161–179.

[12] G. KREISS, *Convergence to steady state of solutions of viscous conservation laws*, in Asymptotic and Numerical Methods for Partial Differential Equations with Critical Parameters, H. G. Kaper and M. Garbey, eds., Kluwer Academic Publishers, Dordrecht, The Netherlands, 1993, pp. 225–237.

[13] J. G. L. LAFORGUE AND R. E. O'MALLEY, JR., *The supersensitivity of the shock layer location for Burgers' equation*, in First Pan-American Workshop in Applied and Computational Mathematics, 1993.

[14] J. G. L. LAFORGUE AND R. E. O'MALLEY, JR., *Supersensitive boundary value problems*, in Asymptotic and Numerical Methods for Partial Differential Equations with Critical Parameters, H. G. Kaper and M. Garbey, eds., Kluwer Academic Publishers, Dordrecht, The Netherlands, 1993, pp. 215–224.

[15] J. G. L. Laforgue and R. E. O'Malley, Jr., *On the motion of viscous shocks and the supersensitivity of their steady-state limits*, Methods Appl. Anal., 1 (1994), pp. 465–487.
[16] J. G. L. Laforgue and R. E. O'Malley, Jr., *Shock layer movement for Burgers' equation*, SIAM J. Appl. Math., 55 (1994), pp. 332–347.
[17] J. G. L. Laforgue and R. E. O'Malley, Jr., *Viscous shock motion for advection-diffusion equations*, Stud. Appl. Math., 95 (1995), pp. 147–170.
[18] J. G. L. Laforgue and R. E. O'Malley. Jr., *Exponential asymptotics, the viscous Burgers' equation, and standing wave solutions for a reaction-advection-diffusion model*, Stud. Appl. Math., 102 (1999), pp. 137–172.
[19] L. G. Reyna and M. J. Ward, *On the exponentially slow motion of a viscous shock*, Comm. Pure Appl. Math. 48 (1995), pp. 79–120.
[20] L. G. Reyna and M. J. Ward, *On exponential ill-conditioning and internal layer behavior*, Numer. Funct. Anal. Optim., 16 (1995), pp. 475–500.
[21] M. J. Ward and L. G. Reyna, *Internal layers, small eigenvalues and the sensitivity of metastable motion*, SIAM J. Appl. Math., 55 (1995), pp. 426–446.
[22] J. D. Buckmaster and G. S. S. Ludford, *Lectures on Mathematical Combustion*, CBMS-NSF Regional Conf. Ser. Appl. Math. 43, SIAM, Philadelphia, 1983.
[23] H.-G. Roos, M. Stynes, and L. Tobiska, *Numerical Methods for Singularly Perturbed Differential Equations. Convection-diffusion and flow problems*, Springer Ser. Comput. Math. 24, Springer-Verlag, Berlin, 1996.
[24] A. Bayliss, D. Gottlieb, B. J. Matkowsky, and M. Minkoff, *An adaptive pseudo-spectral method for reaction diffusion problems*, J. Comp. Phys., 81 (1989), pp. 421–443.
[25] A. Bayliss, T. Belytschko, D. Hansen, and E. Turkel, *Adaptive multi-domain spectral methods*, in Proceedings Fifth International Symposium on Domain Decomposition Methods for Partial Differential Equations, T. F. Chan et al., eds., SIAM, Philadelphia, 1992, pp. 195–203.
[26] R. Peyret, *The Chebyshev multidomain approach to stiff problems in fluid dynamics*, Comput. Methods in Appl. Mech. Engrg., 80 (1990), pp. 129–145.
[27] M. Garbey, *Domain decomposition to solve transition layers and asymptotic*, SIAM J. Sci. Comp., 15 (1994), pp. 866–891.
[28] M. Garbey and H. G. Kaper, *A heterogeneous domain decomposition method for singularly perturbed elliptic boundary value problems*, SIAM J. Numer. Anal., 34 (1997), pp. 1513–1544.
[29] C. Canuto, M. Y. Hussaini, A. Quarteroni, and T. A. Zang, *Spectral Methods in Fluid Dynamics*, Springer-Verlag, New York, 1988.
[30] F. Brezzi, C. Canuto, and A. Russo, *A self-adaptive formulation for the Euler/Navier-Stokes coupling*, Comput. Methods Appl. Mech. Engrg., 73 (1989), pp. 317–330.
[31] R. Arina and C. Canuto, *A self-adaptive domain decomposition for the viscous/inviscous coupling.* I. *Burgers equation*, J. Comput. Phys., 105 (1993), pp. 290–300.
[32] Y. Achdou and O. Pironneau, *The χ-method for the Navier-Stokes equation*, C. R. Acad. Sci. Paris, Sér. I, 312 (1991), pp. 1005–1011.
[33] L. Abrahamsson and S. Osher, *Monotone difference schemes for singular perturbation problems*, SIAM J. Numer. Anal., 19 (1982), pp. 979–992.
[34] F. Desprez and M. Garbey, *Direct numerical simulation of a combustion problem on the Paragon machine*, Parallel Comput., 21 (1995), pp. 495–508.
[35] M. Garbey and D. Tromeur-Dervout, *Massively parallel computation of stiff propagating fronts*, Combustion Theory Modeling, 1 (1997), pp. 271–294.
[36] M. Garbey and D. Tromeur-Dervout, *A new parallel solver for the nonperiodic incompressible Navier-Stokes equations with a Fourier method: Application to frontal polymerization*, J. Comput. Phys., 145 (1998), pp. 316–331.

SIAM J. SCI. COMPUT.
Vol. 22, No. 2, pp. 387–405
© 2000 Society for Industrial and Applied Mathematics

EXPLICIT ALGORITHMS FOR A NEW TIME DEPENDENT MODEL BASED ON LEVEL SET MOTION FOR NONLINEAR DEBLURRING AND NOISE REMOVAL*

ANTONIO MARQUINA† AND STANLEY OSHER‡

Dedicated to the memory of Emad Fatemi

Abstract. In this paper we formulate a time dependent model to approximate the solution to the nonlinear total variation optimization problem for deblurring and noise removal introduced by Rudin and Osher [*Total variation based image restoration with free local constraints*, in Proceedings IEEE Internat. Conf. Imag. Proc., IEEE Press, Piscataway, NJ, (1994), pp. 31–35] and Rudin, Osher, and Fatemi [*Phys. D*, 60 (1992), pp. 259–268], respectively. Our model is based on level set motion whose steady state is quickly reached by means of an explicit procedure based on Roe's scheme [*J. Comput. Phys.*, 43 (1981), pp. 357–372], used in fluid dynamics. We show numerical evidence of the speed of resolution and stability of this simple explicit procedure in some representative 1D and 2D numerical examples.

Key words. image restoration, total variation norm, upwind schemes, nonlinear diffusion, level set motion

AMS subject classifications. 65M06, 76R50, 68U10

PII. S1064827599351751

1. Introduction.

Classical algorithms for image deblurring and/or denoising have been mainly based on least squares, Fourier series, and other $\mathcal{L}^2$-norm approximations, and consequently, the results are liken to be contaminated by Gibbs's phenomena (*ringing*) and/or smearing near edges. Their computational advantage comes from the fact that the classical algorithms are linear; thus, fast solvers are widely available. However, the effect of the restoration is not local in space. Other bases of orthogonal functions have been introduced in order to get rid of those problems, e.g., compactly supported wavelets, but Gibbs's phenomenon (*ringing*) and/or smearing are/is still present for these linear procedures.

The total variation (TV) deblurring and denoising models are based on a variational problem with constraints using the TV norm as a nonlinear nondifferentiable functional. The formulation of these models was first given by Rudin, Osher, and Fatemi in [19] for the denoising model and Rudin and Osher in [18] for the denoising and deblurring case. The main advantage is that their solutions preserve edges very well, but there are computational difficulties. Indeed, in spite of the fact that the variational problem is convex, the Euler–Lagrange equations are nonlinear and ill-conditioned. Linear semi-implicit fixed-point procedures devised by Vogel and Oman (see [26]), and interior-point primal-dual implicit quadratic methods by Chan, Golub,

*Received by the editors February 11, 1999; accepted for publication (in revised form) January 14, 2000; published electronically July 13, 2000.
http://www.siam.org/journals/sisc/22-2/35175.html

†Department of Mathematics, University of California, Los Angeles, 405 Hilgard Avenue, Los Angeles, CA 90095-1555 and Departament de Matemàtica Aplicada, Universitat de València, Dr. Moliner, 50, 46100-Burjassot, Spain (marquina@uv.es, http://gata.uv.es/~marquina). The research of this author was supported by NSF grant INT9602089 and DGICYT grant PB97-1402.

‡Department of Mathematics, University of California, Los Angeles, 405 Hilgard Avenue, Los Angeles, CA 90095-1555 (sjo@math.ucla.edu). The research of this author was supported by NSF grant DMS 9706827 and in part by the ARO through grant DAAG 55-98-1-0323.

and Mulet (see [6]), were introduced to solve the models. Those methods give good results when treating pure denoising problems, but the methods become highly ill-conditioned for the deblurring and denoising case where the computational cost is very high and parameter dependent. Furthermore, those methods also suffer from the undesirable *staircase effect*, namely the transformation of smooth regions (*ramps*) into piecewise constant regions (*stairs*).

In this paper we present a very simple time dependent model constructed by evolving the Euler–Lagrange equation of the Rudin–Osher optimization problem, multiplied by the magnitude of the gradient of the solution. The two main analytic features of this formulation are the following: (1) the level contours of the image move quickly to the steady solution and (2) the presence of the gradient numerically regularizes the mean curvature term in a way that preserves and enhances edges and kills noise through the nonlinear diffusion acting on small scales. We use the entropy-violating Roe scheme [16] for the convective term and central differencing for the regularized mean curvature diffusion term. This makes a very simple, stable, explicit procedure, computationally competitive compared with other semi-implicit or implicit procedures. We show numerical evidence of the power of resolution and stability of this explicit procedure in some representative 1D and 2D numerical examples, consisting of noisy and blurred signals and images. (We use Gaussian white noise and Gaussian blur.) We have observed in our experiments that our algorithm shows a substantially reduced staircase effect.

2. Deblurring and denoising. A recording device or a camera would record a signal or image so that (1) the recorded intensity of a small region is related to the true intensities of a neighborhood of the pixel through a degradation process usually called *blurring* and (2) the recorded intensities are contaminated by random noise.

To fix our ideas we restrict the discussion to $\mathbb{R}^2$. An image can be interpreted as either a real function defined on Ω, a bounded and open domain of $\mathbb{R}^2$ (for simplicity we will assume Ω to be the unit square henceforth), or as a suitable discretization of this continuous image. Our interest is to restore an image which is contaminated with noise and blur in such a way that the process should recover the edges of the image.

Let us denote by u_0 the observed image and u the real image. A model of blurring comes from the degradation of u through some kind of averaging. Indeed, u may be blurred through the application of a kernel: $k(x, s, y, r)$ by means of

$$v_0(x, y) = \int_\Omega u(s, r)\, k(x, s, y, r)\, ds\, dr, \tag{2.1}$$

and we denote this operation by $v_0 = k * u$. The model of degradation we assume is

$$k * u + n = u_0, \tag{2.2}$$

where n is Gaussian white noise, i.e., the values n_i of n at the pixels i are independent random variables, each with a Gaussian distribution of zero mean and variance σ^2. We will use the *signal to noise ratio* of the signal u to measure the level of noise, defined as

$$SNR = \frac{||u - \overline{u}||_{\mathcal{L}^2}}{\sigma},$$

where $\overline{u}$ is the mean of the signal u, i.e., the ratio of the standard deviation of the signal over the standard deviation of the noise. If the kernel k is translation-invariant, i.e.,

there is a function $j(x,y)$ (also called a kernel) such that $k(x,s,y,r) = j(x-s,y-r)$ and the blurring is defined as a "superposition" of js,

$$v_0(x,y) = (j*u)(x,y) = \int_\Omega u(s,r)\, j(x-s,y-r)\, ds\, dr, \tag{2.3}$$

and this isotropic blurring is called *convolution*. Otherwise, if the kernel k is not translation-invariant we call this blurring *anisotropic*. For the sake of simplicity, we suppose that the blurring is coming from a convolution, through a kernel function j such that $j*u$ is a compact integral operator. Typically, j has the following properties, $j(x,y) \geq 0$, $j(x,y) \to 0$, as $(x^2+y^2)^{1/2}$ goes to ∞ and $\int_{\mathbb{R}^2} j(x,y)\, dx\, dy = 1$. For any $\alpha > 0$ the so-called *heat kernel*, defined as

$$j(x,y) = \frac{1}{4\pi\alpha}\, e^{-(x^2+y^2)/4\alpha}, \tag{2.4}$$

is an important example that we will use in our numerical experiments.

The main advantage of the convolution is that if we take the Fourier transform of (2.3) we get

$$\hat{v}_0(k,l) = \hat{j}(k,l)\hat{u}(k,l); \tag{2.5}$$

then, to solve the model (2.2) with $k = j$, we take Fourier transform and we arrive at

$$\hat{j}(k,l)\hat{u}(k,l) + \hat{n}(k,l) = \hat{u_0}(k,l). \tag{2.6}$$

To recover $u(x,y)$, we need to deconvolve, i.e., we have to divide (2.6) by $\hat{j}(k,l)$ and apply the inverse Fourier transform. This procedure is generally very ill-posed. Indeed, j is usually smooth and $\hat{j}(k,l) \to 0$ rapidly as $(k^2+l^2)^{1/2}$ goes to ∞; thus, large frequencies in u_0 get amplified considerably. The function u_0 is generally piecewise smooth with jumps in the function values and derivatives; thus, the Fourier method approximation gives global error estimates of order $O(h)$ (see [11]) and suffers from Gibbs's phenomenon. Discrete direct methods dealing with the linear integral equation (2.6) have been designed by different authors (see [13] and references therein).

One way to make life easier is to consider a variational formulation of the model that regularizes the problem. Our objective is to estimate u from statistics of the noise, blur, and some a priori knowledge of the image (smoothness, existence of edges). This knowledge is incorporated into the formulation by using a functional R that measures the quality of the image u, in the sense that smaller values of $R(u)$ correspond to better images. The process, in other words, consists in the choice of the best quality image among those matching the constraints imposed by the statistics of the noise together with the blur induced by j.

The usual approach consists in solving constrained optimization problem

$$\begin{aligned} &\min_u R(u) \\ &\text{subject to } ||j*u-u_0||^2_{\mathcal{L}^2} = |\Omega|\,\sigma^2, \end{aligned} \tag{2.7}$$

since $n = u_0 - j*u$ and $E(\int_\Omega n^2\, dx) = |\Omega|\,\sigma^2$ ($E(X)$ denotes the expectation of the random variable X) imply that $||j*u-u_0||^2_{\mathcal{L}^2} = \int_\Omega (j*u-u_0)^2\, dx \approx |\Omega|\sigma^2$.

Examples of regularization functionals that can be found in the literature are $R(u) = \|\Delta u\|_{\mathcal{L}^2}, \|\nabla u\|_{\mathcal{L}^2}$, where ∇ is the gradient and Δ is the Laplacian; see [22, 8].

The main disadvantage of using these functionals is that they do not allow discontinuities in the solution, therefore, the edges can not be satisfactorily recovered.

In [19], the *TV norm* is proposed as a regularization functional for the image restoration problem:

$$TV(u) = \int_\Omega |\nabla u|\, dx = \int_\Omega \sqrt{u_x^2 + u_y^2}\, dx. \tag{2.8}$$

The TV norm does not penalize discontinuities in u, and thus allows us to recover the edges of the original image. There are other functionals with similar properties introduced in the literature for different purposes (see, for instance, [7, 5, 25, 2]). The restoration problem can be thus written as

$$\begin{aligned} &\min_u \int_\Omega |\nabla u|\, dx \\ &\text{subject to } \frac{1}{2}\Big(\int_\Omega (j * u - u_0)^2\, dx - |\Omega|\,\sigma^2\Big) = 0. \end{aligned} \tag{2.9}$$

Its Lagrangian is

$$\int_\Omega |\nabla u|\, dx + \frac{\lambda}{2}\Big(\int_\Omega (j * u - u_0)^2\, dx - |\Omega|\,\sigma^2\Big), \tag{2.10}$$

and its Euler–Lagrange equations, with homogeneous Neumann boundary conditions for u, are

$$0 = -\nabla \cdot \left(\frac{\nabla u}{|\nabla u|}\right) + \lambda\,(j * (j * u - u_0)), \tag{2.11}$$

$$0 = \frac{1}{2}\left(\int_\Omega (j * u - u_0)^2\, dx - |\Omega|\,\sigma^2\right). \tag{2.12}$$

There are known techniques (see [3]) for solving the constrained optimization problem (2.9) by exploiting solvers for the corresponding unconstrained problem whose Euler–Lagrange equations are (2.11) for λ fixed. Therefore, for the sake of clarity, we will assume the Lagrange multiplier λ to be known throughout the exposition. For $\nu = \frac{1}{\lambda}$, we can then write the equivalent unconstrained problem as

$$\min_u \int_\Omega \left(\nu\,|\nabla u| + \frac{1}{2}\,(j * u - u_0)^2\right) dx \tag{2.13}$$

and its Euler–Lagrange equation in the more usual form:

$$0 = -\nabla \cdot \left(\frac{\nabla u}{|\nabla u|}\right) + \lambda\, j * (j * u - u_0). \tag{2.14}$$

We call (2.14) the *nonlinear deconvolution model*. The linear deconvolution model would be

$$0 = -\Delta u + \lambda\, j * (j * u - u_0). \tag{2.15}$$

that comes from the Euler–Lagrange equation of the corresponding unconstrained problem with the norm $R(u) = \|\nabla u\|_{\mathcal{L}^2}$.

Since (2.14) is not well defined at points where $\nabla u = 0$, due to the presence of the term $1/|\nabla u|$, it is common to slightly perturb the TV functional to become

$$\int_\Omega \sqrt{|\nabla u|^2 + \beta}\, dx, \tag{2.16}$$

where β is a small positive parameter, or

$$\int_\Omega |\nabla u|_\beta\, dx, \tag{2.17}$$

with the notation ($x \in \mathbb{R}$, $v \in \mathbb{R}^2$)

$$|x|_\beta = \sqrt{x^2 + \beta}, \quad |v|_\beta = \sqrt{|v|^2 + \beta}. \tag{2.18}$$

3. The time dependent model. Vogel and Oman and Chan, Golub, and Mulet devised direct methods to approximate the solution to the Euler–Lagrange equation (2.14) with an a priori estimate of the Lagrange multiplier and homogeneous Neumann boundary conditions. Those methods work well for denoising problems, but the removal of blur becomes very ill-conditioned with a user-dependent choice of parameters. However, stable explicit schemes are preferable when the steady state is quickly reached because the choice of parameters is almost user-independent. Moreover, the programming for our algorithm is quite simple compared to the implicit inversions needed in the above mentioned methods.

Usually, time dependent approximations to the ill-conditioned Euler–Lagrange equation (2.14) are inefficient. This is because a very small time step is required when a simple explicit scheme is used. This is the case with the following formulation due to Rudin, Osher, and Fatemi (see [19]) and Rudin and Osher (see [18]):

$$u_t = -\lambda\, j * (j * u - u_0) + \nabla \cdot \left(\frac{\nabla u}{|\nabla u|} \right) \tag{3.1}$$

with $u(x, y, 0)$ given as initial data (they used as initial guess the original blurry and noisy image u_0) and homogeneous Neumann boundary conditions, i.e., $\frac{\partial u}{\partial n} = 0$ on the boundary of the domain. As t increases, we approach a restored version of our image, and the effect of the evolution should be edge detection and enhancement and smoothing at small scales to remove the noise. This solution procedure is a parabolic equation with time as an evolution parameter and resembles the gradient-projection method of Rosen (see [17]). In this formulation we assume an a priori estimate of the Lagrange multiplier, in contrast with the dynamic change of λ supposed in the Rosen method (see section 6 for details). Equation (3.1) moves each level curve of u normal to itself with normal velocity equal to the curvature of the level surface divided by the magnitude of the gradient of u (see [23, 15, 20]). The constraints are included in the λ-term, and they are needed to prevent distortion and to obtain a nontrivial steady state.

However, this evolution procedure converges very slowly to its steady state since the parabolic term is singular for small gradients. In fact, an ad hoc rule of thumb would indicate that the timestep Δt and the space stepsize Δx need to be related by

$$\frac{\Delta t}{\Delta x^2} \leq c\, |\Delta u|, \tag{3.2}$$

for fixed $c > 0$, for stability. This restriction called the Courant–Friedrichs–Lewy restriction (CFL for short) is what we shall relax. These issues are well documented in numerous experiments. In order to avoid these difficulties, we propose a new time dependent model that accelerates the movement of level curves of u and regularizes the parabolic term in a nonlinear way. In order to regularize the parabolic term we multiply the whole Euler–Lagrange equation (2.14) by the magnitude of the gradient and our time evolution model reads as follows:

$$u_t = -|\nabla u|\,\lambda\, j * (j * u - u_0) + |\nabla u|\,\nabla \cdot \left(\frac{\nabla u}{|\nabla u|}\right). \tag{3.3}$$

We use as initial guess the original blurry and noisy image u_0 and homogeneous Neumann boundary conditions as above, with an a priori estimate of the Lagrange multiplier. Since this model does not increase the total variation of the initial guess, the above choice is reasonable for slightly blurred and noisy images. When the blur is strong we can use as initial guess some regularized linear deconvolution of the blurry and noisy image. In our case we propose the solution of (2.15) with a linear Lagrange multiplier that matches the noise using a fast solver. We have used this choice for the 1D case when strong blur is present (the 2D case is a work in progress). Our current 2D numerical experiments use u_0 as initial guess. The effect of this reformulation, (i.e., preconditioning) is positive in various aspects:

1. The effect of the regularizing term means that the movement of level curves of u is pure mean curvature motion (see [15]).
2. The total movement of level curves goes in the direction of the zeros of $j*u-u_0$ regularized by the anisotropic diffusion introduced by the curvature term.
3. The problem for the denoising case is well-posed in the sense that there exists a maximum principle that determines the solution (see [15]).
4. There are simple explicit schemes, such as Roe's scheme, applied to the convective term and central differencing used for the regularized second order term that behave stably with a CFL restriction of the form

$$\frac{\Delta t}{\Delta x^2} < c$$

   for a fixed number $c > 0$.
5. This procedure is more *morphological* (see [1]) in the pure denoising case; i.e., it operates mainly on the level sets of u and u_0. This is easily seen if we replace u by $h(u)$ and u_0 by $h(u_0)$ with $h' > 0$. Then, (3.3) is invariant, except that $u - u_0$ gets replaced by $(h(u) - h(u_0))/h'(u)$.
6. The numerical steady state reached with this model when the singular term is regularized is different from the one obtained by the TV model, allowing smoother ramps, and thus reducing the staircase effect.
7. Explicit schemes can also be applied for the "anisotropic blurring" case.

The anisotropic diffusion introduced in this model is a nonlinear way to discriminate scales of computation. This never occurs with a linear model, (e.g. the linear deconvolution model), because in this case we would have the linear heat equation with constant diffusion. Thus, our model (3.3) can be seen as a convection-diffusion equation with *morphological* convection and anisotropic diffusion. Items 1, 2, and 5 above are desirable properties of any image processing algorithm; i.e., operations should be performed on the *level contours* of the image. This yields contrast invariant results, and this is the basis for the rigorous analysis performed in [1].

4. Explicit numerical schemes for the 1D model. The 2D model described before is more regular than the corresponding 1D model because the 1D original optimization problem is barely convex. For the sake of understanding the numerical behavior of our schemes, we also discuss the 1D model. The Euler–Lagrange equation in the 1D case reads as follows:

$$0 = -\left(\frac{u_x}{|u_x|}\right)_x + \lambda\, j * (j * u - u_0). \tag{4.1}$$

This equation can be written either as

$$0 = -\left(\frac{u_x}{|u_x|_\beta}\right)_x + \lambda\, j * (j * u - u_0), \tag{4.2}$$

using the small regularizing parameter $\beta > 0$ introduced at the end of the previous section, or

$$0 = -\delta(u_x) u_{xx} + \lambda\, j * (j * u - u_0), \tag{4.3}$$

using the δ-function.

The Rudin–Osher–Fatemi model (ROF model) in terms of the δ-function will read as follows:

$$u_t = -\lambda\, j * (j * u - u_0) + \delta(u_x)\, u_{xx}. \tag{4.4}$$

Our model in 1D will be

$$u_t = -|u_x|\, \lambda\, j * (j * u - u_0) + \frac{\beta}{\beta + u_x^2}\, u_{xx}, \tag{4.5}$$

where $\beta > 0$ is the small regularizing parameter. The parameter $\beta > 0$ in this model is estimated from the local amount of noise. We have found for our model, through our numerical experiments in 1D and 2D, that β can be estimated as the standard deviation of the noise. We have found, through our numerical experiments, that this regularization of the second order term removes spurious oscillations near discontinuities.

We can also state our model in terms of the δ function as

$$u_t = -|u_x|\, \lambda\, j * (j * u - u_0) + |u_x|\, \delta(u_x)\, u_{xx}, \tag{4.6}$$

where a convolution of the δ function must be used in practice. The intensity of this kind of convolution decides which scale the diffusion term is acting. In this paper, we always approximate δ by

$$\delta(z) \approx \beta \cdot (z^2 + \beta)^{-3/2} \tag{4.7}$$

in order to be consistent with the regularization used for the absolute value function (2.18). Consistency in this case means that the derivative of the regularized absolute value function must be equal to the regularized sign function, and the derivative of the above must be equal to the regularized delta function.

A radical way to make the coefficient of u_{xx} nonsingular is to solve the evolution model:

$$u_t = -\frac{1}{\delta(u_x)}\, \lambda\, j * (j * u - u_0) + u_{xx}. \tag{4.8}$$

This model works in such a manner that away from extrema we have a large multiplier of $-j * (j * u - u_0)$ and at extrema it is just the heat equation.

These evolution models are initialized with the blurry and noisy signal u_0 and homogeneous Neumann boundary conditions, and with a prescribed Lagrange multiplier for slightly blurred and noisy signals. If the signal is strongly blurred, then we propose to use an initial guess, u_{00}, which is the solution of the *linear deconvolution model* equation (2.15) with a linear Lagrange multiplier that matches the noise and allows an increase in total variation of the signal.

We estimated $\lambda > 0$ near the maximum value such that the explicit scheme is stable under appropriate CFL restrictions (see below), provided β is chosen to be the standard deviation of the noise.

In order to convince the reader about the speed and programming simplicity of our model, we shall give the details of the first order scheme for the 1D pure denoising model, i.e.,

$$u_t = -|u_x|\,\lambda\,(u - u_0) + \frac{\beta}{\beta + u_x^2}\,u_{xx}. \tag{4.9}$$

Let u_j^n be the approximation to the value $u(x_j, t_n)$, where $x_j = j\Delta x$ and $t_n = n\Delta t$. Then the scheme for the problem (4.9) will be

$$\frac{u_j^{n+1} - u_j^n}{\Delta t} = -|ug_j|\,\lambda\,(u_j^n - u_0(x_j)) + \frac{\beta}{\beta + g_j^2}\,\frac{u_{j+1}^n - 2u_j^n + u_{j-1}^n}{\Delta x^2}, \tag{4.10}$$

where

$$g_j = \frac{u_{j+1}^n - u_{j-1}^n}{2\Delta x}$$

and ug_j is the upwind gradient, i.e.,

$$ug_j = \frac{u_j^n - u_{j-1}^n}{\Delta x}$$

if $g_j(u_j^n - u_0(x_j)) > 0$ and

$$ug_j = \frac{u_{j+1}^n - u_j^n}{\Delta x}$$

if $g_j(u_j^n - u_0(x_j)) < 0$

Our general explicit scheme has the following features:

1. We use central differencing for u_{xx}.
2. The convolution operator j is computed by evolving the heat equation $u_t = u_{xx}$ with the explicit Euler method in time and central differencing in space with a CFL restriction $\frac{\Delta t}{\Delta x^2} < 0.25$, corresponding to an $\epsilon > 0$ of the 1D heat kernel,

$$j(x) = \frac{1}{2\sqrt{\pi\epsilon}}\,e^{-x^2/4\epsilon}, \tag{4.11}$$

in order to ensure homogeneous Neumann boundary conditions. We note that we may compute the convolution operator through a direct convolution with a discrete approximation of the kernel j (*point spread function* (PSF). In

order to make convolution faster the FFT can be used, provided we consider periodic signals and therefore the convolution operator is cyclic. However, the reader should be aware that most of the available FFT packages are designed such that the signals are enlarged with zero values. For the sake of computational efficiency and consistency with homogeneous Neumann boundary conditions a work in progress is to convolve, using the *fast cosine transform*, enlarging the signal with the mirror image.

3. We use upwind Roe differencing (see [16, 10]), checking the direction of propagation by computing the sign of the derivative of the coefficient of $j*(j*u-u_0)$ with respect to u_x times the sign of this term. Indeed, for our evolution model (4.5) it is enough to check the sign of $u_x \cdot j*(j*u-u_0)$. For the model (4.8) we get the same direction of propagation as before. We note that there is no notion of "entropy condition satisfying" discontinuities in image processing; thus we omit the usual "entropy-fix" applied to the Roe solver in this work.
4. The CFL condition depends on λ and β.

Indeed, the parabolic term in our model (4.5) gives a CFL restriction

$$\frac{\Delta t}{\Delta x^2} \leq \frac{\beta + u_x^2}{2\beta}, \tag{4.12}$$

and the convection term gives

$$\frac{\Delta t}{\Delta x} \leq c\,\lambda \sqrt{1 + \frac{\beta}{u_x^2}} \tag{4.13}$$

for fixed c. These restrictions are reasonable at local extrema and near edges, compared with the parabolic CFL restriction that corresponds to the reaction-diffusion ROF model (4.4):

$$\frac{\Delta t}{\Delta x^2} \leq \frac{1}{2\delta(u_x)}, \tag{4.14}$$

which is too stiff along flat regions or at local extrema. The CFL restriction coming from the convection term in the radical model (4.8) is better but also unfortunate:

$$\frac{\Delta t}{\Delta x} \leq \frac{\beta}{3|u_x|(\beta + u_x^2)^{1/2}}. \tag{4.15}$$

Thus, our model is more convenient from this point of view.

5. Explicit numerical schemes for the 2D model. We can express our 2D model in terms of explicit partial derivatives as

$$u_t = -\sqrt{u_x^2 + u_y^2}\;\lambda\, j*(j*u-u_0) + \frac{u_{xx}(u_y^2+\beta) - 2u_{xy}u_xu_y + u_{yy}(u_x^2+\beta)}{u_x^2+u_y^2+\beta}, \tag{5.1}$$

using u_0 as initial guess and homogeneous Neumann boundary conditions (i.e., absorbing boundary). We only consider in this paper slightly blurred images, therefore this choice of the initial guess is reasonable.

We remark that the above regularization of the curvature term is consistent with one used in our 1D model, i.e., for $u_y = 0$ we get (4.5).

The parameter β is the standard deviation of the noise and it is chosen to be very small for a pure deblurring problem. Then, the Lagrange multiplier $\lambda > 0$ is chosen as large as possible subject to the above mentioned CFL restriction $\frac{\Delta t}{\Delta x^2} < c$.

Let u_{ik}^n be the approximation to the value $u(x_i, y_k, t_n)$, where $x_i = i\Delta x$, $y_k = k\Delta y$, and $t_n = n\Delta t$, where Δx, Δy, and Δt are the spatial stepsizes and the time stepsize, respectively. We define the quantities $v_0 = j * u_0$ and $w_{ik}^n = j * j * (u_{ik}^n)$. We point out that we used for j the convolution with the 2D heat kernel, (2.4), in our experiments, aproximated by evolving the 2D heat equation $u_t = u_{xx} + u_{yy}$ by means of the explicit Euler method in time and central differencing in space. Then our first order scheme reads as follows:

$$\frac{u_{ik}^{n+1} - u_{ik}^n}{\Delta t} = -\sqrt{ug_{ik}^{x\ 2} + ug_{ik}^{y\ 2}}\ \lambda\,(w_{ik}^n - v_0(x_i, y_k)) + s_{ik}^n, \tag{5.2}$$

where the second order term is defined by

$$s_{ik}^n := 0 \tag{5.3}$$

if $g_{ik}^{x\ 2} + g_{ik}^{y\ 2} < \beta$ and

$$s_{ik}^n := \frac{g_{ik}^{xx}\, g_{ik}^{y\ 2} - 2g_{ik}^{xy} g_{ik}^x g_{ik}^y + g_{ik}^{yy} g_{ik}^{x\ 2}}{g_{ik}^{x\ 2} + g_{ik}^{y\ 2}} \tag{5.4}$$

otherwise, where

$$g_{ik}^x = \frac{u_{i+1,k}^n - u_{i-1,k}^n}{2\Delta x}, \tag{5.5}$$

$$g_{ik}^y = \frac{u_{i,k+1}^n - u_{i,k-1}^n}{2\Delta y}, \tag{5.6}$$

$$g_{ik}^{xx} = \frac{u_{i+1,k}^n - 2u_{ik}^n + u_{i-1,k}^n}{\Delta x^2}, \tag{5.7}$$

$$g_{ik}^{yy} = \frac{u_{i,k+1}^n - 2u_{ik}^n + u_{i,k-1}^n}{\Delta y^2}, \tag{5.8}$$

$$g_{ik}^{xy} = \frac{u_{i+1,k+1}^n - u_{i-1,k+1}^n - u_{i+1,k-1}^n + u_{i-1,k-1}^n}{2\Delta x \Delta y}; \tag{5.9}$$

ug_{ik}^x is the upwind gradient in the x-direction, i.e.,

$$ug_{ik}^x = \frac{u_{ik}^n - u_{i-1,k}^n}{\Delta x} \tag{5.10}$$

if $g_{ik}^x(w_{ik}^n - v_0(x_i, y_k)) > 0$; and

$$ug_{ik}^x = \frac{u_{i+1,k}^n - u_{ik}^n}{\Delta x} \tag{5.11}$$

if $g^x_{ik}(w^n_{ik} - v_0(x_i, y_k)) < 0$; and ug^y_{ik} is the upwind gradient in the y-direction, i.e.,

$$ug^y_{ik} = \frac{u^n_{ik} - u^n_{i,k-1}}{\Delta y} \tag{5.12}$$

if $g^y_{ik}(w^n_{ik} - v_0(x_i, y_k)) > 0$ and

$$ug^y_{ik} = \frac{u^n_{i,k+1} - u^n_{ik}}{\Delta y} \tag{5.13}$$

if $g^y_{ik}(w^n_{ik} - v_0(x_i, y_k)) < 0$.

A very simple way to extend this scheme to get high order accuracy is to follow Shu–Osher prescription (see [21]). Thus, we consider a method of lines, using an explicit high order Runge–Kutta method in time and using a method of spatial ENO reconstruction (see [24, 9, 21, 12]), of the same order, for the convection term, applied on every time substep.

We have tested the Van Leer second order MUSCL (defined in [24]) spatial reconstruction using the **minmod** function as slope-limiter together with classical second order Runge–Kutta method and the third order PHM spatial reconstruction as in [12], using as slope-limiter the **harmod** function, consisting of the harmonic mean of the lateral slopes when they have the same sign and zero when they have different sign, together with the third order Shu–Osher Runge–Kutta method of [21]. We have found that these explicit methods are stable and give high accuracy under the same CFL restrictions as the first order scheme.

As a sample we shall describe the second order MUSCL method. Since the Runge–Kutta methods used here are linear combination of first order explicit Euler timesteps, it is enough to formulate one Euler step (in fact, in this case it is Heun's method which is the arithmetic mean of two Euler timesteps). Following the notation used above we have

$$\frac{u^{n+1}_{ik} - u^n_{ik}}{\Delta t} = -\sqrt{rug^x_{ik}{}^2 + rug^y_{ik}{}^2}\ \lambda\,(w^n_{ik} - v_0(x_i, y_k)) + s^n_{ik}, \tag{5.14}$$

where the reconstructed upwind gradients rug^x_{ik} and rug^y_{ik} are computed in the following way. We reconstruct the left x-gradient in (x_i, y_k) from the linear function

$$pl(x) := m_{i-1}(x - x_{i-1/2}) + \frac{u^n_{ik} - u^n_{i-1,k}}{\Delta x}, \tag{5.15}$$

where

$$m_{i-1} = \operatorname{minmod}(g^{xx}_{i-1,k}, g^{xx}_{i,k}) \tag{5.16}$$

computed in x_i, i.e.,

$$gl^x_i := pl(x_i), \tag{5.17}$$

where the *minmod* function is defined as

$$\operatorname{minmod}(r, s) := \frac{1}{2}\,\min(|r|, |s|)\,(\operatorname{sgn}(r) + \operatorname{sgn}(s)), \tag{5.18}$$

sgn being the sign function. Analogously, we have the reconstructed right x-gradient, gr^x_i, as

$$gr^x_i := pr(x_i), \tag{5.19}$$

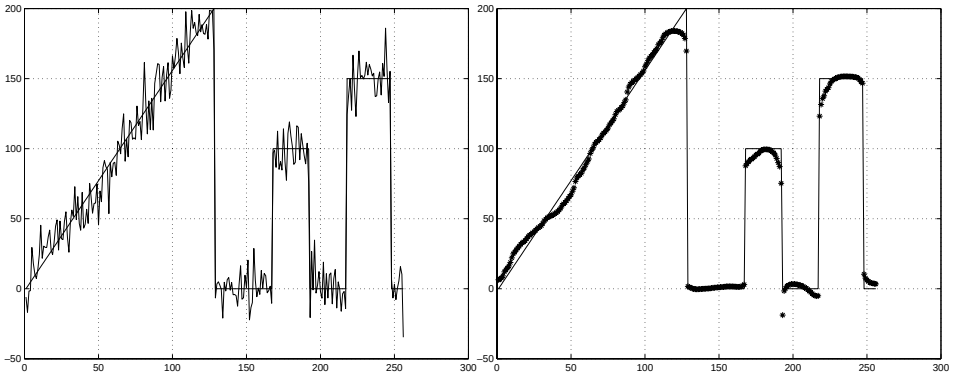


FIG. 6.1. *Left: original vs. noisy* 1D *image. Right: original vs. recovered* 1D *image.*

where

$$pr(x) := m_i(x - x_{i+1/2}) + \frac{u^n_{i+1,k} - u^n_{i,k}}{\Delta x}, \tag{5.20}$$

where

$$m_i = \text{minmod}(g^{xx}_{i,k}, g^{xx}_{i+1,k}). \tag{5.21}$$

Then the reconstructed upwind gradient in the x-direction is defined from the mean value

$$gm^x_i := \frac{gl^x_i + gr^x_i}{2} \tag{5.22}$$

as

$$rug^x_{ik} = gl^x_i \tag{5.23}$$

if $gm^x_i(w^n_{ik} - v_0(x_i, y_k)) > 0$ and

$$rug^x_{ik} = gr^x_i \tag{5.24}$$

if $gm^x_i(w^n_{ik} - v_0(x_i, y_k)) < 0$. The procedure in the y-direction is similar.

6. Numerical experiments. In this section, we perform some numerical experiments in 1D and 2D.

We have used 1D signals with values in the range $[0, 255]$. Figure 6.1 (left) represents the original signal vs. the noisy signal with $SNR \approx 5$. Figure 6.1 (right) represents the original signal vs. the recovered signal after 80 iterations with first order scheme with CFL number 0.25. The estimated $\lambda = 0.05$ was computed as a value near the maximum allowed for stability, using the explicit Euler method in time. We have used $\beta = 15$ in this experiment in order to achieve the appropriate amount of diffusion at small scales and this value corresponds to the standard deviation of the noise. Let us observe the very reduced *staircase effect*, compared with the usual one obtained with either fixed-point iterative methods or nonlinear primal-dual methods (see [4]).

In order to show the power of reduction of the *staircase effect* with our model we apply our explicit procedure to a very unusual noisy signal, proposed by Chambolle

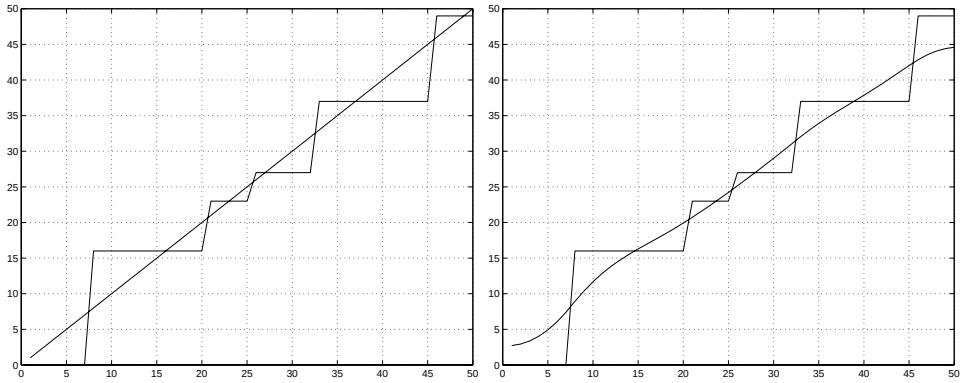


FIG. 6.2. *Left: original ramp* 1D *signal vs. staircased noisy* 1D *signal. Right: noisy signal vs. recovered* 1D *signal.*

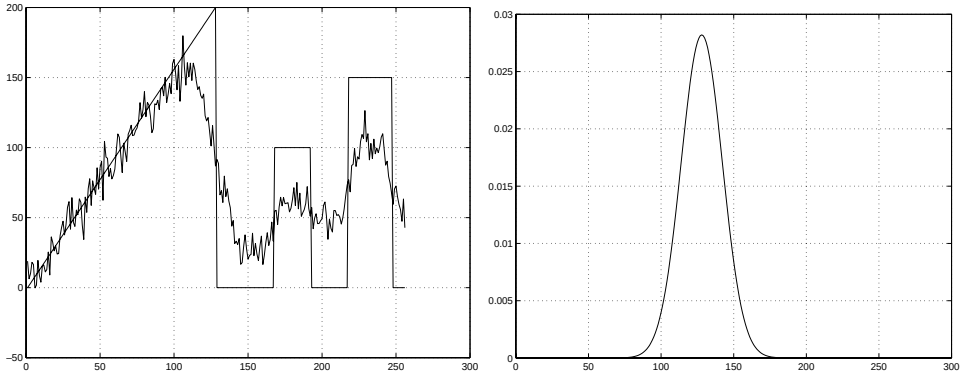


FIG. 6.3. *Left: original vs. blurry and noisy* 1D *signal. Right:* 1D *PSF.*

and Lions in [5]. The original signal we want to recover is simply the function $u(x) = x, x \in [0, 50]$, and the noise has turned it into a piecewise constant nondecreasing function. The standard deviation of the noise is approximately $\sigma \approx 3.5$, (see Figure 5 in [5]). The signal of Figure 6.2 (left) represents the original ramp signal versus the staircased noisy signal with $\sigma = 3.5$. The signal of Figure 6.2 (right) represents the noisy signal versus the recovered signal after 80 iterations with our first order scheme for CFL number 0.25. We use $\beta = \sigma$ and the estimated Lagrange multiplier is $\lambda = 0.005$.

We note that for the above 1D pure denoising problems we always use the noisy signal as initial guess.

However, in the following experiment where strong blur is present we use a different choice. Figure 6.3 (left) represents the original signal vs. the strongly blurred and noisy signal with $\epsilon = 50$ (as in (4.11)), and $SNR \approx 5$. The aproximation to the PSF of the corresponding Gaussian blur is represented in Figure 6.3 (right). We solve the linear deconvolution model (2.15) by using a Lagrange multiplier equal to 1, using the Matlab FFT package, which uses homogeneous periodic boundary conditions. Figure 6.4 (left) represents computed solution vs. the blurry and noisy signal. The linear deconvolved signal u_{00} appears to be an oscillatory approximation to a smooth function. Edges are not apparent with this somewhat denoised result. The denoising is

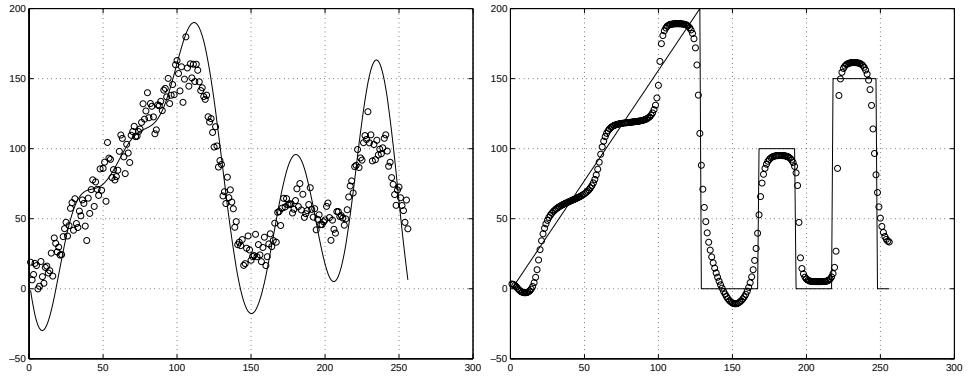


FIG. 6.4. *Left: computed linear deconvolution* u_{00} *vs. blurry and noisy* 1D *signal. Right: original vs. recovered* 1D *signal.*

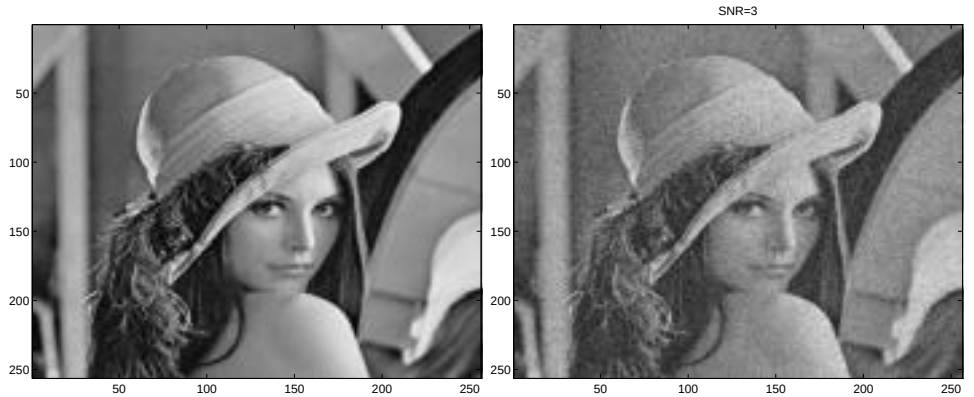


FIG. 6.5. *Left: original image. Right: noisy image, SNR*≈ 3.

due to the regularization effect of the chosen $\lambda = 1$ that matches the noise. Moreover, the total variation of u_{00} is substantially larger than the signal u_0. Then we use our model, using u_{00} as initial guess (u_0 in the equation!), and after 200 iterations with first order scheme with CFL number 0.1, we get the signal represented in Figure 6.4 (right). We use $\beta = 9$, which corresponds to the standard deviation of the noise. The estimated $\lambda = 4.5$ was computed as a value near the maximum allowed for stability. We also observe a reduced *staircase effect*. We performed many other experiments with 1D signals, obtaining similar results.

All our 2D numerical experiments were performed on the original image (Figure 6.5 (left)) with 256×256 pixels and dynamic range in $[0, 255]$.

The third order scheme we used in our 2D experiments was based on the third order Runge–Kutta introduced by Shu and Osher (see [21]) to evolve in time with a third order spatial approximation based on the PHM reconstruction introduced in [12].

Our first 2D experiment was made on the noisy image, see Figure 6.5 (right), with a SNR which is approximately 3. Details of the approximate solutions using the Chan–Golub–Mulet primal-dual method and our time dependent model using the third order Roe's scheme (described above) are shown in Figure 6.6. We used $\lambda \approx 0.0713$ and we perform 50 iterations with $\Delta t/\Delta x^2 = 0.1$. We used the same estimated λ as the

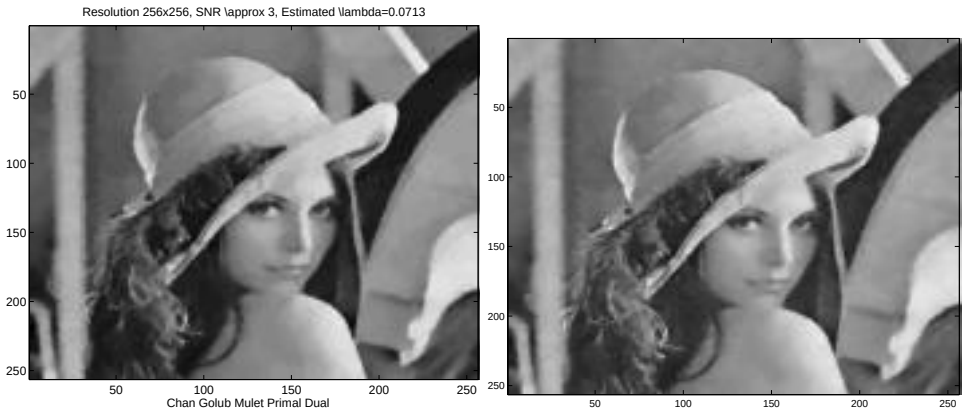


FIG. 6.6. *Left: image obtained by the Chan–Golub–Mulet primal-dual method. Right: image obtained by our time evolution model, with* 50 *timesteps,* $\Delta t/\Delta x^2 = 0.1$, $\beta = 14$, *and* $\lambda = 0.0713$.

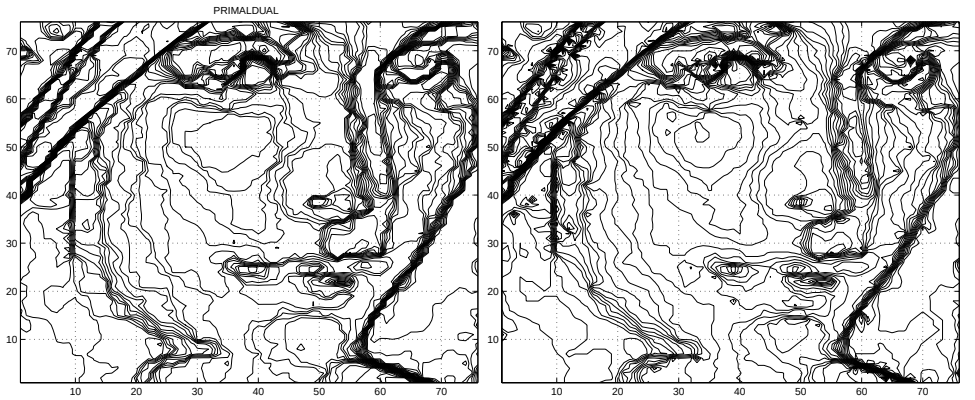


FIG. 6.7. *Left: isointensity contours of part of the image obtained by the primal-dual method. Right: isointensity contours of part of the image obtained by our time evolution model.*

one used for the primal-dual method, and we observed that this value corresponds to a value near the largest we allowed for stability under the timestep restriction. We use $\beta = 14$ which is the standard deviation of the noise corresponding to a signal to noise ratio $SNR = 3$. We also remark that the third order Runge–Kutta method used enhances the diffusion at small scales. The contour plots are shown in Figure 6.7. The numerical steady state obtained with our model is smoother than the one obtained for the TV model, which appears to be staircased. This is because numerically our approximation of the convection term involves hyperbolic upwind ideas.

Our second 2D experiment is a pure deblurring problem. Figure 6.8 (left), corresponds to the original image blurred with Gaussian blur, where $\alpha = 5$ as in (2.4). We remark that we computed the convolution operator j by evolving the 2D heat equation with explicit Euler method in time and central differencing in space with a CFL number of 0.125 in order to ensure homogeneous Neumann boundary conditions. In Figure 6.8 (right), we represent the approximation using our third order Roe's scheme, where we perform 50 iterations with a CFL restriction $\Delta t/\Delta x^2 = 0.1$. We have used $\lambda = 1.5$ (near the maximum value that allows stability for the above CFL restriction) and $\beta = 0.01$, which is small since no noise is present. The isointen-

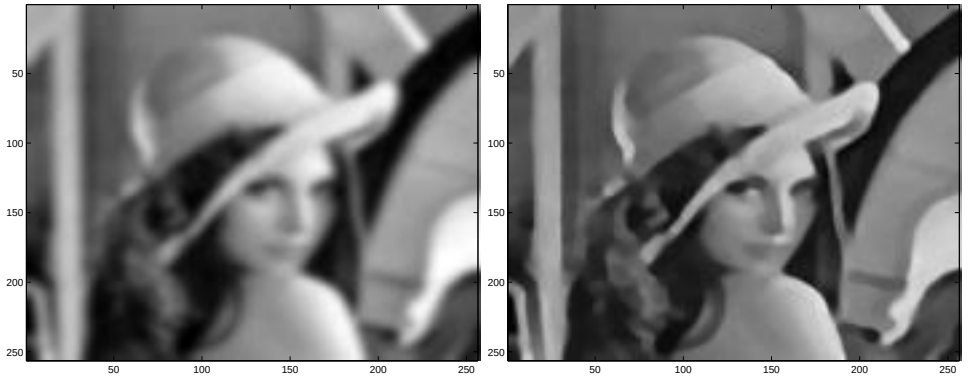


FIG. 6.8. *Left: image blurred with Gaussian blur with* $\alpha = 5$. *Right: image restored with our model, using third order Roe's scheme with* 50 *timesteps and* $\Delta t/\Delta x^2 = 0.1$.

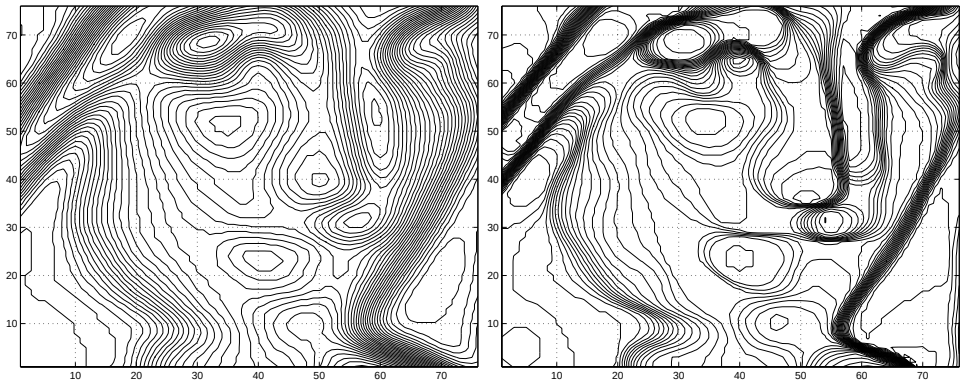


FIG. 6.9. *Left: isointensity contours of part of the blurred image. Right: isointensity contours of part of the image restored by using our time evolution model.*

sity contours showed in Figure 6.9 make clear the edge enhancement obtained through our algorithm.

Our 2D critical experiment was performed on the slightly blurred and noisy image represented in Figure 6.10 (left), with Gaussian blur where $\alpha = 5$ as in (2.4) and $SNR \approx 5$.

We have used $\beta = 7$ (the standard deviation of the noise), and $\lambda = 1.5$, near the largest value allowing stability. We performed 50 iterations with a CFL restriction $\Delta t/\Delta x^2 = 0.1$ using our third order Roe's scheme, obtaining the approximation represented in Figure (6.10, right). Let us observe the denoising and deblurring effect in the isointensity contours picture represented in Figure (6.11).

Finally, we shall include the convergence history of the 1D pure denoising problem experiment presented above. In Figure 6.12 we represent the semilog plot of the $\mathcal{L}^2$-norm of the differences between consecutive iterates vs. the number of iterations and the plot of the evolution of the total variation of the solution. We observe "superlinear" convergence along the first third of the evolution and linear convergence along the remainder. We have obtained similar convergence histories for 1D deblurring and denoising problems. We pointed out that all our experiments were performed with a constant timestep and thus, the computational cost is very low compared with the

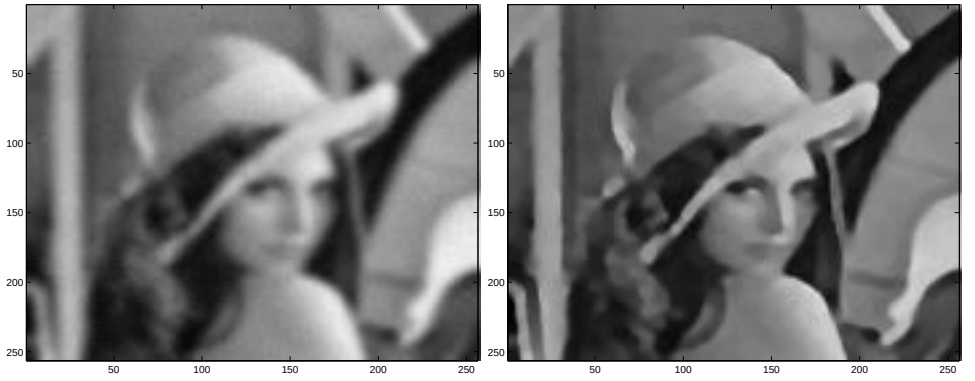


FIG. 6.10. *Left: image blurred with Gaussian blur with* $\alpha = 5$ *and noisy with* $SNR \approx 10$. *Right: image restored with our model, using third order Roe's scheme with* 50 *timesteps and* $\Delta t/\Delta x^2 = 0.1$.

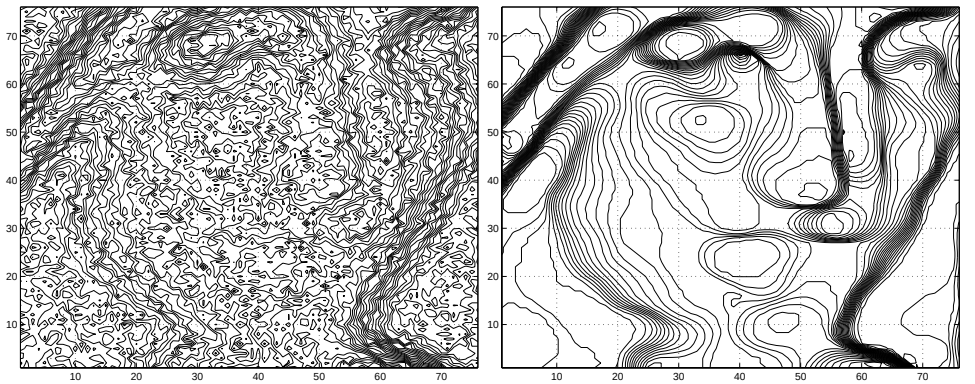


FIG. 6.11. *Left: isointensity contours of part of the blurred and noisy image. Right: isointensity contours of part of the image restored by using our time evolution model.*

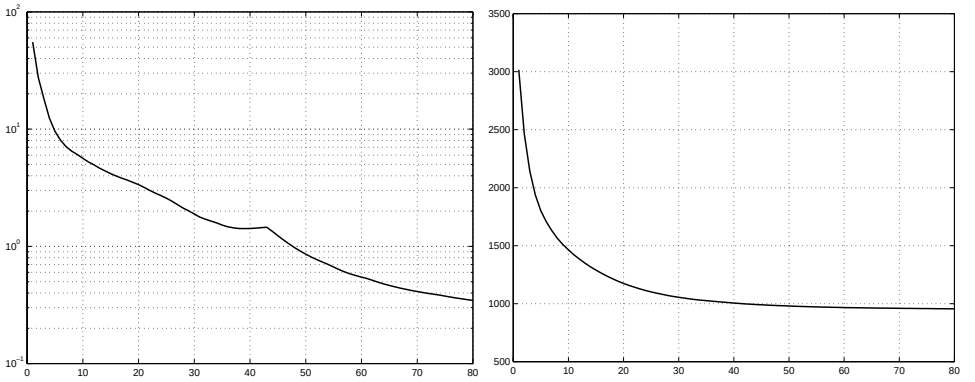


FIG. 6.12. *Left: semilog differences of iterates vs. number of iterations for the pure denoising problem. Right: total variation vs. number of iterations.*

semi-implicit methods. These usually require one third of the number of iterations we needed, but every step of the semi-implicit method requires about five iterations of the preconditioned conjugate gradient method to invert.

7. Concluding remarks. We presented a new time dependent model to solve the nonlinear TV model for noise removal and deblurring together with a very simple explicit algorithm based on Roe's scheme of fluid dymamics. The numerical algorithm is stable with a reasonable CFL restriction, it is easy to program, and it converges quickly to the steady state solution, even for deblurring and denoising problems. The algorithm is fast and efficient since no inversions are needed for deblurring problems with noise. Our time dependent model is based on level set motion that makes the procedure *morphological* and appears to satisfy a maximum principle in the pure denoising case, using as initial guess the noisy image. We also have numerical evidence (through our numerical tests) of this stability in the deblurring case.

REFERENCES

[1] L. Álvarez, F. Guichard, P.-L. Lions, and J.M. Morel, *Axioms and fundamental equations of image processing* Arch. Rational Mech. Anal., 16 (1993), pp. 199–257.

[2] G. Aubert and L. Vese, *A variational method in image recovery*, SIAM J. Numer. Anal., 34 (1997), pp 1948–1979.

[3] P. Blomgren and T.F. Chan, *Modular Solvers for Constrained Image Restoration Problems*, UCLA CAM report 97-52, UCLA, Los Angeles, 1997.

[4] P. Blomgren, T.F. Chan, and P. Mulet, *Extensions to total variation denoising*, Proceedings SPIE 97, San Diego, SPIE, Bellingham, WA, 1997.

[5] A. Chambolle and P.-L. Lions, *Image recovery via total variation minimization and related problems*, Numer. Math., 76 (1997), pp. 167–188.

[6] T. Chan, G. Golub, and P. Mulet, *A nonlinear primal-dual method for total variation-based image restoration*, SIAM J. Sci. Comput., 20 (1999), pp. 1964–1977.

[7] D. Geman and G. Reynolds, *Constrained restoration and the recovery of discontinuities*, IEEE Transaction on Pattern Analysis and Machine Intelligence, 14 (1992), pp. 367–383.

[8] C.W. Groetsch, *The Theory of Tikhonov Regularization for Fredholm Integral Equations of the First Kind*, Pitman, Boston, 1984.

[9] A. Harten, B. Engquist, S. Osher, and S. Chakravarthy, *Uniformly high order accurate essentially non-oscillatory schemes* III, J. Comput. Phys., 71 (1987), pp. 231–303.

[10] R.J. Leveque, *Numerical Methods for Conservation Laws,* Birkhauser-Verlag, Zurich, 1990.

[11] A. Majda, J. McDonough, and S. Osher, *The Fourier method for nonsmooth data,* Math. Comp., 22 (1978), pp. 1041–1081.

[12] A. Marquina, *Local piecewise hyperbolic reconstructions of numerical fluxes for nonlinear scalar conservation laws,* SIAM J. Sci. Comput., 15 (1994), pp. 892-915.

[13] J.G. Nagy and D.P. O'leary, *Restoring images degraded by spatially variant blur*, SIAM J. Sci. Comput., 19 (1998), pp. 1063–1082.

[14] S. Osher and L.I. Rudin, *Feature-oriented image enhancement using shock filters*, SIAM J. Numer. Anal., 27 (1990), pp. 919–940.

[15] S.J. Osher and J.A. Sethian, *Fronts propagating with curvature dependent speed: Algorithms based on a Hamilton-Jacobi formulation*, J. Comput. Phys., 79 (1988), pp. 12–49.

[16] P.L. Roe, *Approximate Riemann solvers, parameter vectors, and difference schemes,* J. Comput. Phys., 43 (1981), pp. 357–372.

[17] J.B. Rosen, *The gradient-projection method for nonlinear programming: Part* II. *Nonlinear constraints,* J. Soc. Indust. Appl. Math., 9 (1961), pp. 514–532.

[18] L. Rudin and S. Osher, *Total variation based image restoration with free local constraints*, in Proceedings IEEE Internat. Conf. Imag. Proc., Austin, TX, IEEE Press, Piscataway, NJ, (1994), pp. 31–35.

[19] L. Rudin, S. Osher, and E. Fatemi, *Nonlinear total variation based noise removal algorithms*, Phys. D, 60 (1992), pp. 259–268.

[20] J.A. Sethian, *Level Set Methods*, Cambridge University Press, London, 1996.

[21] C.W. Shu and S.J. Osher, *Efficient implementation of essentially non-oscillatory shock capturing schemes* II, J. Comput. Phys., 83 (1989), pp. 32–78.

[22] A.N. Tikhonov and V.Y. Arsenin, *Solutions of Ill-Posed Problems*, John Wiley, New York, 1977.
[23] S. Twomey, *On the numerical solution of Fredholm integral equations of the first kind by the inversion of the linear system produced by quadrature*, J. ACM, 10 (1963), pp. 97–107.
[24] B. Van Leer, *Towards the ultimate conservative difference scheme* V. *A second order sequel to Godunov's method,* J. Comput. Phys., 32 (1979), pp. 101–136.
[25] L. Vese, *Variational Problems and PDE's for Image Analysis and Curve Evolution*, Ph.D. thesis, University of Nice, 1996.
[26] C.R. Vogel and M.E. Oman, *Iterative methods for total variation denoising*, SIAM J. Sci. Comput., 17 (1996), pp. 227–238.

SIAM J. SCI. COMPUT.
Vol. 22, No. 2, pp. 406–430
© 2000 Society for Industrial and Applied Mathematics

SCHUR-TYPE METHODS FOR SOLVING LEAST SQUARES PROBLEMS WITH TOEPLITZ STRUCTURE*

HAESUN PARK† AND LARS ELDÉN‡

Abstract. We give an overview of fast algorithms for solving least squares problems with Toeplitz structure, based on generalization of the classical Schur algorithm, and discuss their stability properties. In order to obtain more accurate triangular factors of a Toeplitz matrix as well as accurate solutions for the least squares problems, methods based on corrected seminormal equations (CSNE) can be used. We show that the applicability of the generalized Schur algorithm is considerably enhanced when the algorithm is used in conjunction with CSNE.

Several numerical tests are reported, where different variants of the generalized Schur algorithm and CSNE are compared for their accuracy and speed.

Key words. corrected seminormal equations, displacement representation, downdating, Givens transformations, hyperbolic transformations, least squares problems, QR decomposition, Schur algorithm, seminormal equations, Toeplitz matrix, updating

AMS subject classifications. 65F05, 65F20, 65F35

PII. S1064827598347423

1. Introduction.

We consider the linear least squares (LS) problem

$$\min_x \|Tx - b\|_2, \tag{1.1}$$

where $T \in \mathbf{R}^{m\times n}$, $m \geq n$, is a Toeplitz matrix with $\operatorname{rank}(T) = n$,

$$T = \begin{pmatrix} t_0 & t_{-1} & & \cdots & t_{-n+1} \\ t_1 & t_0 & t_{-1} & & \vdots \\ \vdots & t_1 & t_0 & \ddots & \\ \vdots & & t_1 & \ddots & t_{-1} \\ \vdots & & & \ddots & t_0 \\ \vdots & & & & t_1 \\ \vdots & & & & \vdots \\ t_{m-1} & & & & t_{m-n} \end{pmatrix},$$

and $b \in \mathbf{R}^m$ is an arbitrary vector (b is arbitrary in the sense that neither $[T\ b]$ nor $[b\ T]$ is Toeplitz). Although the results presented in this paper apply for complex Toeplitz problems as well, we will restrict the discussion to real cases for simplicity. Throughout this paper, the notation T will denote an $m \times n$ real Toeplitz matrix.

*Received by the editors December 4, 1998; accepted for publication (in revised form) November 20, 1999; published electronically July 13, 2000.
http://www.siam.org/journals/sisc/22-2/34742.html

†Department of Computer Science and Engineering, University of Minnesota, Minneapolis, MN 55455 (hpark@cs.umn.edu). The research of this author was supported in part by National Science Foundation grants CCR-9209726 and CCR-9509085.

‡Department of Mathematics, Linköping University, S–581 83 Linköping, Sweden (laeld@math.liu.se).

It is well known that the LS problem (1.1) can be solved using the QR decomposition of the matrix T,

$$T = Q \begin{pmatrix} R \\ 0 \end{pmatrix}, \tag{1.2}$$

where $Q \in \mathbf{R}^{m \times m}$ is orthogonal and $R \in \mathbf{R}^{n \times n}$ is upper triangular. To denote the upper triangular factor R for T in (1.2), we will use the notation

$$R = \mathrm{qr}(T). \tag{1.3}$$

The QR decomposition can be computed in $O(mn^2)$ flops (floating point operations, 1 flop $\approx$ 1 addition and 1 multiplication) in general, using, e.g., Householder transformations (see, e.g., [19, Chap. 5]). For a Toeplitz matrix T, several fast algorithms of $O(mn)$ flops exist. The first such algorithm was developed by Sweet [42] taking advantage of the Toeplitz structure and it was improved by Bojanczyk, Brent, and de Hoog [9]. The algorithm due to Chun, Kailath, and Lev-Ari [12] takes advantage of the displacement structure of Toeplitz matrices and it is essentially the same as the algorithm in [9]. It can be shown ([12] and sections 2.1–2.2 in the present paper) that these algorithms are generalizations of a classical algorithm by Schur [21, 36, 37]. In [29], Nagy modified the algorithm presented in [12] to produce R^{-1} instead of R factor in the QR decomposition. He also showed how to compute the vector $Q^T b$ directly instead of computing the full Q factor. For a fast algorithm which is not related to the Schur algorithm but is based on lattice structure and inner product, see [14].

In section 5.2 we will demonstrate that for large Toeplitz LS problems, fast algorithms based on the generalized Schur algorithm are significantly faster than the standard algorithm from LAPACK [2] which is based on Householder transformations. However, the stability properties of the fast algorithms are inferior to those of the standard algorithm [29, 10, 40, 32]. In this paper, we will present several modifications and enhancements that make the fast algorithm more accurate and reliable and investigate stability properties of these fast algorithms that are based on the generalized Schur algorithm. Then several fast Schur-type algorithms are compared by experimentation to identify fast and accurate practical algorithms for Toeplitz LS problems.

Recently, we showed [32] that the accuracy of the upper triangular factor R computed by the generalized Schur algorithm is comparable to that of R computed as the Cholesky factor of $T^T T$, and therefore it is worse than that computed from the stable QR decomposition algorithms such as the Householder or Givens methods. We also showed that in many cases it is possible to *improve the accuracy of R factor* computed by the generalized Schur algorithm by postprocessing the R factor using corrected seminormal equations (CSNE) [5, 3].

Given the upper triangular factor R in the QR decomposition of T, the LS problem (1.1) can be solved using the seminormal equations (SNE)

$$R^T R x = T^T b. \tag{1.4}$$

The SNE method is not recommended in general [5] due to the possible loss of numerical accuracy. However, the SNE method has the advantage that it does not require the Q factor from the QR decomposition (1.2) of T. In addition, when one step of iterative refinement is applied to the SNE (1.4), we obtain the CSNE method, and the accuracy of the solution can be greatly improved [5]. In fact, in all of the existing fast

Toeplitz QR decomposition algorithms, the Q factor is computed based on the similar recursion that is used for computing the R factor, and the orthogonality and accuracy of the Q factor can be poor. It is important to note, however, that the analysis of the SNE and CSNE methods in [5] assumes that R is computed by a backward stable method (e.g., Householder transformations). If R is inaccurately computed, then the applicability of these methods may be severely restricted as shown in [3, sect. 6.6.5].

The rest of the paper is organized as follows. In section 2, the generalization of the Schur algorithm is summarized and several ways for solving Toeplitz LS problems are shown utilizing the generalized Schur algorithm and displacement structure of the LS problems. In section 3, a new forward error analysis for the generalized Schur algorithm is given. Some variants for applying the corrected seminormal equations for improving stability of the fast algorithms are discussed in section 4. Then in section 5, numerical test results comparing computational speed and accuracy of several fast algorithms are presented.

2. Generalized Schur algorithm. In this section, we first review the displacement representation and the generalized Schur algorithm. Then we show that the algorithm can be used for solving the Toeplitz LS problem in several different ways.

Suppose a symmetric positive definite matrix $K \in \mathbf{R}^{n\times n}$ can be represented as

$$K = \sum_{i=1}^{q} U_i^T U_i - \sum_{i=q+1}^{d} U_i^T U_i, \tag{2.1}$$

where $U_i \in \mathbf{R}^{n\times n}$, $1 \le i \le d$, are upper triangular *Toeplitz* matrices. This is called the *displacement representation* of K, and the *displacement rank* is defined to be the minimum value of d that allows such representation [12, 27]. The matrix K that satisfies (2.1) can also be represented as

$$K - Z_n K Z_n^T = \sum_{i=1}^{q} x^{(i)} {x^{(i)}}^T - \sum_{i=q+1}^{d} x^{(i)} {x^{(i)}}^T, \tag{2.2}$$

where Z_n is the lower shift matrix with ones on the first subdiagonal and zeros elsewhere, and the vectors $x^{(i)}$ are related to the matrices U_i by $U_i = \mathrm{up}(x^{(i)})$ [26], [12, Lemma 1], where

$$\mathrm{up}\begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{pmatrix} = \begin{pmatrix} u_1 & u_2 & u_3 & \cdots & \cdots & u_n \\ & u_1 & u_2 & u_3 & \cdots & u_{n-1} \\ & & u_1 & u_2 & \ddots & \\ & & & \ddots & \ddots & \vdots \\ & & & & u_1 & u_2 \\ & & & & & u_1 \end{pmatrix}.$$

According to (2.2), $\mathrm{rank}(K - Z_n K Z_n^T) = d$, which explains the terms displacement structure and displacement rank. A survey of theory and applications of the displacement concept is given in [27], and generalizations are presented in [25].

When K is a symmetric positive definite Toeplitz matrix, it has a displacement representation with $q = 1$ and $d = 2$, and the Cholesky decomposition $K = U^T U$ can be computed by the well-known, original *Schur algorithm* [21, 36, 37]. The relation between the displacement representation and the Schur algorithm is discussed in [24].

The generalized Schur algorithm, summarized in section 2.1, is a fast procedure for computing the upper triangular factor of structured matrices that have a displacement representation. In general, the Cholesky decomposition of a symmetric positive definite matrix $K \in \mathbf{R}^{n\times n}$ with the displacement representation (2.1) can be computed in $O(dn^2)$ flops by a generalization of the Schur algorithm. The algorithm can be used in the following several ways for computing the solution of the LS problem (1.1) with Toeplitz structure:

(1) When $[T\ b]$ (or $[b\ T]$) is Toeplitz, as in linear prediction, the generalized Schur algorithm can be used to compute the triangular factor of $[T\ b]$

$$\begin{pmatrix} R & w \\ 0 & \kappa \end{pmatrix} = \mathrm{qr}([T\ b]), \tag{2.3}$$

where $w \in \mathbf{R}^{n\times 1}$ and $\kappa \in \mathbf{R}$. Then the LS solution is obtained by solving the system $Rx = w$ (a slight modification is needed when $[b\,T]$ is Toeplitz). This method does not require the orthogonal factor.

(2) When T is Toeplitz and $[T\,b]$ is not Toeplitz, the generalized Schur algorithm can be used to compute the triangular factor $R = \mathrm{qr}(T)$ of T. To solve the LS problem, either $Q^T b$ needs to be computed, where Q is the orthogonal factor in the QR decomposition of T [29], or the method of seminormal equations can be used, which does not require the orthogonal factor [3, sect. 6.6.5].

(3) Even when T is Toeplitz but $[T\ b]$ is not Toeplitz, the generalized Schur algorithm can be applied to compute the upper triangular factor of $[T\ b]$ or $[b\ T]$ using a certain displacement representation as shown in section 2.2. Then the LS solution can be obtained by solving a triangular system as in (1) above. No orthogonal factor is needed in this case.

2.1. The generalized Schur algorithm. The generalized Schur algorithm can be derived in many ways. In [9, 12], [27, sect. 3.1], the derivation is based on the *generator* for the displacement representation. Here we prefer to utilize the representation (2.1), since it allows us to use downdating perturbation theory for the error analysis.

We describe the details of the algorithm using a case with $q = 2$ and $d = 4$. The starting point for the derivation is that (2.1) can also be written

$$K = S^T \Lambda S,$$

where

$$S = \begin{pmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \end{pmatrix}, \qquad \Lambda = \begin{pmatrix} I_n & 0 & 0 & 0 \\ 0 & I_n & 0 & 0 \\ 0 & 0 & -I_n & 0 \\ 0 & 0 & 0 & -I_n \end{pmatrix},$$

and I_n denotes the $n \times n$ identity matrix. Now assume that we can find a matrix $\Theta \in \mathbf{R}^{4n\times 4n}$ that satisfies

$$\Theta^T \Lambda \Theta = \Lambda \tag{2.4}$$

such that

$$\Theta S = \begin{pmatrix} U \\ 0 \\ 0 \\ 0 \end{pmatrix}, \tag{2.5}$$

$$S \equiv \begin{pmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \end{pmatrix} = \begin{pmatrix} \alpha_1 & \alpha_2 & \alpha_3 & \alpha_4 \\ & \alpha_1 & \alpha_2 & \alpha_3 \\ & & \alpha_1 & \alpha_2 \\ & & & \alpha_1 \\ 0 & \beta_2 & \beta_3 & \beta_4 \\ & 0 & \beta_2 & \beta_3 \\ & & 0 & \beta_2 \\ & & & 0 \\ 0 & \gamma_2 & \gamma_3 & \gamma_4 \\ & 0 & \gamma_2 & \gamma_3 \\ & & 0 & \gamma_2 \\ & & & 0 \\ 0 & \delta_2 & \delta_3 & \delta_4 \\ & 0 & \delta_2 & \delta_3 \\ & & 0 & \delta_2 \\ & & & 0 \end{pmatrix}, \qquad S' = \begin{pmatrix} U_1' \\ U_2' \\ U_3' \\ U_4' \end{pmatrix} = \begin{pmatrix} \alpha_1 & \alpha_2 & \alpha_3 & \alpha_4 \\ & \alpha_1' & \alpha_2' & \alpha_3' \\ & & \alpha_1' & \alpha_2' \\ & & & \alpha_1' \\ 0 & 0 & \beta_3' & \beta_4' \\ & 0 & 0 & \beta_3' \\ & & 0 & 0 \\ & & & 0 \\ 0 & 0 & \gamma_3' & \gamma_4' \\ & 0 & 0 & \gamma_3' \\ & & 0 & 0 \\ & & & 0 \\ 0 & 0 & \delta_3' & \delta_4' \\ & 0 & 0 & \delta_3' \\ & & 0 & 0 \\ & & & 0 \end{pmatrix}.$$

FIG. 2.1. *Annihilation of the second diagonals of* U_i, $2 \le i \le 4$, *where* $\alpha_i, \beta_i, \gamma_i, \delta_i \in \mathbf{R}$.

where U is upper triangular. Then

$$K = S^T \Lambda S = (\Theta S)^T \Lambda (\Theta S) = U^T U, \tag{2.6}$$

which gives a Cholesky factor of K.

We will now describe how Θ can be constructed as a product of Givens and hyperbolic transformations. For Givens transformations (rotations) and hyperbolic transformations, see, e.g., [20, Chap. 5 and Chap. 12.5]. It is crucial for the accuracy of the Schur algorithm to use "stabilized" variants of hyperbolic transformations; see [8, 39, 41, 29, 10, 40, 32].

By premultiplying S by a sequence of Givens or hyperbolic transformations, selected elements can be annihilated. Let $n = 4$, and define S as in Figure 2.1. First, we apply Givens rotations J_1 and J_2 in the (2,5) and the (9,13) planes, to annihilate β_2 and δ_2, respectively. Due to the Toeplitz structure of each U_i, the same rotations J_1 and J_2 annihilate the entire first superdiagonals in U_2 and U_4, respectively, when applied in appropriate planes, without any extra computation. Next, by a stabilized hyperbolic transformation H_1 in the (2,9) plane, we can annihilate the modified element in the location of γ_2 and annihilate the entire first superdiagonal of the third block without any extra computation. After these steps, we obtain S' shown in Figure 2.1. Due to the structures of U_2', U_3', and U_4', the second row of U_1' will not be modified any further and this becomes the second row of the final matrix U. Since each U_i' has the necessary Toeplitz structure, we can continue the same procedure, now working with the 2×2 submatrices. In general, after $n - 1$ such steps we have obtained an upper triangular matrix U shown in (2.5).

Note that the Givens transformations are applied between blocks 1 and 2, or between blocks 3 and 4, and elementary (two-by-two) hyperbolic transformations are applied between blocks 1 and 3. All Givens transformations J_i and hyperbolic transformations H_i used in this procedure satisfy

$$J_i^T \Lambda J_i = \Lambda, \qquad H_i^T \Lambda H_i = \Lambda,$$

ALGORITHM 1.
Given the displacement representation $K = \sum_{i=1}^{q} U_i^T U_i - \sum_{i=q+1}^{d} U_i^T U_i$, this generalized Schur algorithm computes the Cholesky factor U for K.

(1) $U(1, 1:n) := U_1(1, 1:n)$
(2) **for** $k = 1 : n-1$
(a) $r_1(k : n-1) := U(k, k : n-1)$
(b) $(*$ Update blocks $2, 3, \ldots, q$ to block 1 $*)$
for $i = 2 : q$,
$[r_1(k),\ c,\ s] := \text{givens}(r_1(k), u_{(i)}(k))$
$$\begin{pmatrix} r_1^T(k+1 : n-1) \\ u_{(i)}^T(k+1 : n-1) \end{pmatrix} := \begin{pmatrix} c & s \\ -s & c \end{pmatrix} \begin{pmatrix} r_1^T(k+1 : n-1) \\ u_{(i)}^T(k+1 : n-1) \end{pmatrix}$$
end
(c) $(*$ Update blocks $q+2, \ldots, d$ to block $q+1$ $*)$
for $i = q+2 : d$,
$[u_{(q+1)}(k),\ c,\ s] := \text{givens}(u_{(q+1)}(k), u_{(i)}(k))$
$$\begin{pmatrix} u_{(q+1)}^T(k+1 : n-1) \\ u_{(i)}^T(k+1 : n-1) \end{pmatrix} := \begin{pmatrix} c & s \\ -s & c \end{pmatrix} \begin{pmatrix} u_{(q+1)}^T(k+1 : n-1) \\ u_{(i)}^T(k+1 : n-1) \end{pmatrix}$$
end
(d) $(*$ Downdate block $q+1$ from block 1 $*)$
$[r_b(k),\ c,\ s] := \text{shyp}(r_1(k), u_{(q+1)}(k))$
$$\begin{pmatrix} r_b^T(k+1 : n-1) \\ u_{(q+1)}^T(k+1 : n-1) \end{pmatrix} := \begin{pmatrix} 1 & 0 \\ -c & s \end{pmatrix} \begin{pmatrix} 1/s & -c/s \\ 0 & 1 \end{pmatrix} \begin{pmatrix} r_1^T(k+1 : n-1) \\ u_{(q+1)}^T(k+1 : n-1) \end{pmatrix}$$
(e) $U(k+1, k+1 : n) := r_b(k : n-1)$
end

and therefore the product Θ of all these matrices satisfies (2.4). It now follows from (2.6) that the matrix U is the Cholesky factor of K. The algorithm for arbitrary q and d is analogous and is summarized in Algorithm 1, where the function $\text{givens}(x, y)$ returns $\sqrt{x^2 + y^2}$ and the cosine and sine pair $[c,\ s]$ such that

$$\begin{pmatrix} c & s \\ -s & c \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \sqrt{x^2 + y^2} \\ 0 \end{pmatrix}. \tag{2.7}$$

Similarly, the function $\text{shyp}(x, y)$ returns $\sqrt{x^2 - y^2}$ and the cosine and sine pair $[c,\ s]$ of the stabilized hyperbolic transformations. For details, see [1, 32].

2.2. Displacement representations. We now show the displacement representations for several cases which allow us to utilize the generalized Schur algorithm for solving Toeplitz LS problems:

Case 1: $K = T^T T$;

Case 2: $K_a = [T\ b]^T [T\ b]$, where $[T\ b]$ is not Toeplitz;

Case 3: $K_b = [b\ T]^T [b\ T]$, where $[b\ T]$ is not Toeplitz.

For all cases, we will assume that the Cholesky factor of the matrix K exists. However, in Case 2 we can allow the last diagonal element in the Cholesky factor of K_a to be equal to zero. This situation occurs when the LS problem (1.1) is consistent.

Suppose U is the Cholesky factor for T^TT,

$$T^TT = U^TU, \tag{2.8}$$

and T and U are partitioned as

$$T = \begin{pmatrix} t_0 & f_r^T \\ f_c & T_0 \end{pmatrix} = \begin{pmatrix} T_0 & l_c \\ l_r^T & t_{m-n} \end{pmatrix}, \tag{2.9}$$

where $T_0 \in \mathbf{R}^{(m-1)\times(n-1)}$ is a Toeplitz submatrix of T, $f_r, l_r \in \mathbf{R}^{(n-1)\times 1}$, and $f_c, l_c \in \mathbf{R}^{(m-1)\times 1}$, and

$$U = \begin{pmatrix} u_{11} & u_r^T \\ 0 & U_b \end{pmatrix} = \begin{pmatrix} U_t & u_c \\ 0 & u_{nn} \end{pmatrix}, \tag{2.10}$$

where $U_b, U_t \in \mathbf{R}^{(n-1)\times(n-1)}$ are upper triangular and $u_r, u_c \in \mathbf{R}^{(n-1)\times 1}$. Using these partitionings, the relation (2.8) can be written in two different ways,

$$\begin{pmatrix} t_0^2 + f_c^Tf_c & t_0f_r^T + f_c^TT_0 \\ t_0f_r + T_0^Tf_c & f_rf_r^T + T_0^TT_0 \end{pmatrix} = \begin{pmatrix} u_{11}^2 & u_{11}u_r^T \\ u_{11}u_r & u_ru_r^T + U_b^TU_b \end{pmatrix}, \tag{2.11}$$

and

$$\begin{pmatrix} T_0^TT_0 + l_r{l_r}^T & T_0^Tl_c + t_{m-n}l_r \\ {l_c}^TT_0 + t_{m-n}{l_r}^T & {l_c}^Tl_c + t_{m-n}^2 \end{pmatrix} = \begin{pmatrix} U_t^TU_t & U_t^Tu_c \\ u_c^TU_t & u_c^Tu_c + u_{nn}^2 \end{pmatrix},$$

from which we get

$$\begin{aligned} u_ru_r^T + U_b^TU_b - U_t^TU_t &= (f_rf_r^T + T_0^TT_0) - (T_0^TT_0 + l_rl_r^T) \\ &= f_rf_r^T - l_rl_r^T. \end{aligned} \tag{2.12}$$

For Case 1,

$$\begin{aligned} T^TT - Z_nT^TTZ_n^T &= \begin{pmatrix} u_{11}^2 & u_{11}u_r^T \\ u_{11}u_r & u_ru_r^T + U_b^TU_b \end{pmatrix} - \begin{pmatrix} 0 & 0 \\ 0 & U_t^TU_t \end{pmatrix} \\ &= \begin{pmatrix} u_{11}^2 & u_{11}u_r^T \\ u_{11}u_r & u_ru_r^T + U_b^TU_b - U_t^TU_t \end{pmatrix}, \end{aligned}$$

where Z_n is the lower shift matrix of order n. Using (2.12) we now obtain

$$T^TT - Z_nT^TTZ_n^T = \begin{pmatrix} u_{11} \\ u_r \end{pmatrix}\begin{pmatrix} u_{11} \\ u_r \end{pmatrix}^T + \begin{pmatrix} 0 \\ f_r \end{pmatrix}\begin{pmatrix} 0 \\ f_r \end{pmatrix}^T - \begin{pmatrix} 0 \\ u_r \end{pmatrix}\begin{pmatrix} 0 \\ u_r \end{pmatrix}^T - \begin{pmatrix} 0 \\ l_r \end{pmatrix}\begin{pmatrix} 0 \\ l_r \end{pmatrix}^T. \tag{2.13}$$

From (2.11), u_{11} and u_r can be computed as

$$u_{11} = (t_0^2 + f_c^Tf_c)^{1/2}, \qquad u_r = (1/u_{11})(T_0^Tf_c + t_0f_r).$$

By an inductive argument, we can now give a displacement representation of the form (2.1).

PROPOSITION 2.1 (see [12]). *Assume that $T \in \mathbf{R}^{m\times n}$, $m \geq n$, is a Toeplitz matrix partitioned as in* (2.9) *and* $\operatorname{rank}(T) = n$, *and that T^TT has the Cholesky factor U, partitioned as in* (2.10). *Then we have*

$$U^TU = U_1^TU_1 + U_2^TU_2 - U_3^TU_3 - U_4^TU_4,$$

where

$$U_1 = \mathrm{up}\begin{pmatrix} u_{11} \\ u_r \end{pmatrix}, \quad U_2 = \mathrm{up}\begin{pmatrix} 0 \\ f_r \end{pmatrix}, \quad U_3 = \mathrm{up}\begin{pmatrix} 0 \\ u_r \end{pmatrix}, \quad U_4 = \mathrm{up}\begin{pmatrix} 0 \\ l_r \end{pmatrix}.$$

For Case 2, a displacement representation for $K = [T\ b]^T[T\ b]$ can be derived in several different ways. We present one simple expression which can easily be obtained based on the above displacement representation for T^TT. Let

$$Z_{n+1} = \begin{pmatrix} Z_n & 0 \\ e_n^T & 0 \end{pmatrix} \in \mathbf{R}^{(n+1)\times(n+1)}$$

be the lower shift matrix of order $n+1$. Then

$$[T\ b]^T[T\ b] - Z_{n+1}[T\ b]^T[T\ b]{Z_{n+1}}^T = \begin{pmatrix} T^TT - Z_nT^TTZ_n^T & T^Tb - Z_nT^TTe_n \\ b^TT - e_nT^TTZ^T & b^Tb - e_n^TT^TTe_n \end{pmatrix}. \tag{2.14}$$

A rank 2 matrix of the form

$$\begin{pmatrix} O_{n\times n} & y \\ y^T & \alpha \end{pmatrix}, \qquad \text{where} \qquad y \in \mathbf{R}^{n\times 1}, \tag{2.15}$$

can be represented as a product of two rank 2 matrices as follows:

$$\begin{pmatrix} O_{n\times n} & y \\ y^T & \alpha \end{pmatrix} = \begin{pmatrix} y & y \\ \beta & \gamma \end{pmatrix}\begin{pmatrix} y^T & \beta \\ -y^T & -\gamma \end{pmatrix}, \quad \text{where} \quad \beta = \frac{\alpha+1}{2}, \quad \gamma = \frac{\alpha-1}{2}. \tag{2.16}$$

Combining (2.13) and (2.16), and the fact that the Cholesky factor U for T^TT is the same as the submatrix $R(1:n, 1:n)$ of the Cholesky factor R for $[T\ b]^T[T\ b]$, we have the following result.

PROPOSITION 2.2. *Assume that $T \in \mathbf{R}^{m\times n}$, $m \geq n$, is Toeplitz, $[T\ b] \in \mathbf{R}^{m\times(n+1)}$ is not Toeplitz, and $[T\ b]^T[T\ b]$ has the Cholesky factor R. Then R satisfies*

$$R^TR = \sum_{i=1}^{3} R_i^TR_i - \sum_{i=4}^{6} R_i^TR_i,$$

where

$$R_1 = \mathrm{up}\begin{pmatrix} u_{11} \\ u_r \\ 0 \end{pmatrix}, \quad R_2 = \mathrm{up}\begin{pmatrix} 0 \\ f_r \\ 0 \end{pmatrix}, \quad R_3 = \mathrm{up}\begin{pmatrix} y \\ \beta \end{pmatrix},$$

$$R_4 = \mathrm{up}\begin{pmatrix} 0 \\ u_r \\ 0 \end{pmatrix}, \quad R_5 = \mathrm{up}\begin{pmatrix} 0 \\ l_r \\ 0 \end{pmatrix}, \quad R_6 = \mathrm{up}\begin{pmatrix} y \\ \gamma \end{pmatrix},$$

$y = T^Tb - Z_nT^TTe_n$, $\beta = (\alpha+1)/2$, $\gamma = (\alpha-1)/2$, *and* $\alpha = b^Tb - e_n^TT^TTe_n$.

For Case 3, once the Cholesky factor

$$R_0 = \begin{pmatrix} \kappa & w^T \\ 0 & G \end{pmatrix}, \qquad \text{where} \qquad \kappa \in \mathbf{R}, \qquad w \in \mathbf{R}^{n\times 1}, \qquad G \in \mathbf{R}^{n\times n}, \tag{2.17}$$

of $K_b = [b\ T]^T[b\ T]$ is computed, then the Cholesky factor R for $[T\ b]$ can be easily obtained by using a product J of n Givens rotations to triangularize $\begin{pmatrix} w^T & \kappa \\ G & 0 \end{pmatrix}$ as

$$R = J \begin{pmatrix} w^T & \kappa \\ G & 0 \end{pmatrix}. \tag{2.18}$$

We will derive two different displacement representations that can be used in combination with the generalized Schur algorithm when $K_b = [b\ T]^T[b\ T]$. First, partition Z_{n+1} as

$$Z_{n+1} = \begin{pmatrix} 0 & 0 \\ e_1 & Z_n \end{pmatrix}.$$

Then

$$[b\ T]^T[b\ T] - Z_{n+1}[b\ T]^T[b\ T]Z_{n+1}^T = \begin{pmatrix} b^Tb & b^TT \\ T^Tb & M \end{pmatrix}, \tag{2.19}$$

where

$$M = T^TT - Z_nT^TTZ_n^T - e_1b^Tbe_1^T - e_1b^TTZ_n^T - Z_nT^Tbe_1^T.$$

Letting

$$Z_nT^Tb = \begin{pmatrix} 0 \\ g \end{pmatrix}, \qquad g \in \mathbf{R}^{(n-1)\times 1},$$

we have

$$M = \begin{pmatrix} u_{11}^2 - b^Tb & u_{11}u_r^T - g^T \\ u_{11}u_r - g & f_rf_r^T - l_rl_r^T \end{pmatrix}. \tag{2.20}$$

The following result now follows from (2.13), (2.19), and (2.20).

PROPOSITION 2.3. *Assume that $T \in R^{m\times n}$, $m \geq n$, is Toeplitz, that $[b\ T] \in R^{m\times(n+1)}$ is not Toeplitz, and that $[b\ T]^T[b\ T]$ has the Cholesky factor R_0. Then R_0 satisfies*

$$R_0^TR_0 = \sum_{i=1}^{3} R_i^TR_i - \sum_{i=4}^{6} R_i^TR_i,$$

where

$$R_1 = \mathrm{up}\begin{pmatrix} \sqrt{b^Tb} \\ T^Tb/\sqrt{b^Tb} \end{pmatrix}, \quad R_2 = \mathrm{up}\begin{pmatrix} 0 \\ \beta \\ u_{11}u_r - g \end{pmatrix}, \quad R_3 = \mathrm{up}\begin{pmatrix} 0 \\ 0 \\ f_r \end{pmatrix},$$

$$R_4 = \mathrm{up}\begin{pmatrix} 0 \\ T^Tb/\sqrt{b^Tb} \end{pmatrix}, \quad R_5 = \mathrm{up}\begin{pmatrix} 0 \\ \gamma \\ u_{11}u_r - g \end{pmatrix}, \quad R_6 = \mathrm{up}\begin{pmatrix} 0 \\ 0 \\ l_r \end{pmatrix},$$

$\beta = (u_{11}^2 - b^Tb + 1)/2$, and $\gamma = (u_{11}^2 - b^Tb - 1)/2$.

Now we present another displacement representation that allows us to compute the submatrix G of R_0 by Algorithm 1. Let the matrix G and the vector w of (2.17) be partitioned as

$$\begin{pmatrix} w^T \\ G \end{pmatrix} = \begin{pmatrix} w_1 & w_b^T \\ g_{11} & g_r^T \\ 0 & G_b \end{pmatrix} = \begin{pmatrix} w_t^T & w_n \\ G_t & g_c \\ 0 & g_{nn} \end{pmatrix}, \tag{2.21}$$

ALGORITHM 2.
Given a Toeplitz matrix T of full column rank, and a vector b, this algorithm solves the LS problem $\min_x \|Tx - b\|_2$.

(1) Compute κ, w, and $[g_{11}\ g_r^T]$ of (2.17) and (2.21).
(2) Define the matrices $G_i,\ i = 1, \ldots, 6$, from (2.23), (2.24), and compute G by Algorithm 1.
(3) Transform the matrix $\begin{pmatrix} w^T & \kappa \\ G & 0 \end{pmatrix}$ to upper triangular form by a sequence of n Givens rotations:

$$J \begin{pmatrix} w^T & \kappa \\ G & 0 \end{pmatrix} = \begin{pmatrix} R & c \\ 0 & \kappa' \end{pmatrix}.$$

(4) Solve $Rx = c$.

where $G_b, G_t \in \mathbf{R}^{(n-1)\times(n-1)}$, $g_r, g_c, w_b, w_t \in \mathbf{R}^{(n-1)\times 1}$, and $g_{11}, g_{nn}, w_1, w_n \in \mathbf{R}$. Then

$$G^T G - Z_n G^T G Z_n^T = \begin{pmatrix} g_{11}^2 & g_{11} g_r^T \\ g_{11} g_r & g_r g_r^T + G_b^T G_b - G_t^T G_t \end{pmatrix}.$$

Using the same technique as in Case 1, we then obtain

$$g_r g_r^T + G_b^T G_b - G_t^T G_t = f_r f_r^T + w_t w_t^T - l_r l_r^T - w_b w_b^T. \tag{2.22}$$

The above results are summarized in Proposition 2.4.

PROPOSITION 2.4. *Under the same assumptions as those in Proposition* 2.3*, the submatrix G of the Cholesky factor*

$$R_0 = \begin{pmatrix} \kappa & w^T \\ 0 & G \end{pmatrix}$$

of $[b\ T]^T[b\ T]$ satisfies

$$G^T G = \sum_{i=1}^{3} G_i G_i^T - \sum_{i=4}^{6} G_i G_i^T,$$

where

$$G_1 = \mathrm{up}\begin{pmatrix} g_{11} \\ g_r \end{pmatrix}, \quad G_2 = \mathrm{up}\begin{pmatrix} 0 \\ f_r \end{pmatrix}, \quad G_3 = \mathrm{up}\begin{pmatrix} 0 \\ w_t \end{pmatrix}, \tag{2.23}$$

$$G_4 = \mathrm{up}\begin{pmatrix} 0 \\ g_r \end{pmatrix}, \quad G_5 = \mathrm{up}\begin{pmatrix} 0 \\ l_r \end{pmatrix}, \quad G_6 = \mathrm{up}\begin{pmatrix} 0 \\ w_b \end{pmatrix}. \tag{2.24}$$

To be able to use Algorithm 1 here, first we must compute κ, the vector w, and the first row of G. This can be done efficiently in $O(n)$ operations; see, e.g., [20, Chap. 4]. The pre- and postprocessing that is needed in this case is summarized in Algorithm 2.

We have shown the displacement representations to utilize the generalized Schur algorithm. Note that the displacement representation and displacement rank are

TABLE 2.1
Complexity of the algorithms in flops.

	Total	Preproc.	Gen. Schur	Postproc.	Tri. system solve
Prop. 2.1, SNE	$2mn + 7n^2$	mn	$6n^2$	mn	n^2
Prop. 2.1, CSNE	$4mn + 8n^2$	mn	$6n^2$	$3mn$	$2n^2$
Prop. 2.2	$3mn + 10.5n^2$	$3mn$	$10n^2$	–	$0.5n^2$
Prop. 2.3	$2mn + 12.5n^2$	$2mn$	$10n^2$	$2n^2$	$0.5n^2$
Prop. 2.4	$2mn + 12.5n^2$	$2mn$	$10n^2$	$2n^2$	$0.5n^2$

highly dependent on the structure of the matrix. The displacement rank is not preserved under permutations. For example, when Π is a permutation matrix, the displacement rank of $(T\Pi)^T(T\Pi)$ is not 4 any more, unless $T\Pi$ is also Toeplitz. Also the displacement rank refers to the rank of the matrix $K - ZKZ^T$ *in general*: when T is already upper triangular and Toeplitz, the displacement rank of T^TT is only 1, rather than 4.

The computational complexity of the algorithm for solving the Toeplitz LS problems using each of the displacement representation is summarized in Table 2.1. In the table, the preprocessing refers to the computations to be done to obtain the vectors to be involved in the generalized Schur algorithm. The postprocessing refers to those involved after the triangular factor is obtained, before any triangular system is solved. In the algorithms based on Proposition 2.1, where only the upper triangular matrix U is computed using the generalized Schur algorithm, the LS problem is solved using the SNE and CSNE approach [3, sect. 6.6.5]. In the operation counts we have assumed that the Toeplitz matrix times vector operations are performed by the standard algorithm, requiring mn flops. The complexity of the generalized Schur algorithm with the displacement rank d is $2(d-1)n^2$.

3. Forward error analysis. The forward error analysis given in [32] for the case $q = 2$ and $d = 4$ applies also to the general case. We briefly repeat[1] the derivation from [32] for the special case

$$K = T^TT = U^TU = U_1^TU_1 + U_2^TU_2 - U_3^TU_3 - U_4^TU_4 \tag{3.1}$$

and then derive better results using new perturbation theory according to Chang, Paige, and Stewart [11]. The problem of computing the Cholesky decomposition of a matrix K, based on its displacement representation (3.1), can be considered as that of performing a sequence of updates and downdates, and the analysis of [39] is applicable.

Assume that the computations are performed in a floating point system with unit roundoff μ (see, e.g., [20, p. 61]). Using the error analysis of stabilized hyperbolic transformations [8, 39], it can be shown [32] that the computed matrix $\bar{U}$ satisfies

$$J\begin{pmatrix} U_1 \\ U_2 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} \bar{U} \\ 0 \\ U_3 \\ U_4 \end{pmatrix} + E, \qquad \|E\|_F \le C\mu\|U\|_F + O(\mu^2), \tag{3.2}$$

[1]For simplicity we here omit the treatment of errors in the computation of the elements of U_1 and U_3; see [32].

for some exactly orthogonal matrix J, where C is a polynomial of low degree[2] in the dimensions of T and the displacement rank d. From (3.2) we get

$$\bar{U}^T\bar{U} = U_1^TU_1 + U_2^TU_2 - U_3^TU_3 - U_4^TU_4 + E_c,$$

where

$$\|E_c\|_F \le 2\|E\|_F \,\|\,(\,\bar{U} \quad U_3 \quad U_4\,)\,\|_F + O(\mu^2).$$

Using (3.2) we have $\|\,(\,\bar{U} \quad U_3 \quad U_4\,)\,\|_F \le \|\,(\,U_1 \quad U_2\,)\,\|_F + \|E\|_F$, and since the elements of U_1 and U_2 are from U and T, respectively, we can estimate

$$\|E_c\|_F \le C\mu\,\|U\|_2^2 + O(\mu^2) = C\mu\|T\|_2^2 + O(\mu^2).$$

We can now use the new perturbation result for the Cholesky decomposition [11] to estimate the forward error. First define $W_U \in \mathbf{R}^{n(n+1)/2\times n(n+1)/2}$ as the lower triangular matrix

$$W_U = \begin{pmatrix}
2u_{11} & & & & & & & & & & \\
u_{12} & u_{11} & & & & & & & & & \\
 & 2u_{12} & 2u_{22} & & & & & & & & \\
u_{13} & & & u_{11} & & & & & & & \\
 & u_{13} & u_{23} & u_{12} & u_{22} & & & & & & \\
 & & & 2u_{13} & 2u_{23} & 2u_{33} & & & & & \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & & & & \\
u_{1n} & & & & & & & u_{11} & & & \\
 & u_{1n} & u_{2n} & & & & & u_{12} & u_{22} & & \\
 & & & u_{1n} & u_{2n} & u_{3n} & & u_{13} & u_{23} & u_{33} & \\
 & & & & & & \cdot & \cdot & \cdot & \cdot & \cdot \\
 & & & & & & & 2u_{1n} & 2u_{2n} & 2u_{3n} & \cdot & 2u_{nn}
\end{pmatrix}, \tag{3.3}$$

and put

$$D = \operatorname{diag}(1, \underbrace{\sqrt{2}, 1}_{2}, \ldots, \underbrace{\sqrt{2}, \ldots, \sqrt{2}, 1}_{j}, \ldots, \underbrace{\sqrt{2}, \ldots, \sqrt{2}, 1}_{n}) \in \mathbf{R}^{n(n+1)/2\times n(n+1)/2}.$$

THEOREM 3.1. *Let T and U be defined as in Proposition* 2.1*, and assume that the condition number of T is is not too large* [11]*. Then the approximation $\bar{U}$ computed using the generalized Schur algorithm satisfies*

$$\frac{\|\bar{U} - U\|_F}{\|U\|_2} \le C\mu\kappa_c(T^TT) + O(\mu^2), \tag{3.4}$$

where $\kappa_c(T^TT)$ is the Cholesky condition number [11]

$$\kappa_c(T^TT) = \|(DW_U)^{-1}\|_2\,\|T^TT\|_2^{1/2},$$

which can be bounded as

$$\frac{1}{2}\kappa_2(T) \le \kappa_c(T^TT) \le \frac{1}{\sqrt{2}}\kappa_2^2(T). \tag{3.5}$$

[2]Each occurrence of C will denote a different polynomial.

The theorem is valid also if we take into account the errors in the computation of the elements of U_1 and U_3; cf. [32].

In (3.5), $\kappa_2(A)$ is the usual condition number, measured in the matrix 2-norm [20, p. 57]. It is demonstrated in [11] (see also section 5) that there are matrices T for which $\kappa_c(T^TT)$ is close to the lower bound and others for which $\kappa_c(T^TT)$ is close to the upper bound. Thus, the error estimate (3.4) shows that although the upper triangular factor U is computed by Algorithm 1 without forming T^TT, the error in the computed $\bar{U}$ may in some cases be as large as if it had been computed as the Cholesky factor of T^TT. This is interesting, considering that nowhere in the algorithm was T^TT explicitly formed.

Obviously analogous results can be derived for the general case

$$K = \sum_{i=1}^{q} U_i^T U_i - \sum_{i=q+1}^{d} U_i^T U_i,$$

provided that $\|U_i\|_F$, $i = q+1, \ldots, d$, can be bounded in terms of $\|K\|_F$. In the special case when $K = T$ is a symmetric, positive definite Toeplitz matrix (the displacement rank is 2), we have

$$\frac{\|\bar{U} - U\|_F}{\|U\|_2} \leq C\mu\kappa_c(T) + O(\mu^2),$$

which is the same as for the Cholesky algorithm. In fact, it has been shown in [7] (see also [40]) that in this special case the Schur algorithm is stable even if ordinary "unstabilized" hyperbolic transformations are used.

The error analysis must be modified somewhat for $K_a = [T\ b]^T[T\ b]$. Let the Cholesky decomposition of K_a be

$$K_a = \begin{pmatrix} U^T & 0 \\ z^T & \rho \end{pmatrix} \begin{pmatrix} U & z \\ 0 & \rho \end{pmatrix},$$

where ρ may be equal to zero, and let $V = (U\ z)$. Then the error estimate for $\bar{V}$ becomes

$$\frac{\|\bar{V} - V\|_F}{\|V\|_2} \leq C\mu\hat{\kappa}_c(K_a) + O(\mu^2),$$

where $\hat{\kappa}_c(K_a)$ is a slightly modified condition number,

$$\hat{\kappa}_c(K_a) = \|\hat{W}_U^{-1}\|_2 \|V\|_2^{1/2},$$

and $\hat{W}_U$ is defined by

$$\hat{W}_U = \begin{pmatrix} DW_U & 0 \\ S & U \end{pmatrix}, \quad S = \begin{pmatrix} z_1 & & & & & & & & \\ & z_1 & z_2 & & & & & & \\ & & & z_1 & z_2 & z_3 & & & \\ & & & & & & \cdots & & \\ & & & & & & & z_1 & z_2 & \cdots & z_n \end{pmatrix}.$$

Note that the perturbation of the upper triangular factor U still satisfies the bound (3.5). It is interesting to be able to bound the perturbation of the right-hand side z

separately. Using the same technique as in [11, p. 461], the perturbation of z can be shown to be bounded essentially by $\kappa_2^2(T)$ (cf. the bound (3.5) for κ_c).

A similar analysis can also be made for the algorithms based on Propositions 2.3 and 2.4 for the generalized Schur algorithm applied to $K_b = [b\ T]^T[b\ T]$, and the accuracy will depend on the condition number $\kappa_c(K_b)$. However, when the LS problem is almost consistent, then K_b is very ill-conditioned, and in the extreme case when b is in the range $R(T)$, then K_b is singular. Therefore, the algorithm based on Proposition 2.3 can give large errors, which makes it unsuitable for least squares problems. In the next section we will show that we can modify Algorithm 2 (which is based on Proposition 2.4), using CSNE to refine the upper triangular matrix, in such a way that this algorithm will give accurate results even for almost consistent systems.

4. Improving accuracy by CSNE. In this section we will describe a method based on CSNE which was originally designed [5] for computing the solution of the LS problem when only the R factor in the QR decomposition is available. First, we discuss block downdating using CSNE, and then we describe the application of CSNE block downdating for Toeplitz LS problems. For applying CSNE to Algorithm 1, see [32].

4.1. CSNE for block downdating. Let $A = \begin{pmatrix} H \\ \tilde{A} \end{pmatrix} \in \mathbf{R}^{m\times n}$ with $m \geq n$, where $H \in \mathbf{R}^{k\times n}$, with $k \leq n$. Assume that $\operatorname{rank}(A) = n$. Given the QR decomposition

$$A = Q \begin{pmatrix} R \\ 0 \end{pmatrix},$$

where $Q \in \mathbf{R}^{m\times m}$ is orthogonal and R is upper triangular, the block downdating that we will apply in the next section is to find the upper triangular factor $\tilde{R}$ in the QR decomposition of $\tilde{A}$, using R and A only as data, in no more than $O(mn + n^2)$ flops. The downdating problem for $k = 1$ has been studied in [1, 4, 18, 20, 31, 35, 38] and block downdating for $k > 1$ in [34, 15, 30]. Perturbation and error analysis are given in [38, 16, 17, 39].

Block downdating can be performed by first computing the matrix $Q_1 \in \mathbf{R}^{n\times k}$ from the triangular system $R^TQ_1 = H^T$, the Cholesky factor $\Gamma \in \mathbf{R}^{k\times k}$ of the matrix $I - Q_1^TQ_1$, and then computing the downdated upper triangular factor $\tilde{R}$ as a block in the QR decomposition

$$\begin{pmatrix} Q_1 & R \\ \Gamma & 0 \end{pmatrix} = P \begin{pmatrix} I_k & \hat{H} \\ 0 & \tilde{R} \end{pmatrix},$$

where P is orthogonal. This is the generalization of the LINPACK downdating algorithm [16]. The method of CSNE for LS problems can be applied to the block downdating problem. This is due to the fact that the matrix Γ can be computed as the R factor in the QR decomposition of the residual matrix in the LS problem

$$\min_V \left\| \begin{pmatrix} I_k \\ 0 \end{pmatrix} - AV \right\|_F, \tag{4.1}$$

and we can apply the method of CSNE to (4.1). For details, see [15, 17].

When the matrix A is available in the product form,

$$A = W^TX,$$

ALGORITHM 3. BLOCK DOWNDATING BY CSNE FOR A MATRIX IN PRODUCT FORM. Given the upper triangular factor R for A, $X(= WA)$, and only the first k columns of the orthogonal matrix W, downdate the first k rows of A from R, giving $\tilde{R}$.

(1) Compute Q_1, V, and F from

$$(a)\ R^T Q_1 = X^T W_1, \qquad (b)\ RV = Q_1, \qquad (c)\ F := W_1 - XV.$$

(2) (a) Update Q_1, V, and F:

$$R^T \delta Q_1 = X^T F, \qquad Q_1 := Q_1 + \delta Q_1,$$
$$R\delta V = \delta Q_1, \qquad F := F - X\delta V.$$

(b) Compute the R factor of F to determine Γ:

$$\Gamma = \mathrm{qr}(F).$$

(3) Triangularize $\begin{pmatrix} Q_1 & R \\ \Gamma & 0 \end{pmatrix}$ by a product of Givens rotations P :

$$\begin{pmatrix} I_k & \hat{H} \\ 0 & \tilde{R} \end{pmatrix} := P \begin{pmatrix} Q_1 & R \\ \Gamma & 0 \end{pmatrix}.$$

where $W \in \mathbf{R}^{m\times m}$ is an orthogonal matrix partitioned as

$$W = (\, W_1 \quad W_2 \,), \qquad W_1 \in \mathbf{R}^{m\times k},$$

and $X \in \mathbf{R}^{m\times n}$, the matrix A does not need to be computed explicitly for block downdating. This is because the LS problem

$$\min_V \|W_1 - XV\|_F$$

is equivalent to (4.1) [32]. The block downdating algorithm with CSNE for this case is summarized in Algorithm 3, which will be used in the next subsection.

4.2. CSNE for Algorithm 2. In this subsection, we show how the CSNE method for block downdating can be used to refine the matrix G obtained from Algorithm 2. For this purpose, we rewrite (2.22) as

$$\hat{G}^T \hat{G} = G_t^T G_t + w_t w_t^T + f_r f_r^T, \tag{4.2}$$
$$G_b^T G_b = \hat{G}^T \hat{G} - H^T H, \tag{4.3}$$

where $H^T = (\, l_r \quad g_r \quad w_b \,) \in \mathbf{R}^{(n-1)\times 3}$. Suppose we have $\hat{G}$ from Algorithm 2 (it is easy to see that $\hat{G}$ can be obtained as a byproduct of the algorithm at no extra cost). From (2.22), (2.9), and (2.21), we have

$$\hat{G} = \mathrm{qr}(X), \qquad X = \begin{pmatrix} f_r^T \\ T_0 \\ l_r^T \end{pmatrix} \in \mathbf{R}^{(m+1)\times(n-1)}, \tag{4.4}$$

and the matrix G_b is obtained by downdating the block H from $\hat{G}$. Note that g_r^T and w_b^T are from the first two rows in the upper triangular factor of the matrix $[b\ T]$. We

ALGORITHM 4. SOLUTION OF TOEPLITZ LS PROBLEM USING CSNE REFINEMENT. Given the matrix T and a right-hand side b, solve the LS problem $\min_x \|Tx - b\|_2$.

(1) Compute κ, w, G, and $\hat{G}$ using steps 1–3 of Algorithm 2.
(2) Compute the first three columns of the matrix W (no computation needed for the first column).
(3) Let X be defined by (4.4), and compute refined G_b by downdating the first three rows of $A = W^T X$ using Algorithm 3.
(4) Perform steps 3 and 4 of Algorithm 2.

can find an orthogonal matrix, $\hat{W}^T$, which transforms the first two columns of $[b\ T]$ to upper triangular form, and, accordingly, the first two rows into

$$\begin{pmatrix} \kappa & w_1 & w_b^T \\ 0 & g_{11} & g_r^T \end{pmatrix}.$$

Then H consists of the first three rows of the matrix A defined as

$$A \equiv W^T X = \begin{pmatrix} 0 & 1 \\ \hat{W}^T & 0 \end{pmatrix} \begin{pmatrix} f_r^T \\ T_0 \\ l_r^T \end{pmatrix}.$$

Now, we can apply the CSNE block downdating algorithm considering A as the data matrix for the triangular factor $\hat{G}$ and solve

$$\min_V \left\| \begin{pmatrix} I_3 \\ 0 \end{pmatrix} - AV \right\|_F. \tag{4.5}$$

This problem is equivalent to (cf. subsection 4.1)

$$\min_V \|W_1 - XV\|_F, \qquad W = (W_1\ W_2), \qquad W_1 \in \mathbf{R}^{(m+1)\times 3}. \tag{4.6}$$

Note that the first column of W_1 is $e_{m+1} \in \mathbf{R}^{(m+1)\times 1}$, which is the $(m+1)$th unit vector. The above derivation is summarized in Algorithm 4.

According to Theorem 3.1, the accuracy of the matrix G computed by Algorithm 2 depends on the condition number of G, which may be ill-conditioned even if R (and, equivalently, T) is well-conditioned.

However, in the CSNE method, it is the condition number of $\hat{G}$ defined by (4.2) that determines the final accuracy [17, 32]. We will now demonstrate that if T is well-conditioned, then $\hat{G}$ is well-conditioned, even if G is ill-conditioned. Let $\sigma_i(R)$ denote the singular values of R, and assume that they are ordered $\sigma_1(R) \geq \sigma_2(R) \geq \cdots \geq \sigma_n(R)$.

From (2.18) we see that the upper triangular factor in the QR decomposition of

$$G_0 = \begin{pmatrix} w_t^T \\ G_t \end{pmatrix}$$

is equal to the $(n-1) \times (n-1)$ principal submatrix of R. Therefore, using theory for eigenvalues of symmetric matrices [20, Chap. 8.1], we get

$$\sigma_1(G_0) \leq \sigma_1(R), \quad \sigma_{n-1}(G_0) \geq \sigma_n(R). \tag{4.7}$$

Furthermore, since $\hat{G}^T\hat{G} = G_0^T G_0 + f_r f_r^T$, the smallest singular value of $\hat{G}$ satisfies

$$\sigma_{n-1}(\hat{G}) \geq \sigma_{n-1}(G_0). \tag{4.8}$$

The vector f_r^T is part of the first row of T, which implies

$$\|f_r\|_2 \leq \sigma_1(R),$$

and, using [20, Thm. 8.1.8], we can deduce

$$\sigma_1^2(\hat{G}) \leq \sigma_1^2(G_0) + \|f_r\|_2^2 \leq 2\sigma_1^2(R). \tag{4.9}$$

Combining (4.7)–(4.9), we have

$$\kappa_2(\hat{G}) \leq \sqrt{2}\,\kappa_2(R) = \sqrt{2}\,\kappa_2(T). \tag{4.10}$$

The accuracy of the matrix $\hat{G}$ computed by Algorithm 2 depends essentially on the condition number of $\hat{G}$ and not on that of G [39, 17, 32]. Therefore, if T is well-conditioned, then $\hat{G}$ will be accurately computed by Algorithm 2. Since $\hat{G}$ is accurate, Algorithm 4 will produce an accurate approximation of G [17, 32], which, finally, will lead to an accurate LS solution. In cases where $\hat{G}$ is well-conditioned and T is ill-conditioned, Algorithm 4 will still give a good approximation of G. However, if T is ill-conditioned with three or more small singular values (relative to the largest singular value), this cannot happen because then $\hat{G}$ has at least one small singular value.

5. Numerical tests. In this section, we present test results for the purpose of comparing the accuracy and speed of the various algorithms for Toeplitz LS problems.

5.1. Comparison of accuracy. We have performed numerical tests in MATLAB with IEEE double precision floating point arithmetic with unit round off $\mu \approx 10^{-16}$. The accuracy of the LS solutions computed from the following algorithms are compared:

- Algorithm Cyb: The Q and R factors for T are computed by Cybenko's algorithm [14].
- Algorithm Sq: The R factor and $Q^T b$ are computed by the generalized Schur algorithm as in [29].
- Algorithm Ssne: The R factor is computed by the generalized Schur algorithm (Algorithm 1) and then the solution is computed from the SNE, $R^T R x = T^T b$, cf. [6].
- Algorithm Scsne: The R factor is computed by the generalized Schur algorithm (Algorithm 1) and then the solution is computed from the CSNE.
- Algorithm Rcsne: The R factor is computed by the Toeplitz matrix triangularization algorithm presented in [32] (Algorithm C in that paper, which refines the R factor from Algorithm 1 by CSNE), and then the solution is computed from the CSNE.
- Algorithm Gcsne: The solution vector x is computed by Algorithm 4 presented in this paper.

The relative forward error in the computed solution x,

$$\frac{\|x_c - x\|_2}{\|x_c\|_2},$$

for each method is presented, where x_c is a vector with all of its components equal to $1/\sqrt{n}$. The right-hand-side vector b is computed from $b := Tx_c$. We also made a number of test runs with data where the residual in the LS problem was nonzero. The results were very similar to those that we report here.

An estimate of the backward error was also computed as

$$\begin{aligned} e_b &:= \min\left\{\frac{\|r\|_2}{\|x\|_2}, \sigma_{min}([T\ F])\right\} \\ &= \min\{\|E\|_F : \|(T+E)x - b\|_2 = \min_{x'} \|(T+E)x' - b\|_2\}, \end{aligned}$$

where

$$r = b - Tx, \qquad F = \frac{\|r\|_2}{\|x\|_2}(I - rr^T),$$

and σ_{min} is the smallest singular value of the matrix $[T\ F]$, as in [44]. The relative backward error is then defined as

$$r_b = \frac{e_b}{\|T\|_F}.$$

Test I. The matrix $T \in \mathbf{R}^{5\times 4}$ is similar to that in [33],

$$T = \begin{pmatrix} 1 & 2 & 3 & 1/\epsilon \\ 2 & 1 & 2 & 3 \\ 3 & 2 & 1 & 2 \\ 4 & 3 & 2 & 1 \\ 1/\epsilon & 4 & 3 & 2 \end{pmatrix},$$

and $\epsilon = 10^{-1-i}$, $i = 1, 2, \ldots, 9$, are chosen so that the matrix becomes progressively more ill-conditioned, with condition number $\kappa_2(T)$ ranging from 95 to $9 \cdot 10^9$. The condition number $\kappa_c(T^TT)$ is close to $\kappa_2^2(T)$. The results are presented in Figure 5.1. For reference, we also give the errors obtained by solving the LS problem using Householder transformations.

Test II. In this test, where

$$T = \begin{pmatrix} 1 & t & t & t & \cdots & t \\ 0 & 1 & t & t & \cdots & t \\ 0 & 0 & 1 & t & \cdots & t \\ \vdots & & & \ddots & \ddots & \vdots \\ & & & & 1 & t \\ 0 & \cdots & & & & 1 \end{pmatrix} \in \mathbf{R}^{50\times 50},$$

the algorithms are applied ignoring the fact that the matrix is already triangular. The parameter t was chosen as $t = \epsilon^{-1/4}$ where ϵ is the same as that in Test I. The condition number $\kappa_2(T)$ varied from $4 \cdot 10^3$ to $4 \cdot 10^9$. In this test the dimension of the matrix is too large to allow for the computation of κ_c within reasonable time. The results are shown in Figure 5.2.

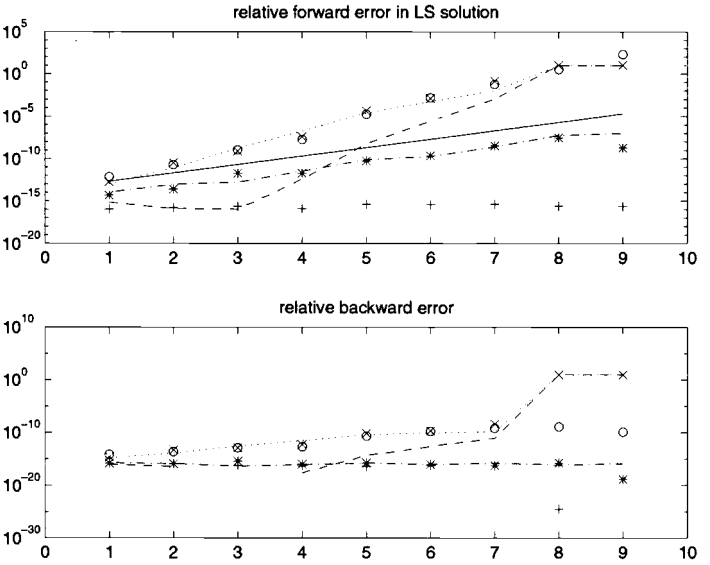


FIG. 5.1. *Test* I. *Relative forward and backward errors for the different algorithms. The following symbols are used: o (Cyb),* $\times$ *(Sq), dotted (Ssne), dashed (Scsne),* $+$ *(Rcsne), dashed-dotted (Gcsne),* $*$ *(Householder). In the upper graph, the solid line represents* $10\mu\kappa_2(T)$*, which can be considered as a reasonable error level for a backward stable algorithm. Note that parts of some curves are missing in the lower plot because the backward error was estimated to be zero. In cases when Algorithm* 1 *broke down, we assigned the value* 10 *to the relative error.*

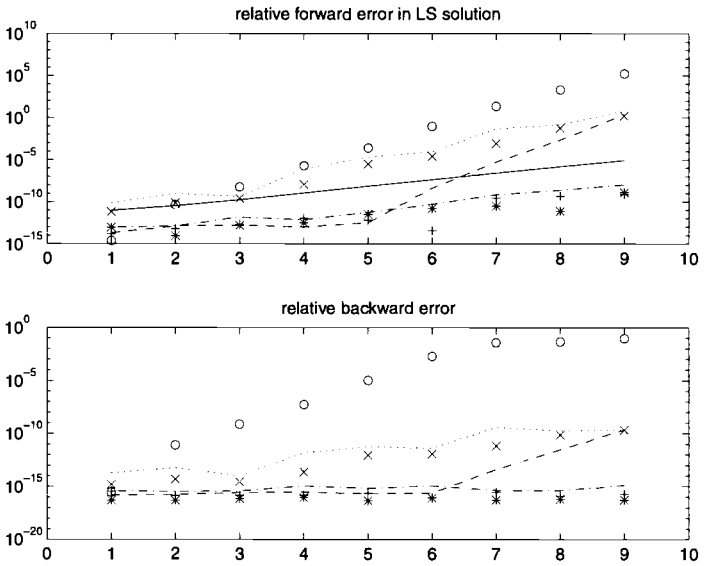


FIG. 5.2. *Test* II.

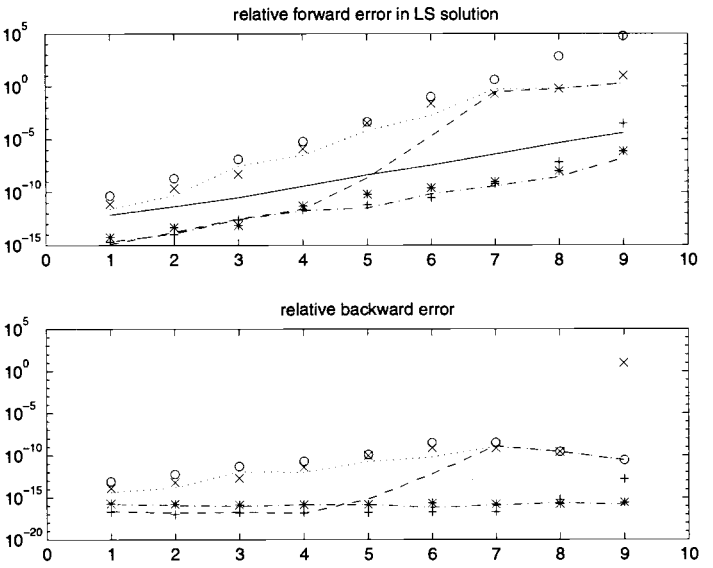


FIG. 5.3. *Test* III.

Test III. This test matrix is a modification of an example from [28]. A set of data was generated as

$$x_n = x(n\Delta t) = \sum_{j=1}^{9} A_j \exp(-\alpha n \Delta t)\cos(\omega_j n \Delta t) + \epsilon r_n,$$

where

$$\begin{aligned} A_j &= j/10, && j = 1, \ldots, 9, \\ \alpha &= \pi/50, && \Delta t = 0.5, \\ \omega_j &= \begin{cases} j\pi/5, & j = 1,2,3,4,5,6,8,9, \\ 7.95\pi/5, & j = 7, \end{cases} \end{aligned}$$

r_n is taken from a normal distribution, $r_n \in N(0,1)$, and ϵ is the same as that in Test I. The matrix $T \in \mathbf{R}^{59\times 19}$ was constructed from $T_{ij} = x_{i-j+19}$. For small values of ϵ, it has one singular value considerably smaller than the rest, and the condition number $\kappa_2(T)$ varied from 340 to $2 \cdot 10^{10}$. In most cases $\kappa_c(T^T T)$ was close to $\kappa_2(T)$. The results are shown in Figure 5.3.

Test IV.

$$T = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ t & 1 & 0 & 0 & 0 \\ 0 & t & 1 & 0 & 0 \\ 0 & 0 & t & 1 & 0 \\ 0 & 0 & 0 & t & 1 \end{pmatrix} \in \mathbf{R}^{5\times 5},$$

where $t = -\log_{10}(\epsilon) + 0.1$ and ϵ is the same as that in Test I. Here $\kappa_2(T)$ varied from 75 to $1 \cdot 10^5$, and $\kappa_c(T^T T)$ was close to $\kappa_2(T)$. The results are shown in Figure 5.4.

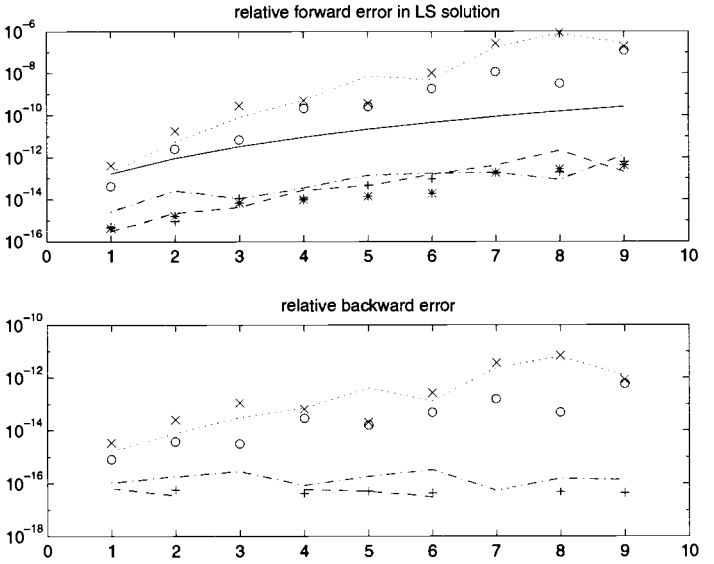


FIG. 5.4. *Test* IV.

Test V. This test matrix is modified from an example in [23]. The matrix is

$$T = \begin{pmatrix} 3 & 2 & 1 & 1 & 2 & 3 \\ 4 & 3 & 2 & 1 & 1 & 2 \\ 5 & 4 & 3 & 2 & 1 & 1 \\ 6 & 5 & 4 & 3 & 2 & 1 \\ 7 & 6 & 5 & 4 & 3 & 2 \\ 8 & 7 & 6 & 5 & 4 & 3 \\ 9 & 8 & 7 & 6 & 5 & 4 \end{pmatrix} + \epsilon * P \in \mathbf{R}^{7\times 6},$$

where P is a random Toeplitz matrix and $\epsilon = 10^{-1-i}$, $i = 1, 2, \ldots, 8$. For $\epsilon = 0$, T is singular, and for the values of ϵ that we chose, $\kappa_2(T)$ varied from $6 \cdot 10^3$ to $8 \cdot 10^{10}$. $\kappa_c(T^T T)$ was much larger than $\kappa_2(T)$. The results are shown in Figure 5.5.

5.2. Comparison of speed. In Table 5.1 we summarize the computational complexity of the algorithms. We give the number of multiplications for computing the LS solution. The fast algorithm by Cybenko [14] is based on the inner products and projections as in a lattice algorithm for linear prediction [13] and it is quite different from any of the other fast algorithms mentioned for the Toeplitz matrix QR decomposition. For a comparison of the complexities of these algorithms, see Nagy [29]. By using the fast Fourier transform it is possible to perform the matrix-vector multiplications for a Toeplitz matrix in $O((m+n)\log(m+n))$ operations (see, e.g., [43, pp. 206–209]), thus reducing the operation count considerably for large n. In the table we have not taken this into account. Of course, on modern computers more factors than operation counts are important for the overall efficiency of algorithms, e.g., the possibility of pipelining vector operations and to parallelize the computations. This will be investigated in our future research.

In Figure 5.6, the actual gain in speed up from using the fast algorithms is illustrated. Here we have used the fast, FFT-based matrix-vector multiplication. The

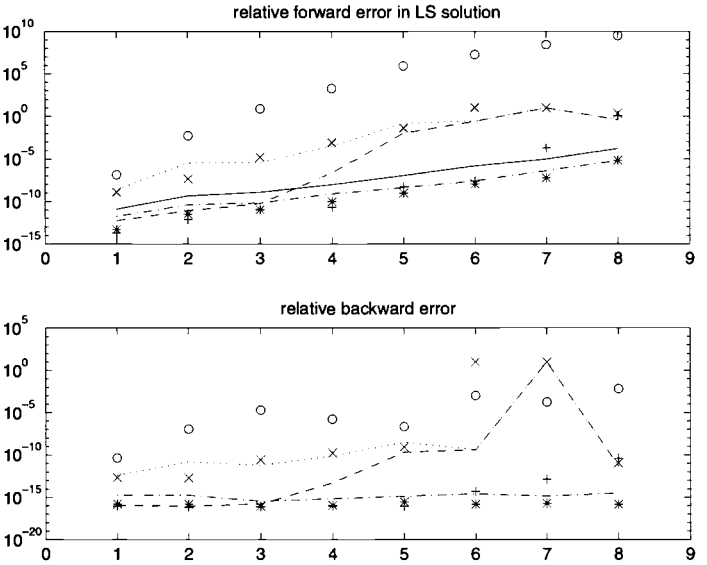


FIG. 5.5. *Test* V.

TABLE 5.1
Computational complexity of the algorithms tested.

Cyb	Sq	Ssne	Scsne	Rcsne	Gcsne
$9mn + 14n^2$	$2mn + 14n^2$	$2mn + 7n^2$	$4mn + 8n^2$	$12mn + 20.5n^2$	$13mn + 24.5n^2$

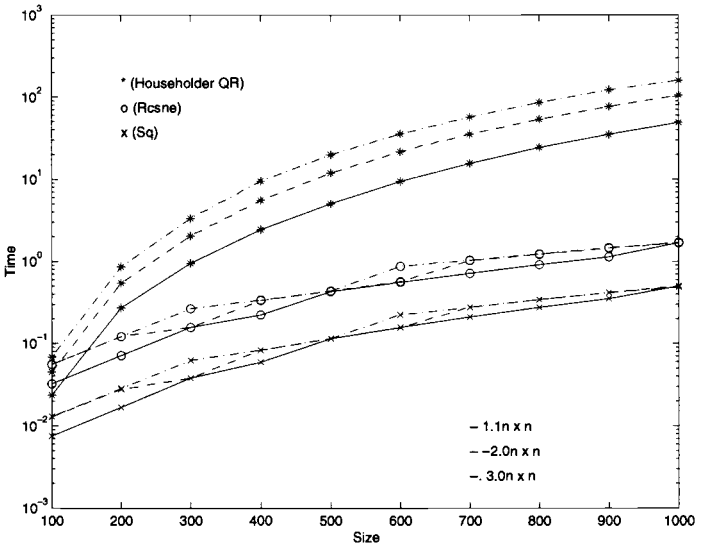


FIG. 5.6. *Execution time in seconds (on SGI Power Challenge) for solving LS problems with* $m \times n$ *Toeplitz matrix, with* $m = 1.1n$ *(solid line),* $m = 2n$ *(dashed line), and* $m = 3n$ *(dash dot line).*

tests were performed on an SGI Power Challenge in the Department of Computer Science and Engineering at the University of Minnesota in Minneapolis. Each output is an average of 20 runs and the test Toeplitz matrices were generated by specifying the first row and the first column, using random numbers. The matrices with column dimensions of $n = 100i$ with $i = 1, 2, \ldots, 10$ were tested. Also for each column dimension n, three different row dimensions of $1.1n$, $2n$, and $3n$ were used. We compared the Householder QR decomposition method with BLAS3 and the two fast algorithms, Algorithm Sq which is the fastest according to the floating operation count and Algorithm Rcsne which is the most accurate. As we can see in Figure 5.6, the gain in speed from using the fast algorithms is significant even for small problems. From the smallest problems of size 110×100, Algorithm Sq was already faster than the Householder QR by a factor of over 3. Then for the problems of size 3000×1000, Algorithm Sq and Algorithm Rcsne were faster than the Householder QR by factors of over 330 and 100, respectively. This test results show that the fast algorithms for Toeplitz LS problems are indeed faster than the slow counterpart in actual implementation.

6. Conclusions and remarks. We have studied fast direct algorithms for the solutions of Toeplitz LS problems. According to our numerical test results, algorithms Cyb, Sq, and Ssne consistently give higher errors than the others, even for relatively well-conditioned problems. For the Sq algorithm, the error estimate (3.4) (cf. also [32]) shows that the accuracy of R may depend on $\kappa_2^2(T)$, and therefore in general it is not accurate enough to qualify as a QR decomposition method. Even if we have no error analysis for the computation of $Q^T b$, its accuracy seems to be comparable to that of R. Since in algorithm Ssne both the computation of R and the solution of the SNE depend on $\kappa_2^2(T)$, this method is not expected to be more accurate than Algorithm Sq.

For well-conditioned problems the Scsne algorithm is about as accurate as the Rcsne and new algorithms. However, as is indicated in the error analysis for Algorithm 1 (see (3.4)), for more ill-conditioned problems, the R factor given by Algorithm 1 is not accurate enough for the assumptions of the theory in [5], [3, sect. 6.6.5] to hold. Therefore, for ill-conditioned problems, the CSNE method cannot make up for the imprecision of R, and the Scsne algorithm is not as accurate as the Rcsne and Gcsne algorithms.

By refining the R factor before the CSNE method is applied to the solution of the LS problem, the Rcsne method works better for more ill-conditioned problems than the Scsne algorithm. The Gcsne algorithm is similar to the Rcsne method in that the computed R factor is refined by the CSNE method. We have tested the fast algorithms on random data and numerous other difficult test problems, and the Rcsne and Gcsne algorithms consistently outperformed the others. However, if the matrix T is ill-conditioned with several small singular values (relative to the largest), then CSNE refinement of the R factor may not work well.

Based on our numerical experience, we expect that a hybrid method that avoids CSNE refinement of the upper triangular matrix in well-conditioned cases might be an accurate and less expensive alternative. Note that if the CSNE step is skipped in the new algorithm, then the operation count is reduced to $2mn + 12.5n^2$. A hybrid method can also be based on Scsne and Rcsne, which estimates the conditioning of R produced by Algorithm 1 and refines it (as in Rcsne) if necessary. More research is needed for testing the choice of tolerances and for implementing the algorithm for larger problems before such a method can be considered a reliable and robust algorithm in practice.

Recently Gu [22] published an algorithm for Toeplitz LS problems, based on the fact that a Toeplitz matrix is a Cauchy-like matrix. However, the operation count for that method is considerably higher than for the methods in the present paper.

Our current research also involves the comparison of the direct fast methods and the iterative methods for the Toeplitz LS problems. The iterative methods are promising since the Toeplitz matrix-vector multiplication can be performed by the FFT and also they do not use the recursions that the fast direct methods rely on. The possibility of circulant preconditioners may improve the speed further to make the iterative methods as fast as or faster than the fast direct methods.

REFERENCES

[1] S. T. Alexander, C.-T. Pan, and R. J. Plemmons, *Analysis of a recursive least squares hyperbolic rotation algorithm for signal processing*, Linear Algebra Appl., 98 (1988), pp. 3–40.

[2] E. Anderson, Z. Bai, C. H. Bischof, J. W. Demmel, J. J. Dongarra, J. J. Du Croz, A. Greenbaum, S. J. Hammarling, A. McKenney, S. Ostrouchov, and D. C. Sorensen, *LAPACK Users' Guide, Release* 2.0, 2nd ed., SIAM, Philadelphia, PA, 1995.

[3] Å. Björck, *Numerical Methods for Least Squares Problems*, SIAM, Philadelphia, PA, 1996.

[4] Å. Björck, H. Park, and L. Eldén, *Accurate downdating of least squares solutions*, SIAM J. Matrix Anal. Appl., 15 (1994), pp. 549–568.

[5] Å. Björck, *Stability analysis of the method of semi-normal equations for least squares problems*, Linear Algebra Appl., 88/89 (1987), pp. 31–48.

[6] A. W. Bojanczyk, R. P. Brent, and F. R. de Hoog, *Stability analysis of a general Toeplitz systems solver*, Numer. Algorithms, 10 (1995), pp. 225–244.

[7] A. W. Bojanczyk, R. P. Brent, F. R. de Hoog, and D. R. Sweet, *On the stability of the Bareiss and related Toeplitz factorization algorithms*, SIAM J. Matrix Anal. Appl., 16 (1995), pp. 40–57.

[8] A. W. Bojanczyk, R. P. Brent, P. van Dooren, and F. R. de Hoog, *A note on downdating the Cholesky factorization*, SIAM J. Sci. Statist. Comput., 8 (1987), pp. 210–221.

[9] A. W. Bojanczyk, R. P. Brent, and F. R. de Hoog, *QR factorization of Toeplitz matrices*, Numer. Math., 49 (1986), pp. 81–94.

[10] S. Chandrasekaran and A. H. Sayed, *Stabilizing the generalized Schur algorithm*, SIAM J. Matrix Anal. Appl., 17 (1996), pp. 950–983.

[11] X.-W. Chang, C. C. Paige, and G. W. Stewart, *New perturbation analyses for the Cholesky factorization*, IMA J. Numer. Anal., 16 (1996), pp. 457–484.

[12] J. Chun, T. Kailath, and H. Lev-Ari, *Fast parallel algorithms for QR and triangular factorization*, SIAM J. Sci. Statist. Comput., 8 (1987), pp. 899–913.

[13] G. Cybenko, *A general orthogonalization technique with applications to time series analysis and signal processing*, Math. Comp., 40 (1983), pp. 323–336.

[14] G. Cybenko, *Fast Toeplitz orthogonalization using inner products*, SIAM J. Sci. Statist. Comput., 8 (1987), pp. 734–740.

[15] L. Eldén and H. Park, *Block downdating of least squares solutions*, SIAM J. Matrix Anal. Appl., 15 (1994), pp. 1018–1034.

[16] L. Eldén and H. Park, *Perturbation analysis for block downdating of a Cholesky decomposition*, Numer. Math., 68 (1994), pp. 457–467.

[17] L. Eldén and H. Park, *Perturbation and error analyses for block downdating of a Cholesky decomposition*, BIT, 36 (1996), pp. 230–246.

[18] G. H. Golub and G. P. Styan, *Numerical computations for univariate linear models*, J. Statist. Comput. Simulation, 2 (1973), pp. 253–272.

[19] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 2nd ed., Johns Hopkins Press, Baltimore, MD, 1989.

[20] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 3rd ed., Johns Hopkins Press, Baltimore, MD, 1996.

[21] J. Götze and H. Park, *Schur-type Methods in Linear Algebra*, Technical report, Department of Electrical and Computer Engineering, Rice University, Houston, TX, 1995.

[22] M. Gu, *New fast algorithms for structured linear least squares problems*, SIAM J. Matrix Anal. Appl., 20 (1999), pp. 244–269.

[23] P. C. Hansen and H. Gesmar, *Fast orthogonal decomposition of rank-deficient Toeplitz matrices*, Numer. Algorithms, 4 (1993), pp. 151–166.
[24] T. Kailath, *A Theorem of* I. *Schur and its Impact on Modern Signal Processing*, in Operator Theory: Advances and Applications, I. Gohberg, ed., Birkhäuser-Verlag, Basel, 1986, pp. 9–30.
[25] T. Kailath and J. Chun, *Generalized displacement structure for block-Toeplitz, Toeplitz-block, and Toeplitz-derived matrices*, SIAM J. Matrix Anal. Appl., 15 (1994), pp. 114–128.
[26] T. Kailath, S.-Y. Kung, and M. Morf, *Displacement ranks of matrices and linear equations*, J. Math. Anal. Appl., 68 (1979), pp. 395–407.
[27] T. Kailath and A. H. Sayed, *Displacement structure: Theory and applications*, SIAM Rev., 37 (1995), pp. 297–386.
[28] W. Kolbel and H. Schafer, *Improvement and automation of the LPSVD algorithm by continuous regularization of the singular values*, J. Magnetic Resonance, 100 (1992), pp. 598–603.
[29] J. G. Nagy, *Fast inverse QR factorization for Toeplitz matrices*, SIAM J. Sci. Comput., 14 (1993), pp. 1174–1193.
[30] S. J. Olszanskyj, J. L. Lebak, and A. W. Bojanczyk, *Rank-k modification methods for recursive least squares problems*, Numer. Algorithms, 7 (1994), pp. 325–354.
[31] C.-T. Pan and R. J. Plemmons, *Least squares modifications with inverse factorizations: Parallel implications*, J. Comput. Appl. Math., 27 (1989), pp. 109–127.
[32] H. Park and L. Eldén, *Stability analysis and fast algorithms for triangularization of Toeplitz matrices*, Numer. Math., 76 (1997), pp. 383–402.
[33] S. Qiao, *Hybrid algorithm for fast Toeplitz orthogonalization*, Numer. Math., 53 (1988), pp. 351–366.
[34] C. Rader and A. Steinhardt, *Hyperbolic Householder transformations*, IEEE Trans. Acoust. Speech Signal Process., 34 (1986), pp. 1589–1602.
[35] M. A. Saunders, *Large-Scale Linear Programming Using the Cholesky Factorization*, Technical Report CS252, Computer Science Department, Stanford University, 1972.
[36] I. Schur, *Über Potenzreihen, die im Inneren des Einheitskreises beschränkt sind*. I, J. Reine Angew. Math., 147 (1917), pp. 205–232.
[37] I. Schur, *On power series which are bounded in the interior of the unit circle*. I., in Operator Theory: Advances and Applications, I. Gohberg, ed., Birkhäuser-Verlag, Basel, 1986, pp. 31–59.
[38] G. W. Stewart, *The effects of rounding error on an algorithm for downdating a Cholesky factorization*, J. Inst. Math. Appl., 23 (1979), pp. 203–213.
[39] G. W. Stewart, *On the stability of sequential updates and downdates*, IEEE Trans. Signal Process., 43 (1995), pp. 2642–2648.
[40] M. Stewart and P. Van Dooren, *Stability issues in the factorization of structured matrices*, SIAM J. Matrix Anal. Appl., 18 (1997), pp. 104–118.
[41] M. Stewart and G. W. Stewart, *On hyperbolic triangularization: Stability and pivoting*, SIAM J. Matrix Anal. Appl., 19 (1997), pp. 847–860.
[42] D. R. Sweet, *Fast Toeplitz orthogonalization*, Numer. Math., 43 (1984), pp. 1–21.
[43] C. Van Loan, *Computational Frameworks for the Fast Fourier Transform*, SIAM, Philadelphia, PA, 1992.
[44] B. Waldén, R. Karlsson, and J. Sun, *Optimal backward perturbation bounds for the linear least squares problem*, Numer. Linear Algebra Appl., 2 (1995), pp. 271–286.

SIAM J. SCI. COMPUT.
Vol. 22, No. 2, pp. 431–448
© 2000 Society for Industrial and Applied Mathematics

LOCALLY ADAPTED TETRAHEDRAL MESHES USING BISECTION*

DOUGLAS N. ARNOLD†, ARUP MUKHERJEE‡, AND LUC POULY§

Abstract. We present an algorithm for the construction of locally adapted conformal tetrahedral meshes. The algorithm is based on bisection of tetrahedra. A new data structure is introduced, which simplifies both the selection of the refinement edge of a tetrahedron and the recursive refinement to conformity of a mesh once some tetrahedra have been bisected. We prove that repeated application of the algorithm leads to only finitely many tetrahedral shapes up to similarity, and we bound the amount of additional refinement that is needed to achieve conformity. Numerical examples of the effectiveness of the algorithm are presented.

Key words. bisection, tetrahedral meshes, adaptive refinement, similarity classes, finite elements

AMS subject classification. 65N50

PII. S1064827597323373

1. Introduction.

The generation of locally adapted conforming tetrahedral meshes is an important component of many modern algorithms, for example, in the finite element solution of partial differential equations. Typically, such meshes are produced by starting with a coarse tetrahedral mesh, selecting certain elements for refinement, somehow refining those elements and others as necessary to maintain conformity, and then possibly repeating this process one (or more than one) time. In this paper we present a simple algorithm for this purpose, and we analyze its behavior. In particular, we consider the question of the shape of tetrahedra that may arise from repeated application of our algorithm and show in section 4 that only a fixed finite number of dissimilar tetrahedra ever arise. A fortiori, the tetrahedral shape cannot degenerate as the mesh is refined in the sense that all the solid angles of all the descendants remain bounded below by a positive constant depending on the tetrahedra in the initial mesh.

The basic refinement step in our algorithm is tetrahedral bisection as in Figure 1. When bisecting a tetrahedron, a particular edge—called the *refinement edge*—is selected for the new vertex. As new tetrahedra are constructed in the course of generating an adapted mesh, their refinement edges must be selected carefully; otherwise element shapes may degenerate. A key aspect of any bisection algorithm is the selection of the refinement edge.

To refine a given conforming mesh we first bisect those tetrahedra that have been selected for refinement. This will usually lead to a nonconforming mesh (a mesh in which neighboring elements do not meet face-to-face). We then apply a recursive procedure to further refine until a conforming mesh is produced. Since it is not obvious

*Received by the editors June 23, 1997; accepted for publication November 24, 1998; published electronically July 13, 2000.
http://www.siam.org/journals/sisc/22-2/32337.html
†Department of Mathematics, Penn State University, University Park, PA 16802 (dna@math.psu.edu). The work of this author was supported by NSF grants DMS-9500672 and DMS-9870399.
‡Department of Mathematics-Hill Center, Rutgers—The State University of New Jersey, Piscataway, NJ 08854-8019 (arup@math.rutgers.edu).
§Department of Mathematics, Swiss Federal Institute of Technology, CH-1015 Lausanne, Switzerland (Luc.Pouly@elca.ch). Current address: ELCA Informatique SA, CH-1000 Lausanne 13, Switzerland. The work of this author was supported by the Swiss National Research Foundation.

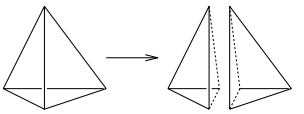


FIG. 1. *Bisection of a single tetrahedron.*

that this procedure will terminate in finitely many steps, in section 3 we provide a rigorous proof that this is the case and establish bounds on the number of steps.

Besides bisection, tetrahedra may be subdivided by octasection. This approach has been studied by Zhang [13] and Ong [10] to obtain uniformly refined meshes. However, octasection cannot be used alone to produce locally adapted conforming meshes. Motivated by work in two dimensions, where local quadrisection has been successfully combined with bisection to obtain conformity (cf., for example, Bank's code PLTMG [2]), Bey [4] has studied the use of local octasection combined with a variety of supplemental refinement strategies which are used to obtain conformity. By contrast, bisection can be used alone to create locally refined conforming meshes, which adds to the simplicity of its implementation. A further advantage of bisection over octasection is that it potentially allows finer control over mesh size. If an error indicator flags an element for refinement, with bisection the element can be divided to reduce its volume by a factor of 2, 4, ... , as required, while with octasection it is not possible to reduce element volume by a factor less than 8.

A number of other authors have proposed bisection-based algorithms for the refinement of tetrahedral meshes. In the pioneering paper [3], Bänsch presented an algorithm for local tetrahedral mesh refinement and discussed element shape degeneration. Although our algorithm appears quite different from Bänsch's, it is essentially equivalent, as we shall discuss in section 2. Another paper which influenced our work is that of Maubach [7]. Maubach considered the question of assigning refinement edges to successive bisections of a single simplex in an arbitrary number of dimensions. His algorithm cannot be easily applied to generate conforming adapted meshes, except for quite special initial meshes. For successive refinement of a single tetrahedron, we establish the precise relationship between our method and his in section 4. We show that his method generates only finitely many similarity classes of simplices (in n dimensions), and from this we deduce the same result for our algorithm. Liu and Joe [6] also study local refinement by bisection. Their algorithm, which is relatively complicated to state and to analyze, is in fact closely related to Bänsch's and therefore to ours. The relationship is discussed in section 2. Liu and Joe [5] prove that their algorithm generates only finitely many similarity classes, although their bound exceeds our sharp one by a large factor. A quite different approach to tetrahedral bisection has been pursued by Rivara [11] and Rivara and Levin [12]. They always use the longest edge of a tetrahedron as the refinement edge. In two dimensions, this approach leads to a finite number of similarity classes, although the number may be arbitrarily large, depending on the initial triangle [1]. As far as we know, the question of finiteness of similarity classes or even of element degeneration for longest edge bisection remains open in three dimensions.

A new aspect of our work is a data structure, which we name *marked tetrahedron*, used to store a geometric tetrahedron together with information necessary to choose its refinement edge and that of its descendants. This data structure is small—it contains just a little additional information beyond the vertices of the tetrahedron—

and it allows us to describe the bisection algorithm simply. Moreover, the marked tetrahedron data structure is useful for insuring mesh conformity. Any conforming tetrahedral mesh can be marked to yield a conforming mesh of marked tetrahedra, and therefore our algorithm does not require any restriction on the initial mesh.

2. Bisection of a single tetrahedron. In this section we describe the marked tetrahedron data structure and present the algorithm `BisectTet`. `BisectTet` bisects a marked tetrahedron by introducing a new vertex at the midpoint of the refinement edge and joining it to the two vertices of the original tetrahedron that do not lie on the refinement edge. It also marks the children (for use in further refinement).

To define a *marked tetrahedron* we introduce some terminology. For a tetrahedron τ, let $\mathcal{V}(\tau)$, $\mathcal{E}(\tau)$, and $\mathcal{F}(\tau)$ denote the set of its vertices, edges, and faces, respectively. For $\varphi \in \mathcal{F}(\tau)$, $\mathcal{E}(\varphi)$ denotes the edges contained in φ. Once a particular edge has been specified as the refinement edge of τ, the two faces that intersect at the refinement edge are called its *refinement faces*. For a marked tetrahedron we specify not only the refinement edge, but also a particular edge of each of the two nonrefinement faces. These are called the *marked edges* of these faces, and we take the refinement edge itself as the marked edges of the two refinement faces. Each marked tetrahedron is also assigned a boolean *flag*. The flag is always unset unless the marked edges of the four faces are all coplanar (we call this a *planar* marked tetrahedron), in which case the flag may or may not be set.

More precisely, a marked tetrahedron τ is a 4-tuple

$$\big(\mathcal{V}(\tau), r_\tau, (m_\varphi)_{\varphi\in\mathcal{F}(\tau)}, f_\tau\big),$$

where

$\mathcal{V}(\tau)$ is a set of four noncoplanar points in $\mathbb{R}^3$, the vertices of τ;
$r_\tau \in \mathcal{E}(\tau)$ is the refinement edge of τ;
$m_\varphi \in \mathcal{E}(\varphi)$ is the marked edge of φ, with $m_\varphi = r_\tau$ if $r_\tau \subset \varphi$;
$f_\tau \in \{0, 1\}$ is the flag, with $f_\tau = 0$ if τ is not planar.

Each marked nonrefinement edge of a marked tetrahedron is either adjacent or opposite to the refinement edge. Thus, we can classify marked tetrahedra into types as follows (cf. Figure 2).

- Type P, planar: the marked edges are coplanar. A type P tetrahedron is further classified as type P_f or type P_u, according to whether its flag is set or not, respectively.
- Type A, adjacent: the marked edges intersect the refinement edge, but are not coplanar.
- Type O, opposite: the marked edges of the nonrefinement faces do not intersect the refinement edge. In this case, a pair of opposite edges are marked in the tetrahedron: one as the refinement edge, and the other as the marked edge of the two nonrefinement faces intersecting there.
- Type M, mixed: the marked edge of just one of the nonrefinement faces intersects the refinement edge.

When a tetrahedron τ is bisected to create children τ_1 and τ_2, a face $\varphi \in \mathcal{F}(\tau_i)$ is called an *inherited face* if $\varphi \in \mathcal{F}(\tau)$, a *cut face* if $\varphi \subsetneqq \varphi'$ for some $\varphi' \in \mathcal{F}(\tau)$, and a *new face* otherwise. Each child has one inherited face, two cut faces, and one new face, which is common to both children. Cf. Figure 3. We are now ready to state `BisectTet`.

ALGORITHM $\{\tau_1, \tau_2\} =$ `BisectTet`(τ)
input: A marked tetrahedron τ

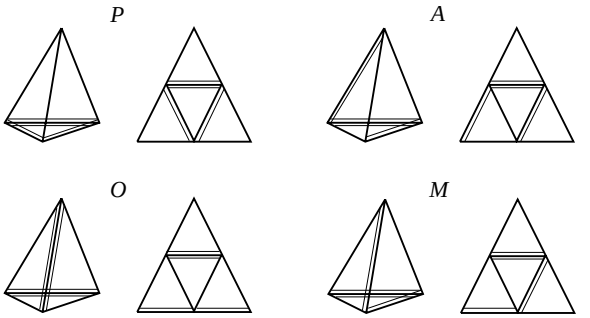


FIG. 2. *The four types of marked tetrahedra: P, A, O, and M. Each marked edge is indicated by a double line and the refinement edge is marked for both faces containing it. Each tetrahedron is shown in three dimensions and cut open and unfolded into two dimensions.*

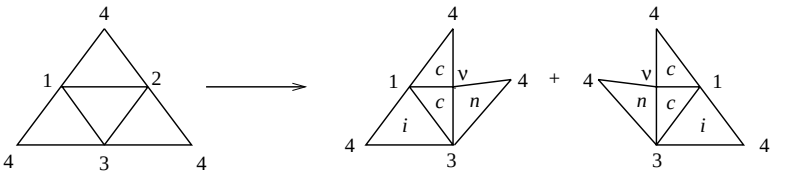


FIG. 3. *Typical bisection of a tetrahedron with the new vertex marked ν, cut faces marked c, inherited faces marked i, and new face marked n.*

output: Marked tetrahedra τ_1 and τ_2

1. Bisect τ by joining the midpoint of its refinement edge to each of the two vertices not lying on the refinement edge. This defines $\mathcal{V}(\tau_i)$ for $i = 1$ and 2.

Mark the faces of the children as follows.

2. The inherited face inherits its marked edge from the parent, and this marked edge is the refinement edge of the child.

3. On the cut faces of the children mark the edge opposite the new vertex with respect to the face.

4. The new face is marked the same way for both children. If the parent is type P_f, the marked edge is the edge connecting the new vertex to the new refinement edge. Otherwise it is the edge opposite the new vertex.

5. The flag is set in the children if and only if the parent is type P_u.

The algorithm is illustrated in Figure 4. Note that the tetrahedra τ_1 and τ_2 output by `BisectTet` will be of the same type. Figure 5 summarizes the relation between the input tetrahedron type and the output tetrahedra type. Note that types M and O are never output.

We close this section by relating the algorithm just described to those of Bänsch [3] and Liu and Joe [5], [6]. The relationship to the work of Maubach [7] will be treated in section 4. Bänsch used a slightly different system for marking a tetrahedron. Namely he associated to each face of the tetrahedron a unique marked edge. He assumes that at least one edge is marked in both faces containing it, and he calls such an edge a *global refinement edge*. This allows the possibility that a tetrahedron has two global refinement edges. Bänsch is not explicit in how such a tetrahedron is bisected. Presumably an arbitrary selection of one of the global refinement edges is made. Assuming such a selection, there is a direct correspondence between Bänsch's marking

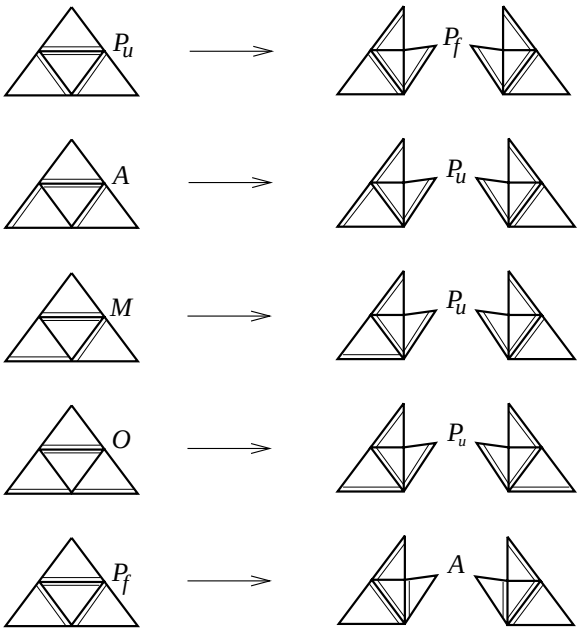


FIG. 4. *Bisection rules for marked tetrahedra.*

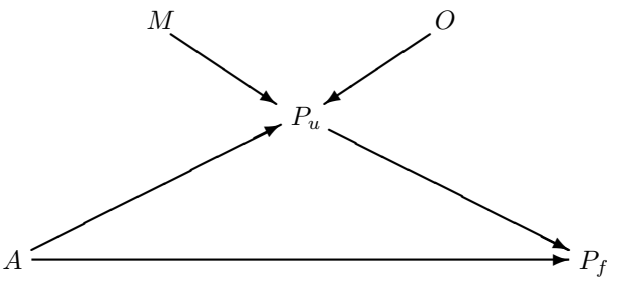


FIG. 5. *The types of tetrahedra produced by* `BisectTet`.

and ours. Bänsch refers to a tetrahedron of type A as red, one of type P as black, and does not explicitly name tetrahedra of type O or M. Our marking also involves a flag which for planar tetrahedra may be either set or unset. Bänsch addresses the same situation by referring to the parent of the tetrahedron: he singles out black tetrahedra with black parents for special treatment. These are precisely our tetrahedra of type P_f. For each type of tetrahedron Bänsch furnishes a picture showing how it should be refined. The algorithm thus described is, via the correspondences just presented, precisely equivalent to `BisectTet`. In this sense, `BisectTet` may be thought of as a convenient reformulation of Bänsch's algorithm.

The relationship to the work of Liu and Joe is less direct. In [5] they present an algorithm for the repeated bisection of an arbitrary tetrahedron. For this they introduce a special tetrahedron with a specific ordering of its vertices. The special tetrahedron and its descendants are always bisected using the longest edge as the refinement edge. They show that, for this particular tetrahedron, the mesh of nth

generation descendants is conforming and contains only one similarity class, which, in fact, depends only on n mod 3. To bisect an arbitrary tetrahedron, they choose an ordering for its vertices and use this to define a unique affine isomorphism with the special tetrahedron. The descendants of the arbitrary tetrahedron are then given as the inverse images of the descendants of the special tetrahedron under this transformation. Liu and Joe prove that the descendants fall into at most 168 distinct similarity classes and give a lower bound for a measure of their shape.

To draw the connection with `BisectTet`, we mark Liu and Joe's special tetrahedron with vertices p_0, p_1, p_2, and p_3, defined in their paper, so that $\overline{p_1p_2}$ is the refinement edge and $\overline{p_1p_3}$ and $\overline{p_2p_3}$ are the other marked edges. This is a planar marking, and we leave the flag unset, so that the special tetrahedron is marked to be type P_u. It can then be verified that applications of `BisectTet` to this marked tetrahedron and its descendants always bisect the longest edge. Indeed, for the first three generations this can be computed explicitly, and it turns out that at the third generation all of the descendants are similar to original special tetrahedron and correspondingly marked, so the bisection of the longest edge continues. Thus, for the special tetrahedron with appropriate marking, Liu and Joe's algorithm coincides with `BisectTet`. For an arbitrary tetrahedron with a selected ordering of its vertices we may use the affine isomorphism of the previous paragraph to transfer the marking from the special tetrahedron, and then we again have equivalence between the algorithm of Liu and Joe and the algorithm `BisectTet`. To summarize, given an arbitrary tetrahedron and an ordering of its vertices, we can determine a marking of this tetrahedron of type P_u, so that our algorithm and Liu and Joe's algorithm for repeated bisection of the tetrahedron are equivalent. In this sense the algorithm of [5] is equivalent to the special case of `BisectTet` applied to a tetrahedron of type P_u and its descendants. As a consequence we may apply our results from section 4 below to deduce that the descendants in fact fall into at most 36 similarity classes.

As remarked by the authors themselves, the question of how the algorithm of [5] can be applied to create a locally refined conforming mesh is far from obvious. This is the subject of [6]. The final algorithm, `QLRB`, which incorporates other algorithms called `BISECT1`, `BISECT2`, `TRANBIS`, `NEWTYPE`, `REFINE`, and `PREBISECT`, is far too complicated to describe here. We are not able to ascertain the precise relationship with `BisectTet` and the algorithm `LocalRefine` presented in the next section, although there are clearly points of similarity. In particular, the tetrahedron classes DD, DSS1, DSS2, and DSS3 introduced in [6] directly correspond to our types O, P, A, and M, respectively. In our analysis of `LocalRefine` in the next section we prove a termination theorem, Theorem 3.1. This is in fact the exact analogue of Theorem 3.1 for `QLRB` of [6].

3. A locally adaptive mesh refinement procedure. In many applications, such as adaptive finite element computations, one wishes to construct a sequence of nested conforming meshes which are adapted to a given criterion. A key step is the construction of a conforming refinement of a given conforming mesh, in which a selected subset of elements has been refined. In this section we describe an algorithm based on `BisectTet` to accomplish this.

Before stating the algorithm we fix some terminology. A *mesh* $\mathcal{T}$ of a domain Ω in $\mathbb{R}^3$ is a set of closed tetrahedra with disjoint interiors and union $\bar{\Omega}$. The *vertex set* of $\mathcal{T}$ is $\mathcal{V}(\mathcal{T}) = \bigcup\{\mathcal{V}(\tau) : \tau \in \mathcal{T}\}$. The *edge set* $\mathcal{E}(\mathcal{T})$ and the *face set* $\mathcal{F}(\mathcal{T})$ are defined similarly. A mesh is *conforming* if the intersection of two distinct tetrahedra is either a common face, a common edge, a common vertex, or empty. If $\tau \in \mathcal{T}$ and

$\nu \in \mathcal{V}(\mathcal{T})$, we say that ν is a *hanging node* of τ if $\nu \in \tau \setminus \mathcal{V}(\tau)$. A mesh is conforming if no tetrahedron in it has a hanging node and every face of every tetrahedron in the mesh either belongs to the boundary or is a face of another tetrahedron in the mesh. A mesh is *marked* if each tetrahedron in it is marked. A marked conforming mesh is *conformingly-marked* if each face has a unique marked edge (that is, when a face is shared by two tetrahedra, the marked edge is the same for both). Given an arbitrary conforming mesh the following marking procedure yields a conformingly-marked mesh. Strictly order the edges of the mesh in an arbitrary but fixed manner, for example, by length with a well-defined tie-breaking rule. Then choose the maximal edge of each tetrahedron as its refinement edge and the maximal edge of each face as its marked edge. Unset the flag on all tetrahedra. (The assumption that the coarse mesh has no flagged tetrahedra will be used in the analysis below.)

We now state the main algorithm of this section.

ALGORITHM $\mathcal{T}' = \texttt{LocalRefine}\ (\mathcal{T}, \mathcal{S})$

input: conformingly-marked mesh $\mathcal{T}$ and $\mathcal{S} \subset \mathcal{T}$

output: conformingly-marked mesh $\mathcal{T}'$

1. $\bar{\mathcal{T}} = \texttt{BisectTets}(\mathcal{T}, \mathcal{S})$
2. $\mathcal{T}' = \texttt{RefineToConformity}(\bar{\mathcal{T}})$

The algorithm in the first step, `BisectTets`, is trivial; we simply bisect each tetrahedron in $\mathcal{S}$:

$$\texttt{BisectTets}(\mathcal{T}, \mathcal{S}) = (\mathcal{T} \setminus \mathcal{S}) \cup \bigcup_{\tau \in \mathcal{S}} \texttt{BisectTet}(\tau).$$

In the second step, we perform further refinement as necessary to obtain a conforming mesh:

ALGORITHM $\mathcal{T}' = \texttt{RefineToConformity}(\mathcal{T})$

input: marked mesh $\mathcal{T}$

output: marked mesh $\mathcal{T}'$ without hanging nodes

1. set $\mathcal{S} = \{\tau \in \mathcal{T} \mid \tau \text{ has a hanging node}\}$
2. if $\mathcal{S} \neq \emptyset$ then
   $\bar{\mathcal{T}} = \texttt{BisectTets}(\mathcal{T}, \mathcal{S})$
   $\mathcal{T}' = \texttt{RefineToConformity}(\bar{\mathcal{T}})$
3. else
   $\mathcal{T}' = \mathcal{T}$

The recursion in the Algorithm `RefineToConformity` could conceivably continue forever. Moreover, even if the recursion terminates, the output mesh may not be conforming (a mesh without hanging nodes can nonetheless be nonconforming; cf. Figure 6). However, we shall prove that the recursion does terminate in the application of `RefineToConformity` in Algorithm `LocalRefine`, and that the resulting output mesh is conformingly-marked. Moreover, we shall bound the amount of refinement which can occur before termination. To state this result precisely, we consider an initial marked mesh $\mathcal{T}_0$ and set $\mathcal{Q}_0 = \mathcal{T}_0$, and $\mathcal{Q}_{k+1} = \texttt{BisectTets}(\mathcal{Q}_k, \mathcal{Q}_k)$, $k = 0, 1, \ldots$. Thus $\mathcal{Q}_1$ consists of all children of tetrahedra in the initial mesh, $\mathcal{Q}_2$ all grandchildren, etc. We assign *generation* k to all tetrahedra in $\mathcal{Q}_k$.

THEOREM 3.1. *Let $\mathcal{T}_0$ be a conformingly-marked mesh with no flagged tetrahedra. For $k = 0, 1, \ldots$, choose $\mathcal{S}_k \subset \mathcal{T}_k$ arbitrarily, and set $\mathcal{T}_{k+1} = \texttt{LocalRefine}(\mathcal{T}_k, \mathcal{S}_k)$. Then for each k, the application of* `RefineToConformity` *from within* `LocalRefine` *terminates producing a conformingly-marked mesh, and each tetrahedron in $\mathcal{T}_k$ has a generation of at most $3k$. Moreover, if the maximum generation of a tetrahedron in*

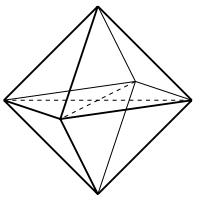


FIG. 6. *A nonconforming mesh without hanging nodes (the barycenter is* not *a vertex of the mesh).*

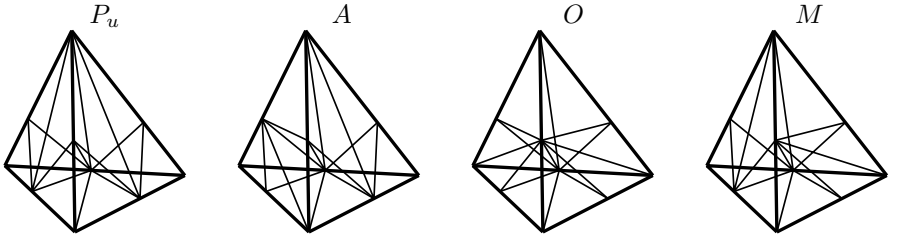


FIG. 7. *The meshes obtained via three applications of* `BisectTet` *beginning with a tetrahedra of type* P_u, A, O, *and* M.

$\mathcal{T}_k$ *is less than* $3m$ *for some integer* m, *then the maximum generation of a tetrahedron in* $\mathcal{T}_{k+1}$ *is less than or equal to* $3m$.

For the proof of the theorem, we need a classification of the edges that arise from repeated bisection of an unflagged marked tetrahedron τ. Let $\mathcal{Q}_0^\tau = \{\tau\}$ and define the meshes $\mathcal{Q}_k^\tau$ in analogy to the definition of the $\mathcal{Q}_k$ above (so $\mathcal{Q}_k^\tau$ contains all descendants of τ of generation k). We define $\mathcal{E}_k(\tau) = \mathcal{E}(\mathcal{Q}_{3k}^\tau)$ and refer to these as the edges of generation k. Thus the edges of generation 0 are exactly the edges of τ itself, and, referring to Figure 7, we verify that the edges of generation 1 are precisely the following 25 line segments:

- the line segment connecting the midpoint of the refinement edge to the midpoint of the opposite edge;
- for each face, the line segment connecting the midpoint of the marked edge to the opposite vertex;
- for each face, the two line segments connecting the midpoint of the marked edge to the midpoints of the two nonmarked edges;
- for each edge, its two children.

LEMMA 3.2. *Let* τ *be an unflagged marked tetrahedron. Then for* $k = 1, 2, \ldots$, *the mesh* $\mathcal{Q}_{3k}^\tau$, *consisting of all descendants of* τ *of generation* $3k$, *is conformingly-marked. If* τ *is of type* P_u, *then all the tetrahedra in* $\mathcal{Q}_{3k}^\tau$ *are of type* P_u, *and otherwise all the tetrahedra in* $\mathcal{Q}_{3k}^\tau$ *are of type* A.

Proof. It is clear from Figure 7 that $\mathcal{Q}_3^\tau$ is conforming. Moreover, the definition of `BisectTet` insures $\mathcal{Q}_3^\tau$ is conformingly-marked (because whenever a face is introduced, it is marked identically in the tetrahedra containing it). From the diagram in Figure 5, we see that tetrahedra in $\mathcal{Q}_3^\tau$ are all either type P_u or type A, depending on whether τ is type P_u. This verifies the lemma in case $k = 1$.

If $\tau' \in \mathcal{Q}_3^\tau$, then the mesh of third generation descendants of τ' is, by the same argument, conformingly-marked and uniformly of type P_u or A. Because $\mathcal{Q}_3^\tau$ is it-

TABLE 1
The types of tetrahedra and generation of edges occurring in the meshes $\mathcal{Q}_k$.

Generations of tetrahedra	Types of tetrahedra		Generations of edges
0	P_u	nonplanar	0
1	P_f	P_u	0,1
2	A	P_f	0,1
3	P_u	A	1
4	P_f	P_u	1,2
5	A	P_f	1,2
6	P_u	A	2
⋮	⋮	⋮	⋮
$3k$	P_u	A	k
$3k+1$	P_f	P_u	$k, k+1$
$3k+2$	A	P_f	$k, k+1$
$3(k+1)$	P_u	A	$k+1$

self conformingly-marked, the mesh obtained by combining all these meshes is again conformingly-marked, verifying the lemma in case $k = 2$. By induction we obtain the result for all k. □

If τ' is a generation $3k$ descendant of an unflagged marked tetrahedron τ, then, by definition, all of the edges of τ' have generation k. The next lemma determines the generations of the edges of descendants of generation $3k+1$ and $3k+2$.

LEMMA 3.3. *Let τ be an unflagged marked tetrahedron and τ' a descendant of τ of generation $3k+1$ or $3k+2$. Then the edges of τ' all have generation k or $k+1$.*

Proof. The tetrahedron τ' is either a child or a grandchild of an unflagged tetrahedron of generation $3k$. From Figure 7, it is easy to see that every edge of a child or a grandchild of an unflagged tetrahedron is either an edge of that tetrahedron or an edge of one of its great grandchildren. Thus each edge of τ' is an edge of a tetrahedron whose generation is either $3k$ or $3k+3$. □

Returning to Theorem 3.1, we easily deduce the following proposition from the preceding lemmas.

PROPOSITION 3.4. *Let $\mathcal{T}_0$ be a conformingly-marked mesh with no flagged tetrahedra, and let $\mathcal{Q}_k$ denote the mesh of all descendants of generation k of tetrahedra in $\mathcal{T}_0$. Then the types of tetrahedra and the generation of edges of occurring in $\mathcal{Q}_k$ are as shown in Table* 1. *Moreover, the meshes $\mathcal{Q}_{3k}$ are conformingly-marked.*

Proof of Theorem 3.1. The proof will proceed in several steps. We use the terminology *descendant mesh of $\mathcal{T}_0$* to refer to any mesh that can arise from $\mathcal{T}_0$ by repeated application of `BisectTet`.

Step 1. If $\mathcal{T}$ is any descendant mesh of $\mathcal{T}_0$ which has maximal tetrahedron generation $3k$, then the application of `BisectTets` in step 2 of `RefineToConformity`($\mathcal{T}$) returns a mesh which again has maximal tetrahedron generation $3k$.

Proof. We refer to Table 1 to see that all the edges of tetrahedra in $\mathcal{T}$ are of most generation k. Now if a tetrahedron in $\mathcal{T}$ has a hanging node, then the edge of the tetrahedron on which the hanging node lies must have generation $k-1$ or less (since its children are also edges in the mesh and so have generation k or less). Hence, again with reference to the table, a tetrahedron with a hanging node has generation strictly less than $3k$. That is, the set $\mathcal{S}$ defined in step 1 of `RefineToConformity` consists of tetrahedra of generation less than $3k$. Consequently the mesh output from `BisectTets` in step 2 of `RefineToConformity`, again has maximal tetrahedron generation $3k$.

Step 2. If $\mathcal{T}$ is any descendant mesh of $\mathcal{T}_0$ which has maximal tetrahedron gener-

ation $3k$, then every mesh constructed in the recursive application of the algorithm `RefineToConformity`$(\mathcal{T})$ has maximal tetrahedron generation $3k$ and, moreover, the algorithm terminates.

Proof. Indeed new tetrahedra are only introduced by the application of `BisectTets` in step 2 of `RefineToConformity`, so the generation bound follows from the previous step. Since there are only finitely many tetrahedra of generation $3k$, the algorithm must terminate.

Step 3. Each tetrahedron in $\mathcal{T}_k$ has generation at most $3k$.

Proof. The proof is by induction on k, with the case $k = 0$ being the obvious final step. By the induction hypothesis, $\mathcal{T}_{k-1}$ consists of tetrahedra of generation at most $3k - 3$. Hence, the mesh $\bar{\mathcal{T}}$ output from `BisectTets` in step 1 of `LocalRefine`$(\mathcal{T}_{k-1}, \mathcal{S}_{k-1})$ has maximum tetrahedron generation $3k - 2 \le 3k$, and the result follows from the preceding claim.

Step 4. The output mesh is conformingly-marked.

Proof. This follows easily from the fact that the output mesh is a descendant of a conformingly-marked mesh and has no hanging nodes.

Step 5. The last sentence of the theorem follows directly from step 2. ∎

4. Similarity classes. In [7] Maubach presented an algorithm, which we refer to as `BisectSimplex`, for the bisection of an arbitrary n-simplex in $\mathbb{R}^n$. After recalling this algorithm, we shall show that in the special case $n = 3$, it is essentially equivalent to `BisectTet`, when `BisectTet` is restricted to tetrahedra of types A and P as input. Define a *tagged simplex* in $\mathbb{R}^n$ as an ordered pair, $\mathfrak{t} = \big((x_0, x_1, \dots, x_n), d\big)$, where $(x_0, x_1, \dots, x_n)$ is an ordered $(n+1)$-tuple of points in general position in $\mathbb{R}^n$ (the vertices of the simplex), and $d \in \{1, 2, \dots, n\}$ is a *tag* ($\overline{x_0 x_d}$ is the refinement edge of $\mathfrak{t}$). With this terminology, Maubach's algorithm may be stated as follows.

ALGORITHM $\{\mathfrak{t}^{(1)}, \mathfrak{t}^{(2)}\} =$ `BisectSimplex`$(\mathfrak{t})$

input: tagged n-simplex $\mathfrak{t}$

output: tagged n-simplices $\mathfrak{t}^{(1)}$ and $\mathfrak{t}^{(2)}$.

1. set $d' = \begin{cases} d-1, & d > 1 \\ n, & d = 1 \end{cases}$
2. create the new vertex $z = \frac{1}{2}(x_0 + x_d)$
3. set $\mathfrak{t}^{(1)} = \big((x_0, x_1, \dots, x_{d-1}, z, x_{d+1}, ..., x_n), d'\big)$
4. set $\mathfrak{t}^{(2)} = \big((x_1, x_2, \dots, x_d, z, x_{d+1}, ..., x_n), d'\big)$

We now define a correspondence between tagged simplices in $\mathbb{R}^3$ and marked tetrahedra of types P and A, and we show that repeated application of either `BisectTet` or `BisectSimplex` to the same geometric tetrahedron yields the same sequence of descendant tetrahedra. Let $\mathcal{T}_T$ be the set of all tagged 3-simplices and $\mathcal{T}_M$ be the set of all marked tetrahedra; also, let $\mathcal{T}_a \subset \mathcal{T}_M$ denote the set of marked tetrahedra of type A or P. Define a mapping $\mathcal{F} : \mathcal{T}_T \to \mathcal{T}_M$ as follows.

For $\mathfrak{t} = \big((x_0, x_1, x_2, x_3), d\big) \in \mathcal{T}_T$, set $\mathcal{F}(\mathfrak{t}) = (\mathcal{V}(\tau), r_\tau, m_\varphi, f_\tau)$ with $\mathcal{V}(\tau) = \{x_0, x_1, x_2, x_3\}$ and

$$r_\tau = x_0 x_1, \quad m_\varphi = \begin{cases} \overline{x_0 x_3} & \text{if } \varphi = x_0 x_2 x_3, \\ \overline{x_1 x_3} & \text{if } \varphi = x_1 x_2 x_3, \end{cases} \quad f_\tau = 1 \quad \text{if } d = 1;$$

$$r_\tau = x_0x_2, \quad m_\varphi = \begin{cases} \overline{x_0x_1} & \text{if } \varphi = x_0x_1x_3, \\ \overline{x_1x_2} & \text{if } \varphi = x_1x_2x_3, \end{cases} \quad f_\tau = 0 \quad \text{if } d = 2;$$

$$r_\tau = x_0x_3, \quad m_\varphi = \begin{cases} \overline{x_0x_2} & \text{if } \varphi = x_0x_1x_2, \\ \overline{x_1x_3} & \text{if } \varphi = x_1x_2x_3, \end{cases} \quad f_\tau = 0 \quad \text{if } d = 3.$$

Note that $\mathcal{F}(\mathfrak{t})$ has type P_f, P_u, or A, when $d = 1$, 2, or 3, respectively. Consequently, $\mathcal{F}(\mathcal{T}_T) \subset \mathcal{T}_a$. In fact, we have the following proposition.

PROPOSITION 4.1. *The mapping $\mathcal{F}$ maps $\mathcal{T}_T$ onto $\mathcal{T}_a$ and is precisely 2 to 1.*

Proof. Given a marked tetrahedron $\tau = (\{v_0, v_1, v_2, v_3\}, r_\tau, m_\varphi, f_\tau) \in \mathcal{T}_a$, we define two tagged 3-simplices in its preimage under $\mathcal{F}$ as follows.

If τ has type A, we assume, without loss of generality, that its refinement and nonrefinement edges are $\overline{v_0v_1}$ and $\{\overline{v_0v_2}, \overline{v_1v_3}\}$, respectively. To get the first tagged simplex, set $d = 3$ and choose $x_0 = v_0$ and $x_d = v_1$ (the end points of the refinement edge). Set $x_1 = v_3$ (the second endpoint of the marked nonrefinement edge containing x_d) and $x_2 = v_2$. To obtain the second tagged simplex, we reverse x_0 and x_3 and x_1 and x_2. Thus, the two tagged 3-simplices corresponding to the given marked tetrahedron are $\big((v_0, v_3, v_2, v_1), 3\big)$ and $\big((v_1, v_2, v_3, v_0), 3\big)$. It is straightforward to check that $\mathcal{F}$ maps each of these tagged simplices to τ and that these are the only preimages of τ under $\mathcal{F}$. The argument is similar in the cases of τ of type P_u or P_f. In the P_u case, we take the numbering so that the refinement and marked nonrefinement edges are $\overline{v_0v_1}$ and $\{\overline{v_0v_2}, \overline{v_1v_2}\}$, respectively, and find the two preimages $\big((v_0, v_2, v_1, v_3), 2\big)$ and $\big((v_1, v_2, v_0, v_3), 2\big)$. In the P_f case with refinement and marked nonrefinement edges $\overline{v_0v_1}$ and $\{\overline{v_0v_2}, \overline{v_1v_2}\}$, the preimages are $\big((v_0, v_1, v_3, v_2), 1\big)$ and $\big((v_1, v_0, v_3, v_2), 1\big)$. ∎

The following theorem exhibits the relation between the algorithms `BisectTet` and `BisectSimplex`.

THEOREM 4.2. *The following diagram commutes.*

$$\begin{array}{ccc} \mathcal{T}_T & \xrightarrow{\ \mathcal{F}\ } & \mathcal{T}_M \\ \Big\downarrow{\scriptstyle \texttt{BisectSimplex}} & & \Big\downarrow{\scriptstyle \texttt{BisectTet}} \\ \mathcal{T}_T \times \mathcal{T}_T & \xrightarrow{\ \mathcal{F}\times\mathcal{F}\ } & \mathcal{T}_M \times \mathcal{T}_M \end{array}$$

Proof. Suppose $\mathfrak{t} = \big((x_0, x_1, x_2, x_3), 3\big) \in \mathcal{T}_T$, and $\{\mathfrak{t}^{(1)}, \mathfrak{t}^{(2)}\} \in \mathcal{T}_T \times \mathcal{T}_T$ is produced by `BisectSimplex`($\mathfrak{t}$). Then $\mathfrak{t}^{(1)} = \big((x_0, x_1, x_2, z), 2\big)$ with $z = (x_0 + x_3)/2$, and so $\mathcal{F}(\mathfrak{t}^{(1)})$ yields the marked tetrahedron $(\{x_0, x_1, x_2, z\}, \overline{x_0x_2}, \{\overline{x_0x_1}, \overline{x_1x_2}\}, 0)$. (Here, only the markings for the nonrefinement faces of the tetrahedron are specified in m_φ with the convention that the given edges are marked for the nonrefinement face containing them.) On the other hand, $\mathcal{F}(\mathfrak{t}) = (\{x_0, x_1, x_2, x_3\}, \overline{x_0x_3}, \{\overline{x_0x_2}, \overline{x_1x_3}\}, 0)$ which is a tetrahedron of type A, and one of the marked tetrahedra produced by applying `BisectTet` to this tetrahedron is $\mathcal{F}(\mathfrak{t}^{(1)})$. A similar verification is easily carried out for the other cases. ∎

Theorem 4.2 implies that if `BisectSimplex` is applied m times to a tagged 3-simplex, the images under the mapping $\mathcal{F}$ of the 2^m descendents are exactly the descendents obtained by applying `BisectTet` m times to the image under $\mathcal{F}$ of the parent. The following corollary is thus immediate.

COROLLARY 4.3. *The 2^m geometric 3-simplices obtained by applying the algorithm* `BisectSimplex` *m times to $\mathfrak{t} \in \mathcal{T}_T$ are exactly the same as those obtained by applying* `BisectTet` *m times to $\mathcal{F}(\mathfrak{t})$.*

Our next goal is a bound on the number of similarity classes produced by repeated application of algorithm `BisectSimplex`. (Two simplices are said to belong to the same similarity class if one can be transformed to the other via a dilation and a rigid motion.) This will be based on the following technical result from [7]. (Theorem 4.1 of [7] states this result in less generality, but the same proof applies to the statement given here.)

LEMMA 4.4. *Let $\mathfrak{t}'$ be a descendent of the tagged n-simplex*

$$\mathfrak{t} = \big((0, e_1, e_1 + e_2, \dots, e_1 + e_2 + \cdots + e_n), n\big)$$

obtained via k applications of `BisectSimplex`*, where $\mathcal{B} = \{e_1, e_2, \dots, e_n\}$ is an arbitrary basis of $\mathbb{R}^n$. Define integers $q \geq 0$ and $r \in \{0, \dots, n-1\}$ by $k = qn + r$. Let $(x_0, x_1, \dots, x_n)$ be the ordered vertices of $\mathfrak{t}'$ and define $y_i = x_i - x_0$ for $i = 1, 2, \dots, n$. Then, there exist two linear mappings π and R from $\mathbb{R}^n$ to $\mathbb{R}^n$ such that $\{\pi(e_1), \pi(e_2), \dots, \pi(e_n)\}$ is a permutation of the basis vectors of $\mathcal{B}$, and $R(e_i) = \pm e_i$, $1 \leq i \leq n$, with*

$$y_i = \left(\frac{1}{2}\right)^q \alpha_i R\pi \sum_{j=1}^{i} e_j, \qquad 1 \leq i \leq n,$$

where

$$\alpha_i = \begin{cases} 1, & 1 \leq i \leq n - r, \\ \frac{1}{2}, & n - r + 1 \leq i \leq n. \end{cases}$$

THEOREM 4.5. *The number of similarity classes of n-simplices produced by the repeated application of* `BisectSimplex` *is bounded by $nn!2^{n-2}$.*

Proof. Without loss of generality we may assume that the initial simplex has tag n, because an arbitrary tagged simplex is a descendant of such a simplex. For example, $\big((x_0, x_1, x_2, \dots, x_n), n-1\big)$ is a child of $\big((x_0, x_1, x_2, \dots, 2x_n - x_0), n\big)$. Moreover we may translate so that the first vertex is at the origin, and thus the initial simplex can be taken to be of the form $\mathfrak{t}$ assumed in the lemma.

From this immediately get an upper bound of $nn!2^n$ similarity classes for the descendants, since there are $n!$ possibilities for the permutations π, 2^n possibilities for the reflections R, and exactly n different vectors α_i. Noting that two different reflections, R and $-R$, give n-simplices in the same similarity class, the bound can be reduced by a factor of 2 to $nn!2^{n-1}$. To prove the theorem it suffices to reduce this by another factor of 2, which we shall do by showing that each n-simplex produced is a translate of another.

Using the notations introduced in Lemma 4.4 and noting that q does not play a role in the count for the number of similarity classes of tetrahedra, we will assume $q = 0$ without any loss of generality. Define the mappings $\hat{\pi} : \mathbb{R}^n \to \mathbb{R}^n$ and $\hat{R} : \mathbb{R}^n \to \mathbb{R}^n$ by

$$\hat{\pi}(e_j) = \begin{cases} \pi(e_{n-r+1-j}), & 1 \leq j \leq n - r, \\ \pi(e_j), & n - r + 1 \leq j \leq n, \end{cases} \tag{1}$$

$$\hat{R}\pi\,(e_j) = \begin{cases} -R\pi(e_j), & 1 \leq j \leq n - r, \\ R\pi(e_j), & n - r + 1 \leq j \leq n, \end{cases} \tag{2}$$

for $j \in \{1, 2, \dots, n\}$. Note that $\hat{\pi}$ is a permutation and $\hat{R}$ a reflection relative to $\mathcal{B}$. The ordered set $(0, y_1, y_2, \dots, y_n)$ represents the vertices of the tagged n-simplex $\mathfrak{t}'$. Denote the ordered set of vertices of another tagged n-simplex $\hat{\mathfrak{t}}$ by $(0, \hat{y}_1, \hat{y}_2, \dots, \hat{y}_n)$ with $\hat{y}_i = \alpha_i \hat{R}\hat{\pi} \sum_{j=1}^{i} e_i$ for $1 \le i \le n$. Let $\Phi : \mathbb{R}^n \to \mathbb{R}^n$ be the translation defined by

$$\Phi(x) = x - R\pi \sum_{j=1}^{n-r} e_j = x - y_{n-r}.$$

Then $\Phi(0) = -R\pi \sum_{j=1}^{n-r} e_j$ and

$$\Phi(y_i) = \begin{cases} -R\pi \sum\limits_{j=i+1}^{n-r} e_j, & 1 \le i < n-r, \\ 0, & i = n-r, \\ -\frac{1}{2} R\pi \sum\limits_{j=1}^{n-r} e_j + \frac{1}{2} R\pi \sum\limits_{j=n-r+1}^{i} e_j, & n-r+1 \le i \le n. \end{cases}$$

Using (1), (2), and the definition of $\hat{y}_i$, we get $\Phi(0) = \hat{y}_{n-r}$ and

$$\Phi(y_i) = \begin{cases} \hat{y}_{n-r-i}, & 1 \le i < n-r, \\ 0, & i = n-r, \\ \hat{y}_i, & n-r+1 \le i \le n. \end{cases}$$

Thus, $\mathfrak{t}'$ is related to $\hat{\mathfrak{t}}$ via the translation Φ as desired. □

For $n = 2$ the bound of four similarity classes furnished by the theorem is easily seen to be sharp. For $n = 3$, the bound is 36. By direct computation on a particular tetrahedron we have verified that the bound of 36 is sharp. (For example, if $\mathfrak{t} = \big((v_0, v_1, v_2, v_3), 3\big)$, where $v_0 = (0, 0, 0)$, $v_1 = (23, 0, 0)$, $v_2 = (7, 0, 11)$, and $v_3 = (17, 5, 33)$, then the upper bound of 36 is attained at the sixth generation.) Maubach [8] has announced a proof that the bound of $nn!2^{n-2}$ is sharp for all n.

In view of Proposition 4.1 and Theorem 4.2, the above result applies to bisection of marked tetrahedra of types A and P: repeated application of `BisectTet` starting from such a marked tetrahedron will produce at most 36 similarity classes. Since one application of `BisectTet` to a tetrahedron of type O or M produces children of type P_u, the repeated bisection of such a tetrahedron will produce at most 72 similarity classes of tetrahedra. In particular, for an arbitrary initial mesh of marked tetrahedra, only finitely many similarity classes will arise in its descendant meshes.

5. Examples. In this section, we give some examples of adapted tetrahedral meshes generated using `LocalRefine`. A coarse initial mesh $\mathcal{T}_0$ is chosen and the meshes $\mathcal{T}_k$ are generated using $\mathcal{T}_k = \texttt{LocalRefine}(\mathcal{T}_{k-1}, \mathcal{S}_{k-1})$ as in section 3 with different criteria being used to determine the sets $\mathcal{S}_k$ in each example.

Example 1. The first example is adapted from Maubach [7]. Let $\mathcal{T}_0$ be the subdivision of the cube $[0, 1]^3$ into six congruent tetrahedra. We choose the longest edge of each face as its marked edge. It can easily be verified that all six tetrahedra are of type A and they belong to the same similarity class. Let

$$\mathcal{H} = \left\{ (x, y, z) \in \mathbb{R}^3 \middle| \left(x - \frac{1}{2}\right)^2 + \left(y - \frac{1}{2}\right)^2 + \left(z - \frac{1}{2}\right)^2 = \frac{1}{16} \text{ with } x \ge \frac{1}{2} \right\}$$

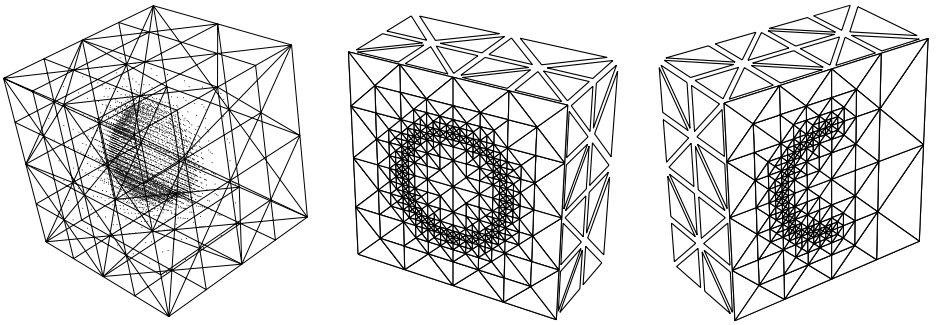


FIG. 8. *The mesh $\mathcal{T}_{16}$ of Example* 1. *The first view shows the vertices of the mesh, the second a cut along the plane $x = 1/2$, and the third a cut along the plane $y = 1/2$.*

be a hemisphere embedded in the cube. We choose $\mathcal{S}_{k-1} = \{\tau \in \mathcal{T}_{k-1} | \tau \cap \mathcal{H} \neq \emptyset\}$, so that we attempt to adapt the meshes to the hemisphere $\mathcal{H}$. Figure 8 shows different views of $\mathcal{T}_{16}$ having 25,448 tetrahedra. The local adaptivity around $\mathcal{H}$ is clear.

The algorithm `LocalRefine` has been incorporated into a code for solving second order elliptic boundary value problems (cf. [9]). Error estimators are used to determine the sets $\mathcal{S}_k$ in the code, leading to refinement in regions where the gradient of the solution changes rapidly.

Example 2. This example is a test problem for which the exact solution is known to be

$$u_{\text{ex}} = (x^2 - x)(y^2 - y)(z^2 - z)e^{-100[(x-1/4)^2+(y-1/4)^2+(z-1/4)^2]},$$

and $\mathcal{T}_0$ was taken to be a subdivision of $[0,1]^3$ having 96 tetrahedra. The problem was solved using piecewise linear finite elements, and a full multigrid solver based on the sequence of adaptively defined meshes. Note that u_{ex} varies very rapidly near the point $(1/4, 1/4, 1/4)$, and has relatively slow variation in other parts of the domain. This behavior is captured well by the adaptive mesh refinement process, as seen in $\mathcal{T}_6$ pictured in Figure 9.

Figure 10 shows a log-log plot of the errors in energy and L^2 norms against $\mathcal{V}(\mathcal{T})^{-1/3}$, which is an indicator of mesh size for locally refined meshes, using both adaptive and uniform refinement. (Lines with slopes 1 and 2 are shown for easy comparison.) Using uniform refinement, the finest mesh has 68,705 vertices and a relative percentage energy error of approximately 15.85% while the maximum number of vertices using adaptive refinement are 62,738 and the corresponding relative percentage energy error is 4.95%.

Example 3. As a final example, we consider a problem that arose in numerical relativity concerning compatible initial metric data for the collision of two black holes. This involves a nonlinear elliptic boundary value problem which we solve with piecewise linear finite elements, an outer Newton iteration, and an inner multigrid iteration based on the sequence of adapted meshes. For details, see [9]. The domain consists of a ball about the origin of radius $64\sqrt{3}$ minus two disjoint balls of radius $\sqrt{3}/2$ and $\sqrt{3}$ centered at $(0, 0, -2\sqrt{3})$ and $(0, 0, 2\sqrt{3})$, respectively. Beginning with the mesh $\mathcal{T}_0$ containing 585 vertices and 2,982 tetrahedra the code produced an adapted mesh $\mathcal{T}_5$ with 63,133 vertices and 346,084 tetrahedra. In Figures 11 and 12, which is a zoom of the previous figure, we show results computed on the intermediate mesh $\mathcal{T}_3$ with

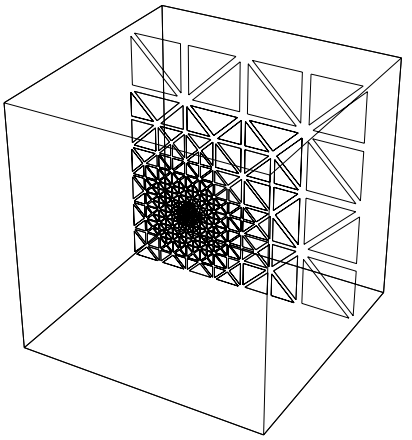


FIG. 9. *Cut along the the plane* $x = 1/4$ *of* $\mathcal{T}_6$ *of Example* 2 *(there are* 11,418 *tetrahedra and* 2,116 *vertices in the mesh).*

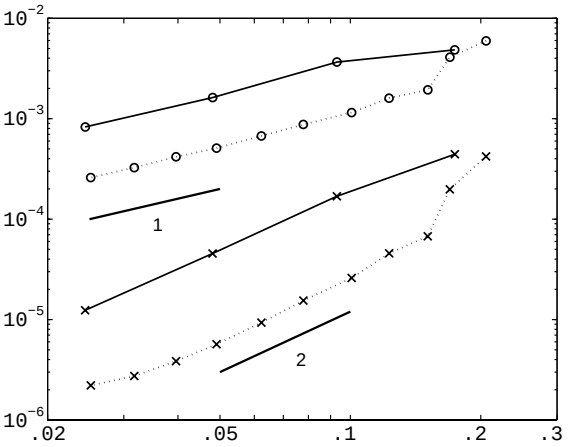


FIG. 10. *The energy* ($\circ$) *and* L^2 ($\times$) *errors and rates for example* 2. *The solid lines denote the energy (respectively,* L^2*) errors for uniform refinement while the dotted lines are for adaptive refinement.*

13,899 vertices and 75,300 tetrahedra.

The figures show a contour plot of the solution on the plane $x = y$ (the plot shade is keyed to the value of the solution). This is easier to interpret in the color version, which can be found in [9]. The intersections of the tetrahedra with the plane are shown slightly shrunken to improve visibility. Also shown are the mesh edges which intersect the boundary. Next, Figure 13 gives evidence of the mesh quality in the initial and final meshes. The shape measure used is the ratio of the circumscribed to inscribed spheres, scaled so that an equilateral tetrahedron has shape measure one. The first histogram shows that in the initial mesh more than 80% of elements have measure less than 2, while the worst elements have quality around 3.5. In the final mesh, 72% of the elements have measure less than 2, while 84% have measure less than 2.5, and the worst elements have measure about 5.4.

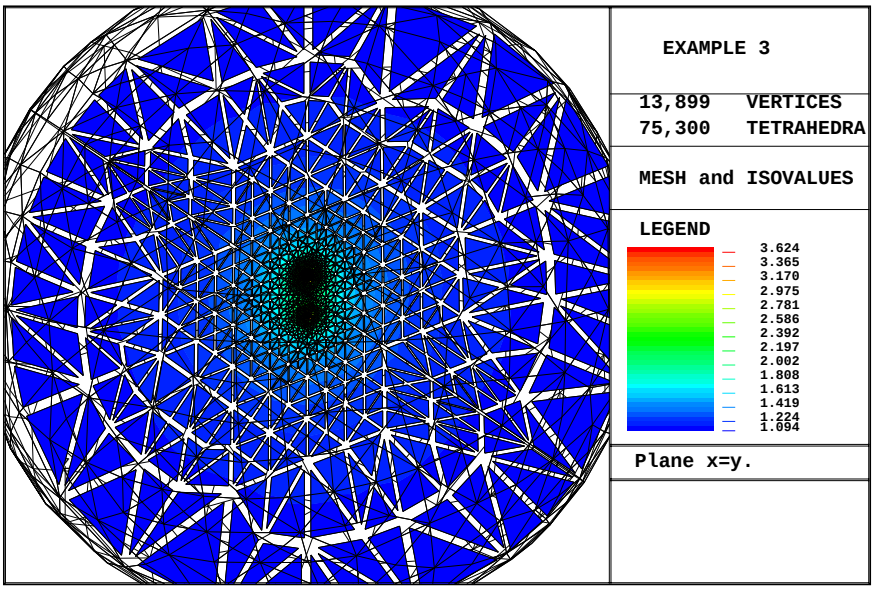


FIG. 11. *Solution to a binary black hole problem.*

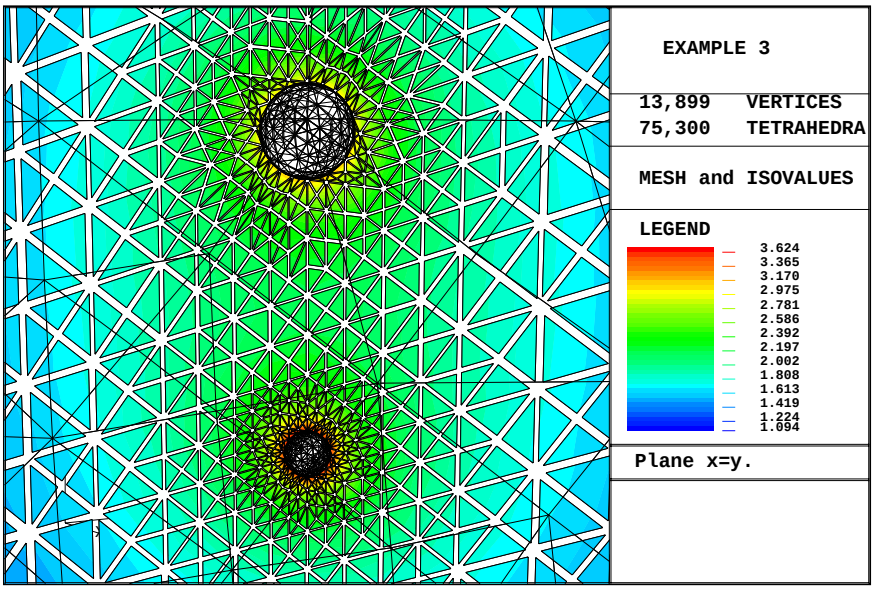


FIG. 12. *Solution to a binary black hole problem, zoom.*

Finally, Figure 14 shows a plot of the total CPU time for the solution of the boundary value problem including the mesh refinement, error estimation, outer iteration, and the multigrid solve on a 1993 DEC 3000 model 500 with a single 150 MHz Alpha processor versus number of mesh vertices. The plot clearly shows that the computation time is very nearly proportional to the number of vertices in the mesh.

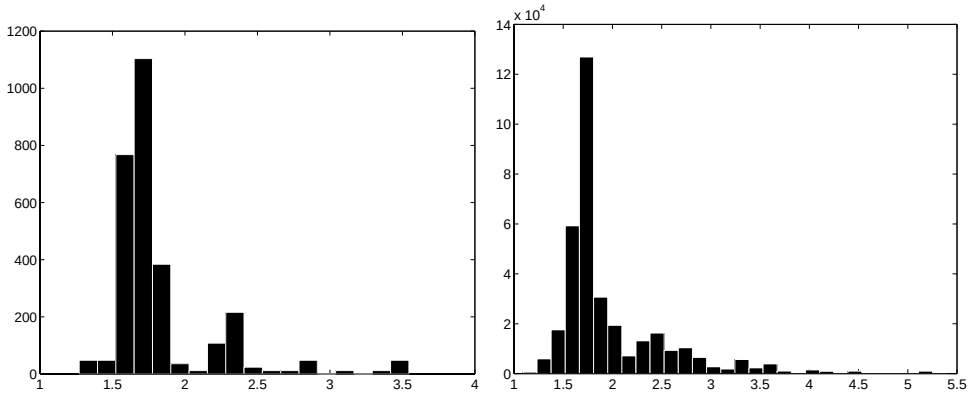


FIG. 13. *Shape measure histograms. Left: coarse mesh with* 2,982 *tetrahedra. Right: fine mesh with* 346,084 *tetrahedra.*

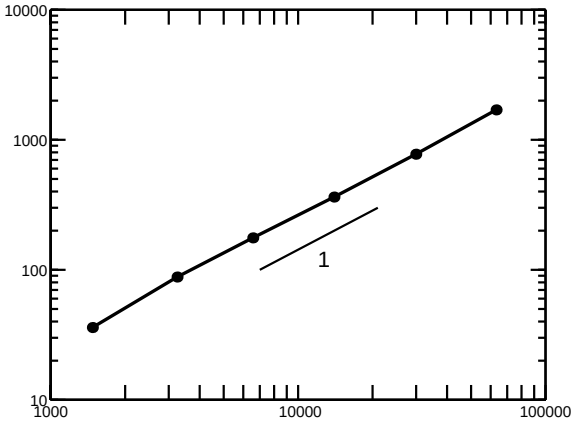


FIG. 14. *CPU seconds, on the y-axis, versus number vertices, on the x-axis, for a binary black hole problem.*

6. Conclusion. In section 2 we introduced the notion of a marked tetrahedron, and presented algorithm `BisectTet` for the bisection of a marked tetrahedron into two children. We discussed its relation which other algorithms in the literature. In section 3 we showed how, beginning with an arbitrary conforming tetrahedral mesh, the algorithm `BisectTet` can be applied to create locally refined conforming tetrahedral meshes, and we bounded the amount of additional refinement that might be required to obtain conformity. In the following section we showed that repeated application of `BisectTet` to a given tetrahedron and its descendants generates at most 36 or 72 dissimilar tetrahedra in all (depending on the marking of the original tetrahedron). In doing so, we showed that for some markings `BisectTet` is, in a certain specific sense, equivalent to the three-dimensional case of Maubach's algorithm for repeated bisection of n-simplices, and for this algorithm we showed that at most $n\,n!2^{n-2}$ dissimilar simplices are created. We closed the paper with numerical examples of the performance of the algorithm.

REFERENCES

[1] A. Adler *On the bisection method for triangles*, Math. Comp., 40 (1983), pp. 571–574.

[2] R. Bank, *PLTMG: A Software Package for Solving Elliptic Partial Differential Equations*, Users' Guide 7.0., SIAM, Philadelphia, 1994.

[3] E. Bänsch, *Local mesh refinement in* 2 *and* 3 *dimensions*, Impact Comput. Sci. Engrg., 3 (1991), pp. 181–191.

[4] Jürgen Bey, *Tetrahedral grid refinement*, Computing, 55 (1995), pp. 271–288.

[5] A. Liu and B. Joe, *On the shape of tetrahedra from bisection*, Math. Comp., 63 (1994), pp. 141–154.

[6] A. Liu and B. Joe, *Quality local refinement of tetrahedral meshes based on bisection*, SIAM J. Sci. Comput., 16 (1995), pp. 1269–1291.

[7] J. M. Maubach, *Local bisection refinement for N-simplicial grids generated by reflection*, SIAM J. Sci. Comput., 16 (1995), pp. 210–227.

[8] J. M. Maubach, *The Amount of Similarity Classes Created by Local n-Simplicial Bisection Refinement*, 1996, manuscript.

[9] A. Mukherjee, *An Adaptive Finite Element Code for Elliptic Boundary Value Problems in Three Dimensions with Applications in Numerical Relativity*, Ph.D. thesis, The Pennsylvania State University, University Park, PA, 1996.

[10] M. E. G. Ong, *Hierarchical Basis Preconditioners for Second Order Elliptic Problems in Three Dimensions*, Ph.D. thesis, University of Washington, Seattle, 1989.

[11] M. C. Rivara, *Local modification of meshes for adaptive and/or multigrid finite-element methods*, J. Comput. Appl. Math., 36 (1991), pp. 79–89.

[12] M. C. Rivara and C. Levin, *A* 3*-D refinement algorithm suitable for adaptive and multi-grid techniques*, Comm. Appl. Numer. Methods., 8 (1992), pp. 281–290.

[13] S. Zhang, *Multi-Level Iterative Techniques*, Ph.D. thesis, The Pennsylvania State University, University Park, PA, 1988.

SIAM J. SCI. COMPUT.
Vol. 22, No. 2, pp. 449–475
© 2000 Society for Industrial and Applied Mathematics

A FINITE-VOLUME METHOD FOR THE MAXWELL EQUATIONS IN THE TIME DOMAIN*

C.-D. MUNZ†, R. SCHNEIDER‡, AND U. VOß‡

Abstract. A finite-volume scheme for the Maxwell equations is proposed using collocated nonorthogonal curvilinear grid arrangements, with the advantage that all components of the electric and magnetic field are stored at the same computational location and time. It is based on construction principles of high-resolution schemes for hyperbolic conservation laws and takes into account the local wave propagation. The implementation of boundary conditions based on the characteristic theory is described. A new divergence correction technique is proposed to preserve locally the charge conservation. This is important if the Maxwell solver is used within an electromagnetic particle-in-cell (PIC) code. The accuracy and efficiency of this finite-volume framework is demonstrated with problems of electromagnetic wave propagation and of the self-consistent motion of charged particles.

Key words. Maxwell–Lorentz equations, particle-in-cell method, divergence correction techniques

AMS subject classifications. 78-08, 78A35, 65M99

PII. S1064827596307890

1. Introduction.

The traditional techniques in computational electromagnetics (CEM) for solving the Maxwell equations in the time domain rely on finite-differences and are called finite-difference time domain (FDTD) methods. One basic approach has been developed by Yee [37]. Here, a so-called staggered grid is used, where approximate values of the components of electromagnetic fields are calculated at different locations, e.g., at different faces and at the center of the grid zones. This staggering stabilizes the explicit central differences. Usually, these schemes are formulated staggered in time also, i.e., the electric and the magnetic fields are given at different time levels t_n and $t_{n+1/2}$. Originally, the staggered finite-difference scheme has been formulated by Yee for Cartesian coordinates and has been extended to structured boundary-fitted grids by Holland [15]. A review of FDTD schemes for the Maxwell equations and their applications has been given by Taflove [32]. Approximating an integral formulation of Maxwell equations based on the Stokes theorem, Weiland [36], and Madsen and Ziolkowski [18, 19] proposed staggered finite-volume schemes for general grids which reduce to the Yee algorithm in the Cartesian case. The main advantage of the staggered approach is that it combines second-order accuracy in space and time with a short stencil of the difference quotients. The disadvantages are that the field components are not given at the same points in space and time, resulting in a complicated data structure. Grid refinement techniques in regions of steep gradients are cumbersome.

In this paper, we consider the finite-volume approach on a collocated grid, where all field components as well as charge and current density are stored at the same computational location. The Maxwell equations are formulated as a system of con-

* Received by the editors August 5, 1996; accepted for publication (in revised form) April 24, 1997; published electronically July 13, 2000.
http://www.siam.org/journals/sisc/22-2/30789.html

†Universität Stuttgart, Institut für Aerodynamik und Gasdynamik, Pfaffenwaldring 21, D-70550 Stuttgart, Germany (claus-dieter.munz@inr.fzk.de). The research of this author was carried out in part at Forschungszentrum Karlsruhe.

‡Forschungszentrum Karlsruhe, Technik und Umwelt, Institut für Neutronenphysik und Reaktortechnik, Postfach 3640, D-76021 Karlsruhe, Germany (rudolf.schneider@ihm.fzk.de).

servation laws and are integrated component-by-component over each grid cell to achieve evolution equations for integral values. These values change in time due to fluxes through the grid zone interfaces. Mohammadian, Shankar, and Hall [22] approximated the conservation formulation by a classical second-order accurate method, the Lax–Wendroff method. The intention of this paper is to consider high-resolution schemes for hyperbolic conservation laws that take into account the direction of the local wave propagation. (For a review see [17], [25].) By that fact, they succeed to combine both robustness at strong gradients and high-order accuracy. They are dissipative but prevent the increase of numerical damping in time by the compressive nonlinear behavior (see, e.g., [23, 25]). Very recently, several authors proposed Maxwell solvers based on construction principles of the high-resolution schemes. Shang [26] and Shang and Gaitonde [27] used dimensional splitting and applied the high-resolution techniques in each one-dimensional sweep. This approach is restricted to second-order accuracy and to structured boundary-fitted grids. Cioni, Fezoui, and Steve [4] proposed a finite-volume/finite-element method. In both approaches the basic building block of the flux calculation is formulated as a generalization of the Steger–Warming flux splitting algorithm [30].

We propose in this paper a finite-volume Maxwell solver for curvilinear nonorthogonal coordinates without using dimensional splitting. The local wave propagation between adjacent grid zones is determined by the solution of Riemann problems. This has been proposed by Godunov [10] for the Euler equations and is usually called Godunov method. In the case of the Maxwell equations with constant permitivity and magnetic permeability and for a Cartesian grid this method and the flux-splitting method become identical to the original characteristic-based Courant–Isaacson–Rees scheme [7]. We prefer in this context the Godunov approach for two reasons: First, this approach gives a clear picture and may be an obvious starting point for how to extend the algorithm to nonhomogeneous media with discontinuous permitivity and magnetic permeability. Second, the difficulties arising in a proper implementation of the boundary conditions can be circumvented applying the Godunov idea. For that, fictitious boundary grid cells are introduced and the values of the physical variables in these cells are defined in such a way that the physical flux at the boundary is approximated correctly. To get high-order accuracy the solution inside each grid zone has to be reconstructed from its integral approximate values. We compare numerical results of total variation diminishing (TVD) interpolation scheme (see, e.g., [23, 17]) with those using the essentially nonoscillatory (ENO) interpolation of Harten and Osher [12] up to fifth-order accuracy in both time and space.

Studying kinetic phenomena of charge particles in plasma physics, the numerical solution of the Maxwell–Lorentz equations has to be computed, where the Maxwell solver is used as one part of a complex simulation program. An attractive numerical method to solve this time-dependent nonlinear Maxwell–Lorentz problem is the particle-in-cell (PIC) approach based on particle-mesh techniques [2, 14]. The basic idea of this PIC method can be summarized as follows: The electromagnetic fields obtained by the numerical solution of the Maxwell equations are interpolated to the actual locations of the charged particles, representing the considered plasma. According to the Lorentz force the charged particles are redistributed and the new phase space coordinates are determined by solving numerically the usual laws of dynamics. To close the chain of self-consistent interplay the particles have to be located with respect to the computational grid in order to determine the new charge and current density which act as sources for the Maxwell equations in the following iteration cy-

cle. It is a well known fact, that the different steps within the PIC treatment of charged particles introduce numerical errors and, consequently, charge conservation is not guaranteed exactly or at least approximately on this discrete level of approximation. Hence, sophisticated and appropriate correction methods have to be applied to obtain physically relevant results from the numerical solution. In the present work a new divergence correction technique to preserve locally the charge conservation is formulated in the framework of the considered high-resolution schemes.

2. The Maxwell equations.

2.1. The classical form. The time-dependent Maxwell equations are usually formulated as

$$\partial_t \mathbf{E} - c^2 \nabla \times \mathbf{B} = -\frac{\mathbf{j}}{\epsilon_0}\,, \tag{2.1a}$$

$$\partial_t \mathbf{B} + \nabla \times \mathbf{E} = 0\,, \tag{2.1b}$$

$$\nabla \cdot \mathbf{E} = \frac{\rho}{\epsilon_0}\,, \tag{2.1c}$$

$$\nabla \cdot \mathbf{B} = 0\,, \tag{2.1d}$$

where $\mathbf{E}, \mathbf{B}, \rho$, and $\mathbf{j}$ denote the electric field, the magnetic induction, the charge, and the current density, respectively. For sake of clarity as well as with respect to our special purposes we consider a finite domain in a homogeneous medium where the permitivity and the magnetic permeability are constant and set equal to one. The electric and magnetic field constants in vacuum are related to the speed of light according to $\epsilon_0 \mu_0 c^2 = 1$.

It is well known that if the charge conservation equation

$$\partial_t \rho + \nabla \cdot \mathbf{j} = 0 \tag{2.2}$$

is valid for all times and if the initial data for $\mathbf{E}$ and $\mathbf{B}$ satisfy (2.1c) and (2.1d), respectively, then these relations also hold for all times. This follows directly from (2.1a), (2.1b) and the fact that the divergence of the curl of any (differentiable) vector-field equals zero. The evolution equations (2.1a), (2.1b) are hyperbolic and the Cauchy problem describing initial data in the whole space is well-posed. In practical calculations a bounded computational domain must be dealt with and boundary values have to be prescribed. These boundary conditions and their proper numerical implementation have to be based on the characteristic theory and are described in section 3.4.

2.2. The conservation form. The Maxwell equations (2.1a), (2.1b) may be written as a system of linear evolution equations

$$\partial_t \mathbf{u} + \mathcal{K}_1 \partial_{x_1} \mathbf{u} + \mathcal{K}_2 \partial_{x_2} \mathbf{u} + \mathcal{K}_3 \partial_{x_3} \mathbf{u} = \mathbf{q}(\mathbf{u}), \tag{2.3}$$

where $\mathbf{u}$ is the vector of the electric field and magnetic induction

$$\mathbf{u} = (E_1, E_2, E_3, B_1, B_2, B_3)^{\mathrm{T}},$$

and the 6×6 matrices $\mathcal{K}_i$ are defined as

$$\mathcal{K}_i = \begin{pmatrix} \mathbf{0} & -c^2 \mathcal{M}_i \\ \mathcal{M}_i & \mathbf{0} \end{pmatrix}, \quad i = 1, 2, 3, \tag{2.4}$$

with

$$\mathcal{M}_1 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{pmatrix}, \ \mathcal{M}_2 = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{pmatrix}, \ \mathcal{M}_3 = \frac{1}{x_2^\kappa} \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}. \tag{2.5}$$

We consider the Maxwell equations for Cartesian coordinates $\mathbf{x} = (x_1, x_2, x_3) = (x, y, z)$ as well as cylindrical coordinates $\mathbf{x} = (x_1, x_2, x_3) = (z, r, \phi)$. The parameter κ in (2.5) allows a switch between both coordinate systems, namely, $\kappa = 0$ for Cartesian and $\kappa = 1$ for cylindrical coordinates. The right-hand side $\mathbf{q}$ of (2.3) contains the current density and terms introduced by the geometry and is given by

$$\mathbf{q}(\mathbf{u}) = -\frac{1}{\epsilon_0} \left(j_1, j_2, j_3, 0, 0, 0\right)^T + \frac{\kappa}{x_2} \left(c^2 B_3, 0, 0, -E_3, 0, 0\right)^T. \tag{2.6}$$

Since the matrices $\mathcal{K}_i$ are constant, the Maxwell equations may be written in conservation form,

$$\partial_t \mathbf{u} + \partial_{x_1} \mathbf{f}_1(\mathbf{u}) + \partial_{x_2} \mathbf{f}_2(\mathbf{u}) + \partial_{x_3} \mathbf{f}_3(\mathbf{u}) = \mathbf{q}(\mathbf{u}), \tag{2.7}$$

with $\mathbf{f}_i(\mathbf{u}) = \mathcal{K}_i \mathbf{u}$, $i = 1, 2, 3$, which is the basis for the construction of finite-volume methods. The new aspect in this formulation is that the differential operator is considered as the divergence applied component-by-component to the physical fluxes $\mathbf{f}_i(\mathbf{u})$.

2.3. The constrained formulation. Numerical methods in the time domain are usually based on the evolution equations (2.1a), (2.1b) only. But, in the numerical approximation the relations (2.1c) and (2.1d) are, in general, not preserved exactly. Due to approximation errors the discrete divergence of the discrete curl operator may be nonzero:

$$\widetilde{\nabla} \cdot \left(\widetilde{\nabla} \times\right) = O(h^k), \tag{2.8}$$

where $\widetilde{\nabla}\cdot, \widetilde{\nabla}\times$ denote the discrete divergence and curl, respectively, h denotes some typical space increment, and k denotes some order. In pure field calculations where the charge conservation equation (2.2) is satisfied exactly, this seems not to be a severe problem, because this defect does not drastically increase in time and falsify the results [24, 4]. A quite different situation occurs when the Maxwell solver is applied as one part of a PIC code for the self-consistent simulation of charged particles. In the PIC approach the charge and current density are assigned to the grid points applying special interpolation schemes (see, e.g., [2, 14]). The motion of the particles is obtained by solving the Lorentz equations numerically. Both steps introduce numerical errors and a discrete analogue of the charge conservation equation (2.2) is not guaranteed exactly or even with a defect of the order $O(h^k)$. Since only the current density is used for the numerical field calculation based on (2.1a), (2.1b), the consistency of $\nabla \cdot \mathbf{E}$ with the charge density ρ may be lost. One way to get physically relevant results is to introduce sophisticated particle motion and current density approximations (see, e.g., [35]) and charge and current density assignment [9] to the numerically caused lack in (2.2). Another possibility to achieve consistency is to impose the divergence conditions (2.1c), (2.1d) directly by the introduction of correcting potentials [2].

Here, we follow the approach of Assous et al. [1] and introduce Lagrange multipliers which may be considered potentials correcting the electric field and the magnetic

induction. This constrained formulation reads as

$$\partial_t \mathbf{E} - c^2 \nabla \times \mathbf{B} + c^2 \nabla \phi = -\frac{\mathbf{j}}{\epsilon_0}\,, \tag{2.9a}$$

$$\partial_t \mathbf{B} + \nabla \times \mathbf{E} + \nabla p = 0\,, \tag{2.9b}$$

$$\nabla \cdot \mathbf{E} = \frac{\rho}{\epsilon_0}\,, \tag{2.9c}$$

$$\nabla \cdot \mathbf{B} = 0\,, \tag{2.9d}$$

where ϕ and p denote the potentials to meet the conditions (2.1c), (2.1d). Assuming that $\mathbf{E}$ and $\mathbf{B}$ satisfy the correct boundary conditions we require homogeneous Dirichlet boundary conditions on p and ϕ:

$$\begin{array}{l} \phi(\mathbf{x},t)=0 \\ p(\mathbf{x},t)=0 \end{array} \qquad \text{for } \mathbf{x} \in \Gamma \text{ and all } t, \tag{2.10}$$

where Γ denotes the boundary of the domain G. If the conservation equation (2.2) for ρ and $\mathbf{j}$ is satisfied, then (2.9) is equivalent to (2.1). This becomes obvious by applying the divergence to (2.9a), (2.1b) and using (2.2) which results in Laplace equations for ϕ and p. On account of the boundary conditions (2.10) this directly yields $\phi \equiv 0$ and $p \equiv 0$.

3. The finite-volume approach.

3.1. Integral formulation and discretization. A finite-volume scheme approximates the conservation equations (2.7) integrated over the grid cells. In the following we restrict ourselves to two space dimensions and assume that $\mathbf{E}$ and $\mathbf{B}$ do not depend on x_3. We consider a structured boundary-fitted grid, where the physical area G is covered by quadrilateral grid zones $V_{ij} : G = \bigcup_{ij} V_{ij}$. Integrating the system (2.7) component-by-component over the space-time volume $V_{ij} \times [t_n, t_{n+1}]$ and applying Gauß's theorem to the integral over the divergence of the flux components one obtains

$$\mathbf{u}_{ij}^{n+1} = \mathbf{u}_{ij}^{n} - \frac{1}{|V_{ij}|} \int_{t_n}^{t_{n+1}} \oint_{\partial V_{ij}} \mathcal{F}(\mathbf{u})\, \mathbf{n}_{ij}\, dS\, dt + \frac{1}{|V_{ij}|} \int_{t_n}^{t_{n+1}} \int_{V_{ij}} \mathbf{q}(\mathbf{u})\, dV\, dt. \tag{3.1}$$

Here, $\mathbf{u}_{ij}^n$ denotes the cell average of $\mathbf{u}$ over the cell V_{ij} with area $|V_{ij}|$ at time t_n:

$$\mathbf{u}_{ij}^{n} = \frac{1}{|V_{ij}|} \int_{V_{ij}} \mathbf{u}(\mathbf{x}, t_n)\, dV, \tag{3.2}$$

$\mathcal{F}(\mathbf{u})$ denotes the 6×2 flux matrix $(\mathbf{f}_1(\mathbf{u}), \mathbf{f}_2(\mathbf{u}))$, and $\mathbf{n}_{ij} \in \mathbb{R}^2$ denotes the outwards directed normal at the boundary ∂V_{ij} of grid cell V_{ij}. As sketched out in Figure 1, the boundary of a quadrilateral grid cell is given by $\partial V_{ij} = \sum_{\beta=1}^{4} S_{ij,\beta}$ and the line integral in (3.1) can be replaced by

$$\oint_{\partial V_{ij}} \mathcal{F}(\mathbf{u})\, \mathbf{n}_{ij}\, dS = \sum_{\beta=1}^{4} \int_{S_{ij,\beta}} \mathcal{F}(\mathbf{u})\, \mathbf{n}_{ij,\beta}\, dS, \tag{3.3}$$

where $\mathbf{n}_{ij,\beta} = ((n_1)_{ij,\beta}, (n_2)_{ij,\beta})^{\mathrm{T}}$ denotes the outwards directed unit normal at the edge $S_{ij,\beta}$. The integral equations (3.1) establish exact evolution equations for the mean values of the solution.

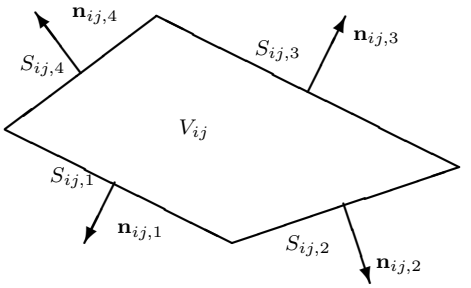


FIG. 1. *Quadrilateral grid zone V_{ij} with boundary ∂V_{ij} and outwards directed normals $\mathbf{n}_{ij,\beta}$.*

A finite-volume scheme is a direct approximation of the integral formulation (3.1) and is usually written in the form

$$\mathbf{u}_{ij}^{n+1} = \mathbf{u}_{ij}^{n} - \frac{\Delta t}{|V_{ij}|} \sum_{\beta=1}^{4} \mathbf{G}_{ij,\beta}^{n} + \Delta t \mathbf{q}_{ij}^{n+1/2}, \tag{3.4}$$

where the so-called numerical flux $\mathbf{G}_{ij,\beta}$ is an approximation of the physical flux $\mathcal{F}(\mathbf{u})$ through the boundary segment $S_{ij,\beta}$:

$$\mathbf{G}_{ij,\beta}^{n} \approx \frac{1}{\Delta t} \int_{t_n}^{t_{n+1}} \int_{S_{ij,\beta}} \mathcal{F}(\mathbf{u}) \mathbf{n}_{ij,\beta} \, dS \, dt. \tag{3.5}$$

The vector $\mathbf{q}_{ij}^{n+1/2}$ approximates the source terms averaged over V_{ij} and the time interval $[t_n, t_{n+1}]$. The main task in the context of finite-volume schemes is to find a suitable numerical flux (3.5) as a function of the averaged quantities. This is described in the next section. As indicated by the superscript "n" of the numerical flux in (3.4) we are interested in an explicit approximation using the values at the old time level t_n.

For the approximation of the source term we apply in the following a splitting method as proposed by Marchuk [20]. In each time step, first the homogeneous Maxwell equations are solved neglecting the term $\mathbf{q}_{ij}^{n+\frac{1}{2}}$ in (3.4). The values obtained are then used as initial values for the numerical solution of the system of ordinary differential equations

$$\partial_t \mathbf{u} = \mathbf{q}(\mathbf{u}) \tag{3.6}$$

introducing the sources into the scheme. To solve (3.6) a standard ordinary differential equation solver may be chosen. We remark, that if the source step (3.6) as well as the Maxwell step (3.4) are solved accurately up to second-order in time, then the so-called Strang splitting should be applied to preserve the overall second-order accuracy of the algorithm (see [20, 31]). Here, the order of the different steps is changed in each time cycle.

3.2. The numerical flux calculation. The finite-volume method is completely defined by specifying the numerical fluxes (3.5). First, the integral over the boundary

segments has to be approximated. In the following, we restrict ourselves for the sake of simplicity to second-order accuracy and use the midpoint rule,

$$\mathbf{G}^n_{ij,\beta} = |S_{ij,\beta}|\,((n_1)_{ij,\beta}\,\mathcal{K}_1 + (n_2)_{ij,\beta}\,\mathcal{K}_2)\;\mathbf{u}(\mathbf{x}_{MP,\beta}, t_n), \tag{3.7}$$

where $\mathbf{x}_{MP,\beta}$ denotes the midpoint of edge β. Clearly, a higher order accurate numerical quadrature formula has to be chosen for higher order schemes. Next, an appropriate approximation of the solution at this midpoint has to be calculated. Here, we follow the ideas of Godunov [10] who developed this method for the equations of gas dynamics. During the last years, the Godunov method has been successfully applied to other systems of conservation laws. For a review, we refer to the excellent book of LeVeque [17]. Assuming the approximation to be piecewise constant at time level t_n and equal to the integral values in each grid zone, the local wave propagation at the point $\mathbf{x}_{\mathrm{MP},\beta}$ into normal direction is determined by the break-up of the discontinuity at the grid interface $S_{ij,\beta}$. Information about this is provided by the local solution of a Riemann problem which is an initial-value problem of the form

$$\partial_t \mathbf{u} + \mathcal{C}_{ij,\beta}\,\partial_\zeta \mathbf{u} = 0, \tag{3.8}$$

$$\mathbf{u}(\zeta, 0) = \begin{cases} \mathbf{u}_l & \text{for } \zeta < 0, \\ \mathbf{u}_r & \text{for } \zeta > 0, \end{cases} \tag{3.9}$$

with

$$\mathcal{C}_{ij,\beta} = (n_1)_{ij,\beta}\,\mathcal{K}_1 + (n_2)_{ij,\beta}\,\mathcal{K}_2. \tag{3.10}$$

The values $\mathbf{u}_l$ and $\mathbf{u}_r$ are the states in the grid zone (i, j) and its neighbor cell according to the value of β. The flux at $\zeta = 0$, generated by the solution of the Riemann problem and multiplied by the length $|S_{ij,\beta}|$ is then defined to be the numerical flux (3.7) of the Godunov scheme.

The solution of the Riemann problem (3.8), (3.9) is obtained by the theory of characteristics and is briefly summarized in the following. The matrix $\mathcal{C}_{ij,\beta}$ has the three different, double-valued real eigenvalues

$$\lambda_1 = \lambda_2 = -c, \qquad \lambda_3 = \lambda_4 = 0, \qquad \lambda_5 = \lambda_6 = c \tag{3.11}$$

and a complete set of right eigenvectors $\mathbf{r}^1, \ldots, \mathbf{r}^6$, which may be written as columns of the matrix

$$\mathcal{R} = \left(\mathbf{r}^1, \ldots, \mathbf{r}^6\right) = \begin{pmatrix} bc & 0 & a & 0 & -bc & 0 \\ -ac & 0 & b & 0 & ac & 0 \\ 0 & c & 0 & 0 & 0 & -c \\ 0 & -b & 0 & a & 0 & -b \\ 0 & a & 0 & b & 0 & a \\ 1 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}, \tag{3.12}$$

with $a := (n_1)_{ij,\beta}, b := (n_2)_{ij,\beta}$. The corresponding left eigenvectors $\mathbf{l}^1, \ldots, \mathbf{l}^6$ are the rows of the matrix

$$\mathcal{R}^{-1} = \left(\mathbf{l}^1, \ldots, \mathbf{l}^6\right)^{\mathrm{T}} = \frac{1}{2c} \begin{pmatrix} b & -a & 0 & 0 & 0 & c \\ 0 & 0 & 1 & -bc & ac & 0 \\ 2ac & 2bc & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2ac & 2bc & 0 \\ -b & a & 0 & 0 & 0 & c \\ 0 & 0 & -1 & -bc & ac & 0 \end{pmatrix}. \tag{3.13}$$

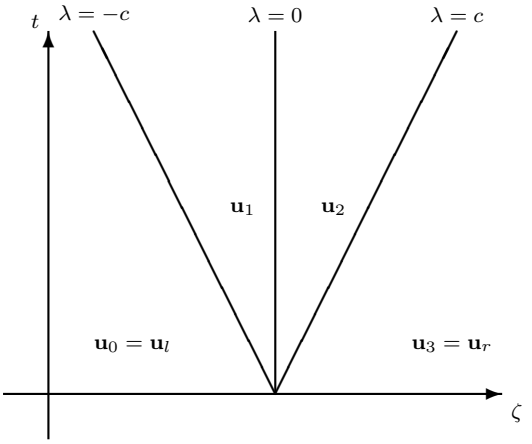


FIG. 2. *(ζ, t)-diagram of the exact solution of the Riemann problem. The solution consists of four constant states $\mathbf{u}_l, \mathbf{u}_1, \mathbf{u}_2$, and $\mathbf{u}_r$ separated by the characteristics.*

Figure 2 shows the (ζ, t)-diagram of the exact solution of the Riemann problem. It consists of four constant states separated by the three characteristics. The intermediate constant states are given by

$$\mathbf{u}_1 = \mathbf{u}_l + \sum_{j=1}^{2} \alpha_j \mathbf{r}^j = \mathbf{u}_r - \sum_{j=0}^{3} \alpha_{6-j} \mathbf{r}^{6-j}, \tag{3.14}$$

$$\mathbf{u}_2 = \mathbf{u}_l + \sum_{j=1}^{4} \alpha_j \mathbf{r}^j = \mathbf{u}_r - \sum_{j=0}^{1} \alpha_{6-j} \mathbf{r}^{6-j}, \tag{3.15}$$

with coefficients

$$\alpha_j = \mathbf{l}_j(\mathbf{u}_r - \mathbf{u}_l), \quad j = 1, \ldots, 6. \tag{3.16}$$

The numerical flux can now be calculated as

$$\mathbf{G}^n_{ij,\beta} = |S_{ij,\beta}|\, \mathcal{C}_{ij,\beta} \mathbf{u}_1 = |S_{ij,\beta}|\, \mathcal{C}_{ij,\beta} \mathbf{u}_2, \tag{3.17}$$

which can be shown in a straightforward manner using the definitions (3.10), (3.14), and (3.15).

We remark that (3.17) can be rewritten in several forms. In the context of this paper we prefer the flux-vector splitting form given by

$$\mathbf{G}^n_{ij,\beta} = |S_{ij,\beta}| \left(\mathcal{C}^+_{ij,\beta} \mathbf{u}_l + \mathcal{C}^-_{ij,\beta} \mathbf{u}_r \right), \tag{3.18}$$

with

$$\mathcal{C}^{\pm}_{ij,\beta} = \frac{1}{2} \left(\mathcal{C}_{ij,\beta} \pm |\mathcal{C}_{ij,\beta}| \right), \tag{3.19}$$

where the absolute value, $|\mathcal{C}_{ij,\beta}|$, is defined with the aid of the diagonalized matrix according to

$$|\mathcal{C}_{ij,\beta}| = \mathcal{R}\ \mathrm{diag}(c, c, 0, 0, c, c)\ \mathcal{R}^{-1} \tag{3.20}$$

with $\mathcal{R}$ and $\mathcal{R}^{-1}$ given by (3.12) and (3.13), respectively. In this formulation the total flux is decomposed into a flux to the "right" associated with $\mathcal{C}^{+}_{ij,\beta}$, having nonnegative eigenvalues only, and a flux to the "left" associated with $\mathcal{C}^{-}_{ij,\beta}$, having nonpositive eigenvalues only (see [30]). The first matrix acts on $\mathbf{u}_l$, while the other acts on $\mathbf{u}_r$, taking into account the correct domain of influence.

All these considerations are valid only if it is guaranteed that the waves generated at different grid zone interfaces do not interact. This leads to a CFL time step restriction in the form $c\frac{\Delta t}{h} \leq 1$, where h denotes the smallest length of a grid zone. For a Cartesian grid and for smooth solutions the finite-volume scheme defined in this way agrees with the Courant–Isaacson–Rees scheme [7], which is based on the characteristic form of the equations. The Godunov scheme may be considered as an extension of this scheme to general grids. Since the direction of the wave propagation is taken into account, this numerical scheme is inherently very robust and able to resolve steep gradients without generating spurious oscillations. However, it is only of first-order accuracy in both space and time and introduces too much numerical dissipation for practical calculations. In the following section this lack is removed by an extension of the scheme to higher order accuracy.

3.3. Higher order accuracy. To get higher order accuracy in space the reconstruction from the approximate integral values has to be improved. This idea has been proposed by van Leer [34] and is called MUSCL approach. In the following, we restrict ourselves to a piecewise linear reconstruction, which establishes second-order accuracy. Two extensions of the MUSCL approach to boundary-fitted grids are shortly described.

The first extension has been proposed by Durlofsky, Engquist, and Osher [8] for triangular grid cells. It can be adapted to quadrilateral grids in a straightforward way. The linear representation of each component $w \in \{E_1, E_2, E_3, B_1, B_2, B_3\}$ is given by the truncated Taylor expansion

$$w^n(\mathbf{x}) = w^n_{ij} + (\mathbf{x} - \mathbf{x}_{ij}) \cdot \widetilde{\nabla} w^n_{ij} \qquad \text{for } \mathbf{x} \in V_{ij}, \tag{3.21}$$

where $\mathbf{x}_{ij}$ denotes the barycenter of cell (i,j) and $\widetilde{\nabla} w^n_{ij}$ an approximation of the gradient of component w in cell (i,j) at time t_n. To approximate this gradient, the field averages of the cell under consideration and of two of its neighbors are linearly interpolated (see Figure 3). $\widetilde{\nabla}^{(1)} w^n_{ij}$ is the gradient of the plane through $(\mathbf{x}_{ij}, \mathbf{u}^n_{ij}), (\mathbf{x}_{i+1,j}, \mathbf{u}^n_{i+1,j})$, and $(\mathbf{x}_{i,j+1}, \mathbf{u}^n_{i,j+1})$. Similarly, $\widetilde{\nabla}^{(2)} w^n_{ij}$ is obtained using $\mathbf{u}^n_{ij}; \mathbf{u}^n_{i,j+1}$ and $\mathbf{u}^n_{i-1,j}$; $\widetilde{\nabla}^{(3)} w^n_{ij}$ using $\mathbf{u}^n_{ij}, \mathbf{u}^n_{i-1,j}$, and $\mathbf{u}^n_{i,j-1}$; and $\widetilde{\nabla}^{(4)} w^n_{ij}$ using $\mathbf{u}^n_{ij}, \mathbf{u}^n_{i,j-1}$, and $\mathbf{u}^n_{i+1,j}$ and the coordinates of the corresponding barycenters (see Figure 3). To avoid spurious oscillations near steep gradients the one in this set with the smallest norm is used in (3.21). For more details see [8].

To enhance the time accuracy a truncated Taylor expansion also in t is used where the necessary approximation of the time derivative of $\mathbf{u}$ at time t_n is replaced by approximations of the space derivatives according to the differential equations

$$\mathbf{u}^{n+\frac{1}{2}}(\mathbf{x}) = \mathbf{u}^n(\mathbf{x}) - \frac{\Delta t}{2}\left(\mathcal{K}_1 \widetilde{\partial}_{x_1} \mathbf{u}^n + \mathcal{K}_2 \widetilde{\partial}_{x_2} \mathbf{u}^n\right), \tag{3.22}$$

where $\widetilde{\partial}_{x_1} \mathbf{u}$ and $\widetilde{\partial}_{x_2} \mathbf{u}$ denote the approximations of the space derivatives as calculated in (3.21).

This technique for enhancing the accuracy with respect to time has first been used in the Lax–Wendroff scheme (see, e.g., [17]) and, hence, is often called Lax–Wendroff

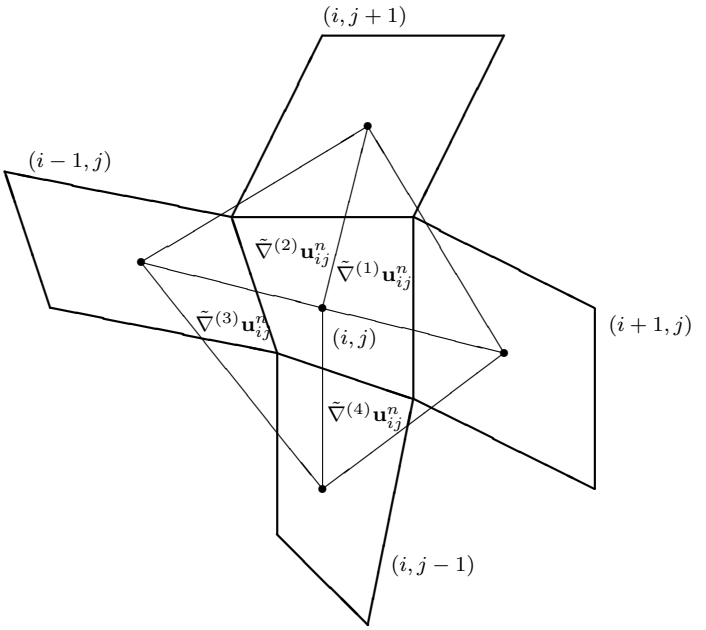


FIG. 3. *Candidates for slope calculation in the ansatz of Durlofsky et al.*

procedure. The values of the interpolation polynomials at the midpoint of a side face from both sides are then used as data of the Riemann problem (3.8), (3.9) to determine the numerical flux. However, the application of the Durlofsky, Engquist, and Osher ansatz to quadrilateral cells makes use of the structure of the grid only with regard to data administration.

The second ansatz, proposed by Colella in [6], refers to more information about the grid structure. He proposed a two-dimensional MUSCL scheme without splitting, also extended to boundary-fitted coordinates requiring the existence of a smooth coordinate mapping $(\xi, \eta) \mapsto (x, y)$ between the coordinate space and the physical space with a rectangular mesh in the coordinate space (ξ, η). Here, the slope is assumed to be constant in each component along the lines $\xi = \text{const.}$ and $\eta = \text{const.}$ These constant vectors are denoted by $\widetilde{\mathbf{u}_\xi}$ and $\widetilde{\mathbf{u}_\eta}$. To avoid spurious oscillations near steep gradients the slopes have to be calculated carefully. One possibility is to compare the left and right difference quotients and to take that one with the smaller absolute value if their signs coincide. For different signs the slope is set equal to zero. This is the Minmod limiting procedure:

$$\widetilde{\mathbf{u}_\xi} = \text{Minmod}\left(\frac{\mathbf{u}_{i+1,j} - \mathbf{u}_{ij}}{\|\mathbf{x}_{i+1,j} - \mathbf{x}_{ij}\|}, \frac{\mathbf{u}_{ij} - \mathbf{u}_{i-1,j}}{\|\mathbf{x}_{ij} - \mathbf{x}_{i-1,j}\|}\right), \tag{3.23}$$

$$\widetilde{\mathbf{u}_\eta} = \text{Minmod}\left(\frac{\mathbf{u}_{i,j+1} - \mathbf{u}_{ij}}{\|\mathbf{x}_{i,j+1} - \mathbf{x}_{ij}\|}, \frac{\mathbf{u}_{ij} - \mathbf{u}_{i,j-1}}{\|\mathbf{x}_{ij} - \mathbf{x}_{i,j-1}\|}\right). \tag{3.24}$$

Further appropriate slope calculations are reviewed, e.g., in [17], [23]. All procedures have the TVD property which guarantees for one-dimensional scalar problems that the total variation of the solution does not increase in time. The values $\mathbf{u}^n_{i\pm,j}, \mathbf{u}^n_{i,j\pm}$

at the cell's boundary are obtained by a one-dimensional setting (see [34]) for each direction separately:

$$\mathbf{u}^n_{i\pm,j} = \mathbf{u}^n_{ij} \pm \frac{|\mathbf{x}_{ij} - \mathbf{x}_{i\pm\frac{1}{2},j}|}{2}(\widetilde{\mathbf{u}_\xi})^n_{ij}, \tag{3.25}$$

$$\mathbf{u}^n_{i,j\pm} = \mathbf{u}^n_{ij} \pm \frac{|\mathbf{x}_{ij} - \mathbf{x}_{i,j\pm\frac{1}{2}}|}{2}(\widetilde{\mathbf{u}_\eta})^n_{ij}, \tag{3.26}$$

where $\mathbf{x}_{i\pm\frac{1}{2},j}$ and $\mathbf{x}_{i,j\pm\frac{1}{2}}$ denote the midpoints of the boundary segments. For the temporal extrapolation

$$\mathbf{u}^{n+1/2}_{i\pm,j} = \mathbf{u}^n_{i\pm,j} + \frac{\Delta t}{2}(\widetilde{\mathbf{u}_t})_{ij}, \tag{3.27}$$

$$\mathbf{u}^{n+1/2}_{i,j\pm} = \mathbf{u}^n_{i,j\pm} + \frac{\Delta t}{2}(\widetilde{\mathbf{u}_t})_{ij} \tag{3.28}$$

via the Lax–Wendroff procedure now the equations transformed onto the (ξ, η) curved coordinate system

$$\mathbf{u}_t + \frac{(y_\eta \mathcal{A}\mathbf{u} - x_\eta \mathcal{B}\mathbf{u})_\xi}{\det J} + \frac{(-y_\xi \mathcal{A}\mathbf{u} + x_\xi \mathcal{B}\mathbf{u})_\eta}{\det J} = 0 \tag{3.29}$$

have to be used to replace the time derivatives by space derivatives. Here, x_ξ, x_η, y_ξ, and y_η denote the partial derivatives of the transformation with respect to ξ and η and $\det J$ denotes the determinant of the Jacobi matrix of the transformation.

These second-order TVD schemes are quite robust but introduce some clipping of extrema. Harten and Osher [12] succeeded to construct a higher order ENO reconstruction of the solution. The ENO interpolation procedure minimizes numerical oscillations near strong gradients by using data from the smoothest part of the solution. First, a table of divided differences is calculated and a searching algorithm determines which portion of the surrounding solution is smoothest. The stencils of the interpolation polynomial chosen are allowed to shift to this region—away from the larger differences. For more details we refer to [11, 12]. Also to get higher than second-order accuracy in time the application of the Lax–Wendroff procedure seems to be no longer favorable because space derivatives of higher order occur which also have to be approximated in an appropriate way. A more convenient possibility is to use a Runge–Kutta time approximation as proposed by Shu and Osher [28]. A third-order Runge–Kutta scheme, eg., is given by

$$\begin{aligned} u^{(1)} &= u^{(n)} + \Delta t L(u^{(n)}), \\ u^{(2)} &= \frac{3}{4}u^{(n)} + \frac{1}{4}u^{(1)} + \frac{1}{4}\Delta t L(u^{(1)}), \\ u^{(n+1)} &= \frac{1}{3}u^{(n)} + \frac{2}{3}u^{(2)} + \frac{2}{3}\Delta t L(u^{(2)}), \end{aligned} \tag{3.30}$$

where L denotes the spatial approximation, and $u^{(1)}$, $u^{(2)}$ the intermediate steps.

3.4. Implementation of boundary conditions. The Maxwell equations form a system of hyperbolic equations and hence boundary conditions and their implementation are only well-posed if they take into account the wave propagation as given by the theory of characteristics. In the presented finite-volume approach this is established again using the solution of the Riemann problem. At the grid cells adjacent

to the boundaries of the computational domain initial-boundary-value problems instead of Riemann problems have to be solved locally. Our goal is to reformulate these initial-boundary-value problems as Riemann problems. For that, first fictitious grid zones are introduced around the computational domain, called dummy grid cells. Then in these grid zones values for the physical variables are specified in such a way that the solution of the Riemann problem at the boundary yields the proper physical conditions.

For this, we need the solution of the Riemann problem (3.8), (3.9) at $\zeta = 0$. As there occur vanishing eigenvalues, the boundary $\zeta = 0$ coincides with the characteristic curve separating the two intermediate states $\mathbf{u}_1$ and $\mathbf{u}_2$ according to (3.14) and (3.15), respectively. The value $\mathbf{u}_0 = (E_{1,0}, E_{2,0}, \ldots, B_{3,0})^{\mathrm{T}}$ of the Riemann solution $\mathbf{u}$ at $\zeta = 0$ can be defined to be $\mathbf{u}_1$ or $\mathbf{u}_2$, which does not affect the flux calculation (3.17). For symmetry reasons we define $\mathbf{u}_0$ to be the arithmetic average of $\mathbf{u}_1$ and $\mathbf{u}_2$. With $\mathbf{n} = (a, b, 0)^{\mathrm{T}}$ the outer normal at the boundary, $\mathbf{u}_0$ is given by

$$\mathbf{u}_0 = \frac{1}{2} \begin{pmatrix} E_{1,r} + E_{1,l} + bc(B_{3,r} - B_{3,l}) \\ E_{2,r} + E_{2,l} - ac(B_{3,r} - B_{3,l}) \\ E_{3,r} + E_{3,l} - bc(B_{1,r} - B_{1,l}) + ac(B_{2,r} - B_{2,l}) \\ B_{1,r} + B_{1,l} - \frac{b}{c}(E_{3,r} - E_{3,l}) \\ B_{2,r} + B_{2,l} + \frac{a}{c}(E_{3,r} - E_{3,l}) \\ B_{3,r} + B_{3,l} + \frac{b}{c}(E_{1,r} - E_{1,l}) - \frac{a}{c}(E_{2,r} - E_{2,l}) \end{pmatrix}. \tag{3.31}$$

Considering, for instance, a boundary on the left-hand side of the computational domain, (3.31) has to be solved for the dummy state $\mathbf{u}_l$. First, we remark that there exists no uniquely determined solution. Rather, there are an infinite number of possibilities to define the dummy state $\mathbf{u}_r$. This arbitrariness reflects the fact that in the dummy cell the characteristic variables corresponding to waves which do not enter the computational domain have no influence on the solution at the boundary. These arbitrary values in the dummy cells also do not affect the numerical values because the characteristic-based flux calculation automatically picks out the proper information. Secondly, boundary conditions can not be imposed independently of the state inside the last cell of the computational domain. This reflects the fact that the part of characteristic information at the boundary, which is transported from the last cell inside the domain to the boundary, must coincide with the values prescribed there. For physical relevant boundary conditions, however, the latter is automatically met.

The main advantage of the outlined technique is that imposing the boundary conditions concerns only the dummy cells and can be managed in a first step of the calculation cycle. Then the discrete equations can be solved without any further modification in the whole computational domain, leading to a highly vectorizable algorithm. Furthermore, the exchange of boundary values necessary for parallelization based on domain decomposition can be handled in a similar way. In the following we discuss the boundary conditions for perfect electrical conducting surfaces, for symmetry, irradiation of electromagnetic energy into the domain, and for open boundaries which are introduced to limit artificially the computational domain. In the latter case, waves should propagate out of the domain without generating reflections. All boundary conditions are formulated for the case of curvilinear coordinates.

First, we consider the boundary condition of a perfect conducting wall, where the fields cannot penetrate into the surface. In this case the physical boundary conditions

read as

$$\mathbf{n} \times \mathbf{E}_0 = 0, \qquad \mathbf{n} \cdot \mathbf{B}_0 = 0. \tag{3.32}$$

These conditions lead to

$$aE_{2,0} - bE_{1,0} = 0, \qquad E_{3,0} = 0, \qquad aB_{1,0} + bB_{2,0} = 0. \tag{3.33}$$

For a given state $\mathbf{u}_r$ or $\mathbf{u}_l$, the other one has to be defined in such a way that the relations (3.33) are valid at the boundary. In the case, where $\mathbf{u}_r$ is given, a possible choice for $\mathbf{u}_l$ is

$$\begin{aligned} E_{1,l} = -E_{1,r}, \quad E_{2,l} = -E_{2,r}, \quad E_{3,l} = -E_{3,r}, \quad B_{3,l} = B_{3,r}, \\ B_{1,l} = (b^2 - a^2)B_{1,r} - 2abB_{2,r}, \\ B_{2,l} = (a^2 - b^2)B_{2,r} - 2abB_{1,r}. \end{aligned} \tag{3.34}$$

Now we assume that the boundary of interest is an axis of symmetry which does not coincide with lines parallel to the x_1- or x_2-axis. At this axis the following conditions

$$E_{3,0} = 0, \qquad \mathbf{n} \cdot \mathbf{E}_0 = 0 \tag{3.35}$$

and

$$B_{3,0} = 0, \qquad \mathbf{n} \cdot \mathbf{B}_0 = 0 \tag{3.36}$$

hold. As mentioned above, these conditions are not sufficient to find a solution for the fictitious grid zones from (3.31). With the choice

$$E_{3,l} = -E_{3,r}, \qquad B_{3,l} = -B_{3,r}, \tag{3.37}$$

however, the values of the other components of the electromagnetic fields in the dummy grid cells are obtained from (3.31) resulting in

$$E_{1,l} = (b^2 - a^2)E_{1,r} - 2abE_{2,r}, \quad E_{2,l} = (a^2 - b^2)E_{2,r} - 2abE_{1,r}, \tag{3.38a}$$

$$B_{1,l} = (b^2 - a^2)B_{1,r} - 2abB_{2,r}, \quad B_{2,l} = (a^2 - b^2)B_{2,r} - 2abB_{1,r}. \tag{3.38b}$$

Another situation occurs when the electrical device is driven by external fields irradiated at certain scheduled edges of the device. Since there exist two outgoing characteristics which transport information from the computational domain to the boundary two compatibility conditions have to be satisfied by the boundary conditions at $\zeta = 0$. Considering a left boundary, those are the characteristics associated with the negative eigenvalues and hence the first two components of the characteristic variable $\mathbf{v} = \mathcal{R}^{-1}\mathbf{u}$ have to satisfy

$$v_{1,0} = cB_{3,0} + bE_{1,0} - aE_{2,0} = cB_{3,r} + bE_{1,r} - aE_{2,r}, \tag{3.39}$$

$$v_{2,0} = E_{3,0} - cbB_{1,0} + caB_{2,0} = E_{3,r} - cbB_{1,r} + caB_{2,r}. \tag{3.40}$$

Therefore only two of the three values of the subsystems E_1, E_2, B_3 and B_1, B_2, E_3 can be prescribed. Assuming that the third component of the electric field and magnetic induction is not imposed explicitly, a possible choice is

$$B_{3,l} = B_{3,r}, \qquad E_{1,l} = 2E_{1,0} - E_{1,r}, \qquad E_{2,l} = 2E_{2,0} - E_{2,r} \tag{3.41}$$

and

$$E_{3,l} = E_{3,r}, \qquad B_{1,l} = 2B_{1,0} - B_{1,r}, \qquad B_{2,l} = 2B_{2,0} - B_{2,r}. \tag{3.42}$$

Open boundaries are conditions motivated by numerical simulations and necessary to limit artificially the computational domain. A criterion of an open or nonreflecting boundary has been formulated by Hedstrom [13]: No wave should travel into the computational domain which means that the amplitudes of incoming waves are constant with respect to time. Incoming and outgoing traveling waves can be easily identified by the use of characteristic variables as proposed by Thompson [33]. Considering a left boundary as before, the incoming waves are those associated with the positive eigenvalues. Then, the open boundary condition reads as

$$\partial_t v_{5,0} = \partial_t v_{6,0} = 0, \tag{3.43}$$

for all t, whereas the equations for the other four characteristic variables remain unchanged.

In the first step we calculate the physical solution at the boundary. The transformation back to the conserved variables $\mathbf{u} = \mathcal{R}\mathbf{v}$ shows that the solution satisfying (3.43) at the boundary is the solution of the Riemann problem

$$\partial_t \mathbf{u} + \mathcal{C}^- \partial_\zeta \mathbf{u} = 0, \tag{3.44}$$

$$\mathbf{u}(\zeta, 0) = \begin{cases} \mathbf{u}_l & \zeta < 0, \\ \mathbf{u}_r & \zeta > 0, \end{cases} \tag{3.45}$$

with $\mathcal{C}^-$ given by (3.19). This Riemann problem can be solved, yielding $\mathbf{u}_0 = \mathbf{u}_r$, as $\mathcal{C}^-$ has only nonpositive eigenvalues. Now we have the physical solution on the boundary as in the examples before, with the only difference, that the value on the boundary depends on the actual value inside the computational domain.

In the second step, now we proceed as before. Inserting the solution $\mathbf{u}_0 = \mathbf{u}_r$ into (3.31) gives $\mathbf{u}_l = \mathbf{u}_r$ as a possible choice for the dummy cells. We remark that this can be regarded as a special case of irradiation. The values of two characteristic quantities at the boundary have to remain constant, but the constant is not known. Depending on the outgoing information $\mathbf{u}_r$, the correct information is irradiated so that the outgoing information is "neutralized." We further emphasize that the proposed boundary treatment is based on the physical transport of information by its construction. As an upwind scheme the presented finite-volume approach models properly the physical transport and therefore the arbitrarily prescribed values have no influence on the numerical solution.

4. Numerical results. Typical test examples for Maxwell solvers are transverse magnetic (TM) and transverse electric (TE) waves, for which exact solutions are available (see [16]). In this case the Maxwell equations are homogeneous, i.e., $\mathbf{j} \equiv 0$ in (2.6). For the following TM problem we consider the computational domain $G = \{(x, y) |\ 0 \le x \le x_0, 0 \le y \le y_0\}$. The initial data are given by

$$E_3(\mathbf{x}, 0) = E_0 \sin(ax) \sin(by) \tag{4.1}$$

with

$$a = \frac{m\pi}{x_0}, \qquad b = \frac{n\pi}{y_0}, \tag{4.2}$$

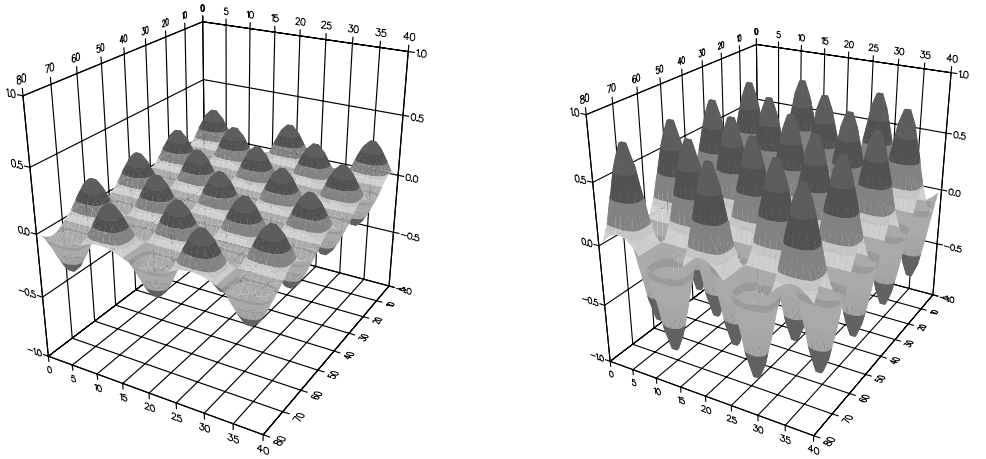


FIG. 4. *Numerical results for a TM wave calculation with $m = 8$ and $n = 5$. The plots show the spatial distribution of the component E_3 at $t = 75ns$ (left) and $t = 150ns$ (right).*

where $\frac{m}{2}$ and $\frac{n}{2}$ is the number of waves in x and y direction, respectively. All other components of $\mathbf{u}$ are set equal to zero everywhere. The exact solution of this initial-value problem is given by the solution of a wave equation for E_3, B_1 and B_2, and $E_1 \equiv E_2 \equiv B_3 \equiv 0$.

In the following we first show numerical results for a regular Cartesian grid with constant space increments, surrounded by a perfect conducting wall. Here, the computational domain G with $x_0 = 80\,\mathrm{m}$, $y_0 = 40\,\mathrm{m}$ is discretized by a grid of 80×40 grid zones. An overview of the numerical solution for this initial value problem is given in Figure 4 for $E_0 = 1\mathrm{V/m}$, $m = 8$, and $n = 5$. The temporal evolution of E_3 is depicted at the time $t = 75\,\mathrm{ns}$ (left) and $t = 150\,\mathrm{ns}$ (right), corresponding to 28 and 56 time steps, respectively.

A first glimpse of the ability of the different finite-volume schemes to capture these TM waves becomes obvious considering slices parallel to the x and y axis, respectively, at $y_0 = 20$ (upper part) and $x_0 = 40$ (lower part) in Figure 5. The plots show results at time $t = 75\,\mathrm{ns}$ (left) and $t = 150\,\mathrm{ns}$ (right). The first-order scheme $(_{+++})$ has already damped the oscillations nearly up to a constant at the time $t = 150\mathrm{ns}$. For practical calculations a first-order method seems not to be appropriate. The second-order TVD-MUSCL scheme clearly reproduces the shape of the wave but reduces the amplitude approximately by a factor 0.5 during the calculation. The slope calculation is based on Sweby's flux limiter function with $k = 1.5$ (see, e.g., [17, 23]). The clipping of extrema can clearly be seen at $t = 75\,\mathrm{ns}$. The ENO-reconstruction of third-order accuracy produces much better results. The form of the wave as well as the amplitude are reproduced very well. To get a more quantitative picture of the approximation quality of the different methods the L^2 error is calculated as a function of time. This error averaged over the whole time interval is listed in Table 1 for the E_3, B_1, and B_2 component.

A serious problem of many electromagnetic field calculations is the accurate resolution of high frequency waves. Here, the ability of the numerical method to capture waves accurately with a small number of grid points is important, especially, in multidimensional calculations to reduce memory requirement and computer time. Due to the reduction of the numerical damping the high-order methods are superior. In the

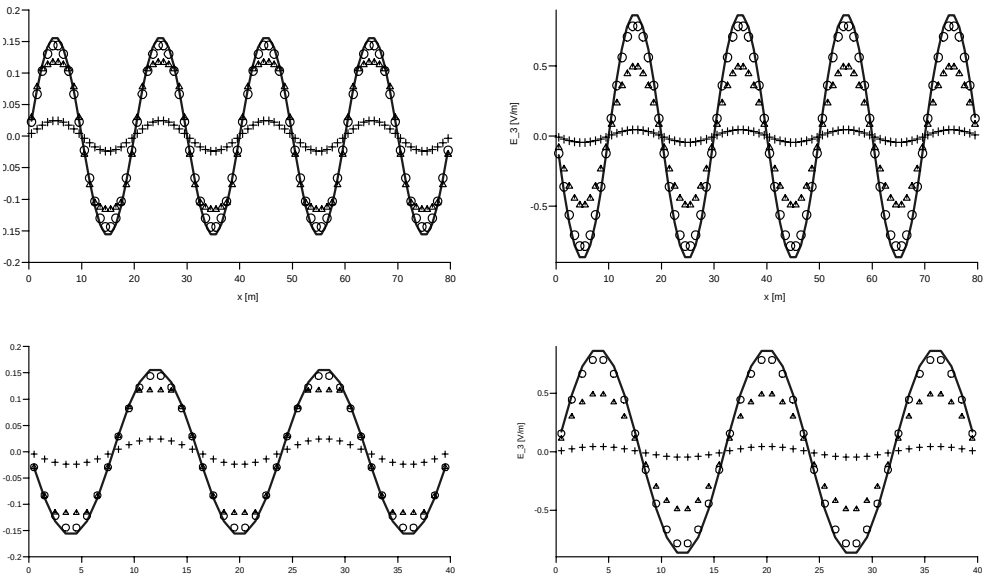


FIG. 5. *Comparison of numerical results for a TM wave calculation with $m = 8$ and $n = 5$ at $t = 75ns$ (left) and $t = 150ns$ (right). The upper plots show the slice of the E_3 component at $y_0 = 20\,m$, the lower plots show the slice at $x_0 = 40\,m$; the exact solution is plotted by a solid line, first-order and second-order TVD and third-order ENO schemes are plotted by "+ + +," "ΔΔΔ," and "o o o," respectively.*

TABLE 1

L^2-error averaged over the time intervall $0 \le t \le 150ns$ of the different versions of the finite-volume scheme calculated for the TM problem with $m = 8$ and $n = 5$.

	E3	B1	B2
1^{st} order	12.62	31.71	25.30
TVD	1.34	3.86	2.61
ENO3	0.90	2.23	1.77
ENO4	0.24	0.61	0.46
ENO5	0.08	0.22	0.18

following we study this behavior comparing a second-order TVD with higher order ENO schemes. For a specified value of the error we determine how many grid points per wavelength are necessary for the different methods to keep this bound. These numbers are listed in Table 2 used to capture the waves with a given error. For example, the first column in this table indicates that, using the fifth-order ENO instead of the second-order TVD scheme, the number of grid zones can be reduced by a factor of more than four in each space dimension, i.e., the number of grid zones in a three-dimensional calculation can be decreased by a factor 64. These results clearly indicate that if a numerical solution with a very small error should be obtained, then a high order accurate method is favorable. Especially for three-dimensional calculations, with a high-oscillating solution, this is important because the memory requirement can be drastically reduced. Of course, the computational effort increases with better accuracy. Not only differences of higher order have to be evaluated and compared but also the order of the Runge–Kutta method used for the time approximation has to be increased. If larger errors are tolerable the efficiency of the high-order accurate approximation is lower. For tolerable 10% error the difference in the number of grid

TABLE 2
Number of grid zones per wavelength needed to obtain a prescribed error limit (in per cent).

	1 %	3%	5%	10%
1^{st} order	–	–	–	77
TVD	49	25	19	14
ENO3	20	14	12	9
ENO4	14	11	9.5	7.5
ENO5	11	9	8	6.5

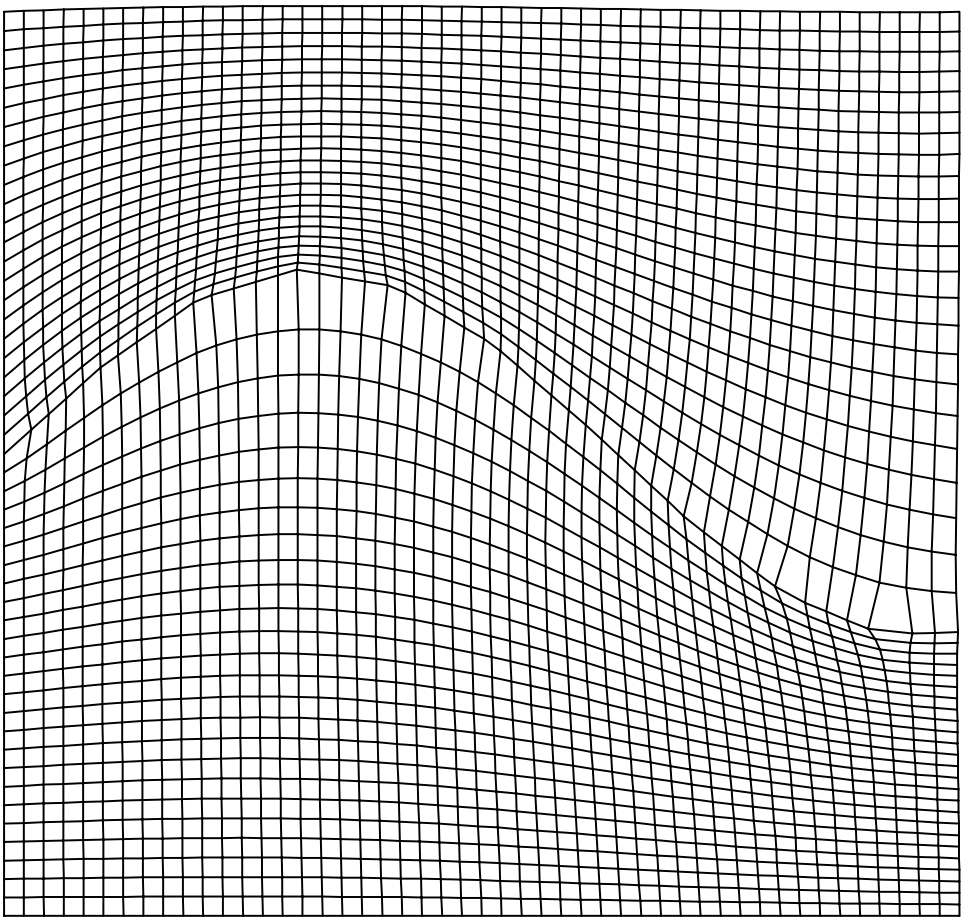


FIG. 6. *Distorted computational mesh used for testing the methods for curvilinear grids.*

zones is a factor two only. But using a first-order method at about five times more grid zones must be used as with the second-order accurate method.

The implementation for nonorthogonal grids has been tested on the grid shown in Figure 6. The computational domain $G = \{(x, y)|0 \leq x \leq 50\text{m}, 0 \leq y \leq 50\text{m}\}$ is covered by a distorted grid with large cells next to very small ones. The results for the same TM problem as sketched out above are shown in Figure 7. The comparison between the exact (upper plots) and the numerical results (lower plots) at time $t = 90\,\text{ns}$ (left) and $t = 180\,\text{ns}$ (right) (corresponding to 180 and 360 time steps, respectively)

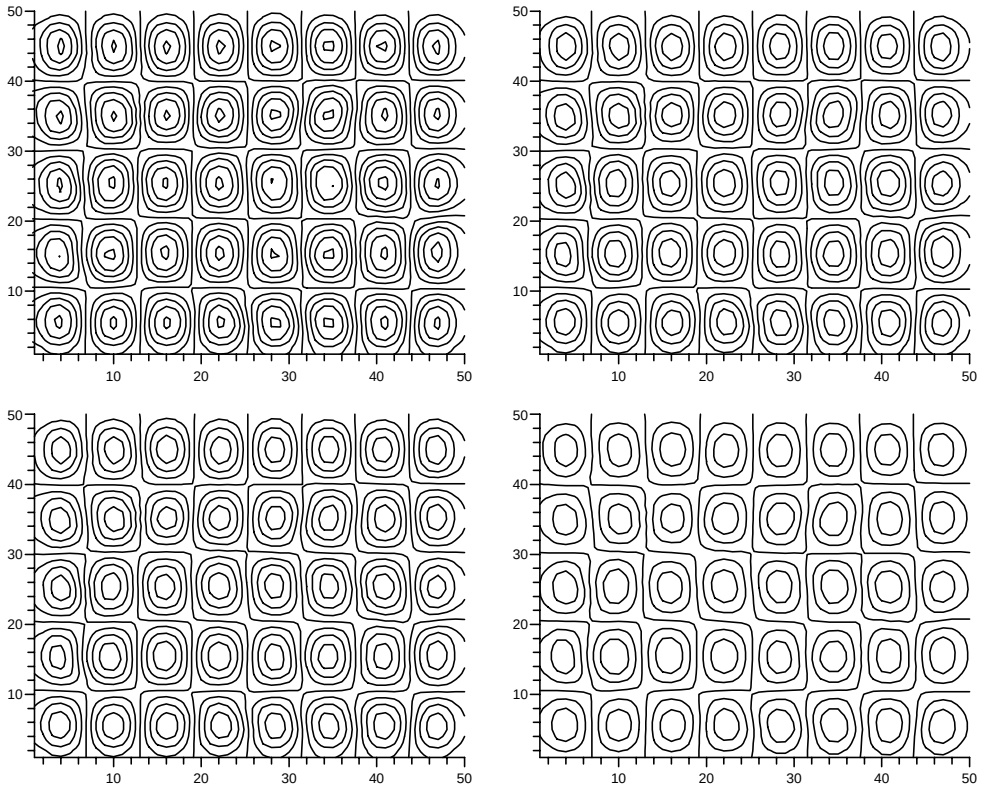


FIG. 7. *Contour plots of the exact solution (top) and the numerical results (bottom) for the TM problem with* $m = 8$ *and* $n = 5$ *at two different times,* $t = 90ns$ *on the left and* $t = 180ns$ *on the right, obtained on the grid depicted in Figure* 6 *with* 98×98 *cells.*

TABLE 3

L^2 error averaged over time, $0 \le t \le 180ns$*, for the TM problem with* $m = 8$ *and* $n = 5$ *on different levels of refinement of the distorted grid shown in Figure* 6.

	E3	B1	B2
49×49	6.72	11.80	18.40
98×98	2.10	3.79	5.78
196×196	0.70	1.32	2.00

Method of Colella [6]

	E3	B1	B2
49×49	6.89	11.97	18.88
98×98	2.21	3.92	6.09
196×196	0.75	1.37	2.13

Method of Durlofsky, Engquist, and Osher [8]

indicates that the necessary approximations of the normals at the cells edges do not at all affect the structure of the solution. The results shown in this figure are obtained using the scheme proposed by Colella [6]. When applying the scheme of Durlofsky, Engquist, and Osher [8] the results look quite similar. In order to compare the two schemes in more detail the L^2 norm of the difference between exact and numerical solution is plotted as a function of time in Figure 8. The time averaged L^2 norm, given in Table 3, leads to an experimental order of convergence (EOC) which is quite similar for both schemes and about 1.7.

Due to the clipping phenomenon of the TVD schemes the value two is not completely obtained. In essence, the approach of Colella in [6] as well as the one of Durlofsky, Engquist, and Osher in [8] for gradient approximation in boundary-fitted

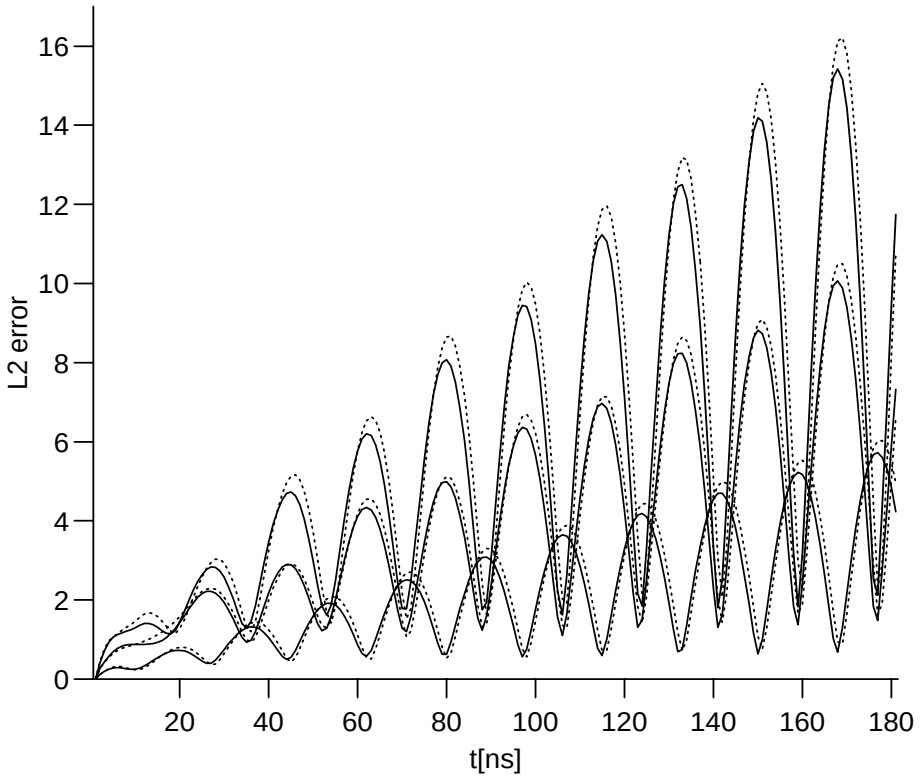


FIG. 8. *Comparison of the L^2 error of the schemes based on the slope calculation of Colella (solid line) and of Durlofsky, Engquist, and Osher (dotted line). The time-dependent errors of the B_2, B_1, and E_3 component are plotted at the top, in the middle, and at the bottom, respectively, for the TM problem ($m = 8, n = 5$) on a distorted 98×98 grid.*

coordinates are powerful and efficient and have a sufficient accuracy for practical applications.

Simultaneously checking the numerical treatment of open boundary and energy irradiation conditions, we consider the radiation field of a simple two-dimensional dipole located at the origin of the (x, y)-plane. The finite-size computational domain for this problem consisting of a quarter ring with an inner and outer radius r_i and r_o, respectively, is shown in the upper part of Figure 9. At r_i the theoretical result of the dipole field

$$\begin{aligned} B_{1,0}(t) &= \frac{E_0}{c}\frac{y}{r_i}\psi(r_i, t), \\ B_{2,0}(t) &= -\frac{E_0}{c}\frac{x}{r_i}\psi(r_i, t), \\ \psi(r_i, t) &= \sin(\omega t)J_1(kr_i) - \cos(\omega t)Y_1(kr_i) \end{aligned} \tag{4.3}$$

is irradiated, where J_1 and Y_1 are Bessel functions of first and second kind, E_0 is fixed to $-19,54V/m$, and the frequency $\omega = 0,628/ns$ and the wave number k are related according to $\omega = ck$. Open boundary conditions are imposed at r_o while symmetric boundary conditions are necessary at the axes. The contour plot seen in Figure 9 is a snapshot of the numerical B_2 field at $t = 30$ ns, clearly indicating that no visible

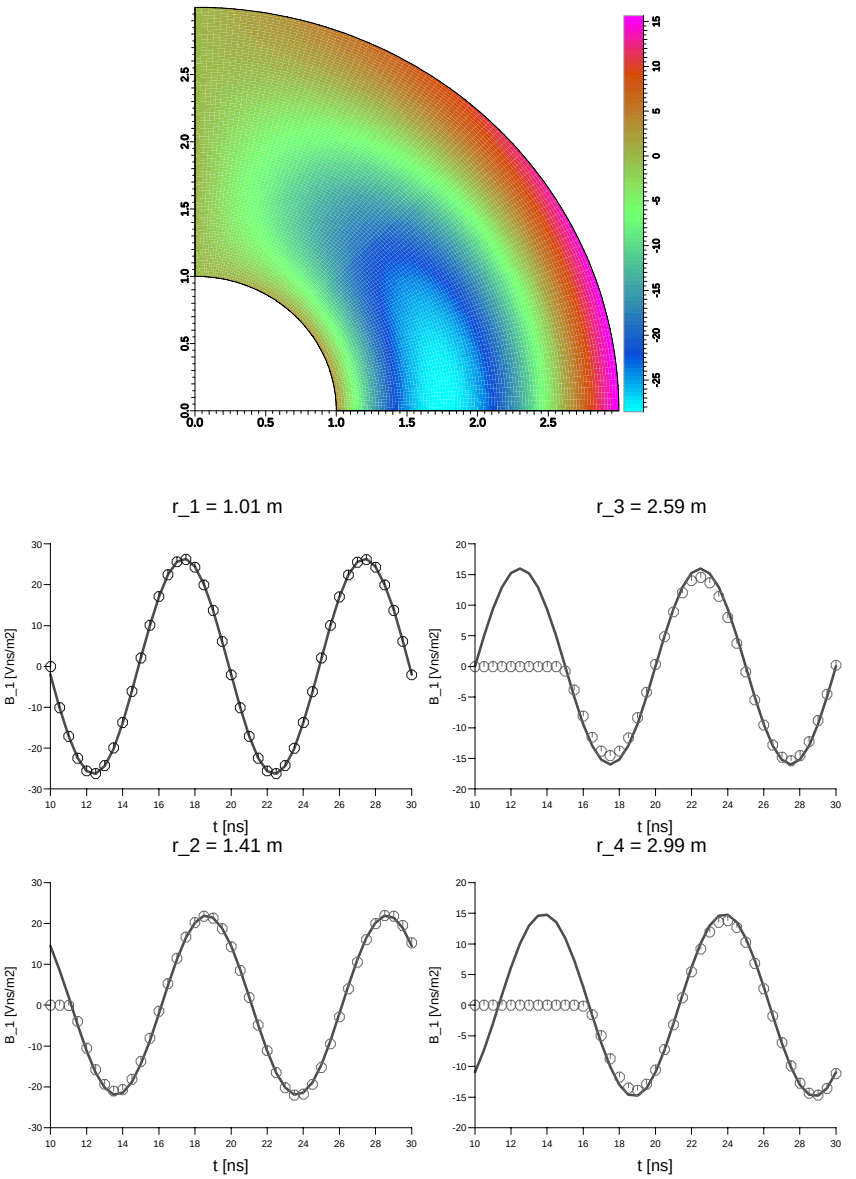


FIG. 9. *The contour plot shows the spatial distribution of the numerical and exact B_2 dipol field solution right and left of the bisector of the angle of the first quadrant, respectively, at $t = 30$ ns. A quantitative comparison between the exact solution (solid lines) and the numerical result (open circles) of the B_1 field component at four different fixed points is depicted in the lower plot.*

wave reflection occurs at the truncated numerical domain at r_o. By scanning the time variation of the B_1 component at four fixed space points ($r_1 = 1.01\,\mathrm{m}$, $r_2 = 1.41\,\mathrm{m}$, $r_3 = 2.59\,\mathrm{m}$, $r_4 = 2.99\,\mathrm{m}$) lying at the bisector of the angle of the first quadrant, the exact solution (solid lines) and numerical approximation (open circles) reveal the high quality and accuracy of the applied methods.

5. Numerical approximation of the constrained formulation. In this section we investigate a modification of the proposed finite-volume methods to enforce the divergence condition (2.1c). We restrict ourselves to describe the correction of the electric field, as the errors in charge conservation do only affect the electrical field. For the magnetic induction, where no influence of source terms occurs, the accuracy of the finite-volume method was sufficient in all our calculations.

One basic approach has been proposed by Boris [3], who applied a correction potential after the fields have been advanced. This may be formulated as a direct approximation of the constrained formulation which is fully implicit in time with respect to the Lagrange multiplier, while the other part is not changed. The implicit part can be reformulated as an elliptic equation for the potential whose numerical solution needs a lot of computational effort. This technique may also be appropriate for the correction of the magnetic field. Other correction techniques have been proposed by Marder [21] and within a finite-element framework by Assous et al. [1] and Starke [29].

In the following we propose a new technique: we approximate the hyperbolic-elliptic system (2.9a), (2.9c) by the strictly hyperbolic problem

$$\partial_t \mathbf{E} - c^2 \nabla \times \mathbf{B} + c^2 \nabla \phi = -\frac{\mathbf{j}}{\epsilon_0}, \tag{5.1a}$$

$$\frac{1}{\gamma^2} \partial_t \phi + \nabla \cdot \mathbf{E} = \frac{\rho}{\epsilon_0}, \tag{5.1b}$$

where an artificial coupling of both equations is introduced by adding the term $\partial_t \phi / \gamma^2$ to the elliptic equation (2.9c). As a hyperbolic system, this system requires initial conditions also for ϕ which are imposed setting $\phi = 0$ initially. The system (5.1) can be written together with (2.1b) as a hyperbolic evolution equation in the form (2.3) but now with the vector

$$\mathbf{u} = (E_1, E_2, E_3, B_1, B_2, B_3, \phi)^{\mathrm{T}}, \tag{5.2}$$

and modified matrices $\mathcal{K}_i, i = 1, 2, 3$. The eigenvalues of these matrices are

$$\lambda_1 = -c\gamma, \quad \lambda_{2/3} = -c, \quad \lambda_4 = 0, \quad \lambda_{5/6} = c, \quad \lambda_7 = c\gamma, \tag{5.3}$$

written in increasing order. By this, the hyperbolic-elliptic problem has been replaced by a hyperbolic system. For the defect $d = \nabla \cdot E - \rho/\epsilon_0$ a wave equation

$$d_{tt} - \gamma^2 c^2 \triangle d = (\rho_t + \nabla \cdot \mathbf{j})_t \tag{5.4}$$

with initial values

$$d(\mathbf{x}, 0) = \nabla \cdot \mathbf{E}_0(x) - \frac{\rho(\mathbf{x}, 0)}{\epsilon_0} \qquad \forall\, \mathbf{x} \in \Omega, \tag{5.5}$$

$$d_t(\mathbf{x}, 0) = \rho_t(\mathbf{x}, 0) + \nabla \cdot \mathbf{j}(\mathbf{x}, 0) \qquad \forall\, \mathbf{x} \in \Omega \tag{5.6}$$

holds. This means that the errors in Gauß's law (2.1c) do not spread out with infinite speed but with finite speed $c\gamma$, where the parameter γ is assumed to be larger than one. Here, we adopted the idea of Chorin [5] to couple the velocity equations of incompressible fluid flow with the divergence-free constraint of the velocity field which is called the method of artificial compression. The physical interpretation in this context is that the pressure waves propagate as acoustic waves with large but finite artificial speed. We remark that the artificial coupling of the Maxwell equations

and Gauß's law will give the proper stationary solution, if the method converges, because ϕ_t/γ^2 vanishes and Gauß's law is satisfied according to (5.1b). The Lagrange multiplier ϕ then equals zero.

The system (5.1) combined with the evolution equation for the magnetic field (2.1b) may be approximated by an explicit high-resolution scheme in a similar way to that proposed in section 3. But, the CFL condition now reads as

$$\gamma c \frac{\Delta t}{h} \leq 1, \tag{5.7}$$

where h denotes the smallest length of a grid zone. If γ is chosen to be much larger than one, this CFL-condition (5.7) becomes a severe restriction of the time step size. But usually the numerical charge errors are small and the transport of these errors to the boundary of the computational domain at a rate near the speed of light should be sufficient to avoid their increase in time. It is well known that in PIC calculations for relevant technical devices usually the divergence correction via a correction potential is carried out after a couple of time steps only. Nevertheless, to get the possibility of an efficient solution for larger values of γ, we split up the system (5.1), (2.1b) into the evolutionary Maxwell equations (2.3) and

$$\partial_t \mathbf{E} + c^2 \nabla \phi = 0, \tag{5.8}$$

$$\partial_t \phi + \gamma^2 \nabla \cdot \mathbf{E} = \gamma^2 \frac{\rho}{\epsilon_0}, \tag{5.9}$$

which contains all the terms involved by the Lagrange multiplier. Recasted as a system of evolution equations it has the form

$$\partial_t \mathbf{w} + \mathcal{K}_1' \partial_{x_1} \mathbf{w} + \mathcal{K}_2' \partial_{x_2} \mathbf{w} + \mathcal{K}_3' \partial_{x_3} \mathbf{w} = \mathbf{g} \tag{5.10}$$

with the vector $\mathbf{w} = (E_1, E_2, E_3, \phi)^{\mathrm{T}}$ and $\mathbf{g} = (0, 0, 0, \gamma^2 \frac{\rho}{\epsilon_0})^{\mathrm{T}}$. The matrices $\mathcal{K}_1'$ and $\mathcal{K}_2'$ have the eigenvalues

$$\lambda_1 = -c\gamma, \quad \lambda_2 = \lambda_3 = 0, \quad \lambda_4 = c\gamma. \tag{5.11}$$

After this preliminary remark, it is obvious that a high-resolution finite-volume scheme can be constructed in the same way as given in section 3.

The advantage of the splitting approach is, first, that the Maxwell solver without modifications can be applied and, second, that a subcycling for the correction step procedure can easily be introduced. According to the value of γ smaller time steps may be chosen in the correction part to satisfy (5.7). Several correction steps are performed within one sweep solving the Maxwell equations.

For numerical purposes also boundary conditions on ϕ have to be added. To have a meaningful correction technique, open boundary conditions have to be imposed on d. This is done by using the initial values $d = 0$ all outside the computational domain. This yields $\phi_t = 0$ outside the computational domain, and together with the initial condition $\phi(\mathbf{x}, 0) = 0$ resulting again in $\phi(\mathbf{x}, t) = 0$ on the boundary.

The numerical results for the following artificial but comprehensive test problem, where the initial fields at time $t = 0$ are set equal to zero in the whole computational domain

$$\mathbf{E}_0(\mathbf{x}) = 0, \qquad \mathbf{B}_0(\mathbf{x}) = 0 \qquad \forall \mathbf{x} \in \mathbf{G}, \tag{5.12}$$

demonstrate the quality of this new correction approach. The charge and current density are prescribed in such a way, that the charge conservation equation (2.2) is violated:

$$\begin{aligned} &\mathbf{j}(\mathbf{x},t)=0 \\ &\rho(\mathbf{x},t)=\rho_0 \sin(\omega t)\delta(\mathbf{x}-\mathbf{x}_0) \end{aligned} \qquad \forall(\mathbf{x},t) \tag{5.13}$$

with some constant ρ_0 and frequency ω. The exact solution of this problem is given by the electric field of a source located at $\mathbf{x}_0$ oscillating with the frequency ω. Solving this problem with a Maxwell solver based on the hyperbolic evolution equations (2.1a), (2.1b) without any divergence correction, we would obtain the trivial solution $\mathbf{E} \equiv 0, \mathbf{B} \equiv 0$ as the charge density does not enter the numerical procedure.

Numerical results for $G = \{(x,y), 0 \le x \le 50\,\mathrm{m}, 0 \le y \le 50\,\mathrm{m}\}$ are presented in Figure 10. The upper picture shows the divergence of the electric field at time $t = 5\,\mathrm{ns}$. Around $x_0 = 25\,\mathrm{m}, y_0 = 25\,\mathrm{m}$ the divergence has a peak whose exact height is 1. It is clearly visible that the divergence errors are transported to the boundary of the computational domain. A quantitative comparison between the exact and numerically obtained maximum of $\nabla \cdot \mathbf{E}$ as a function of time is seen in the lower part of Figure 10. Aside from the exact solution, results for $\gamma = 30$ and $\gamma = 20$ are shown. These results show clearly that the method gives the desired correction of the electrical field and is able to follow the oscillating of the source.

A further numerical result obtained from the area of PIC applications is depicted in Figure 11 and shows explicitly the importance of the correction of the electrical field. Here, an electron beam is emitted within a small area at the cathode ($x = 0$) and accelerated to the anode ($x = 0.1m$) under the action of a constant external field $E_1 = -2\frac{MV}{m}$. The anode-cathode gap of this simple diode configuration coincides with the computational domain discretized by a grid of 100×100 grid zones. The snapshots of the noncorrected (left) as well as the corrected (right) electron distribution and the E_1 component of the electrical filed are recorded at time $t = 10ns$, where the beam has already reached a stationary (steady) state.

In the case where no correction procedure is applied (left), the electron distribution reveals constriction and filamentation of the beam while nonphysically gradients are observed in the electrical field both as a consequence of numerical errors in charge conservation. The situation is drastically improved toward physical reliance (right) performing the proposed hyperbolic correction with $\gamma = 3$. For this electron beam example the hyperbolic correction technique is checked against the standard correction potential approach yielding comparable overall agreement. Furthermore, a gain of a factor of four in CPU time is obtained choosing the hyperbolic instead of the usual elliptic correction approach.

6. Conclusions. The finite-volume approach to solve numerically the Maxwell equations is an interesting alternative to finite-difference and finite-element schemes. With the finite-element approach it shares the flexibility of the grid so that computational domains with complicated geometries can be handled. The finite-volume schemes proposed in this paper are based on TVD and ENO-schemes for hyperbolic conservation laws and known to be very robust even in the vicinity of steep gradients. Hence, these methods are favorable for all classes of problems where fields undergo strong changes. Due to the construction principle to take into account the local wave propagation they may be considered as a natural extension of the method of characteristics.

For Cartesian grids the standard finite-difference schemes need less computational

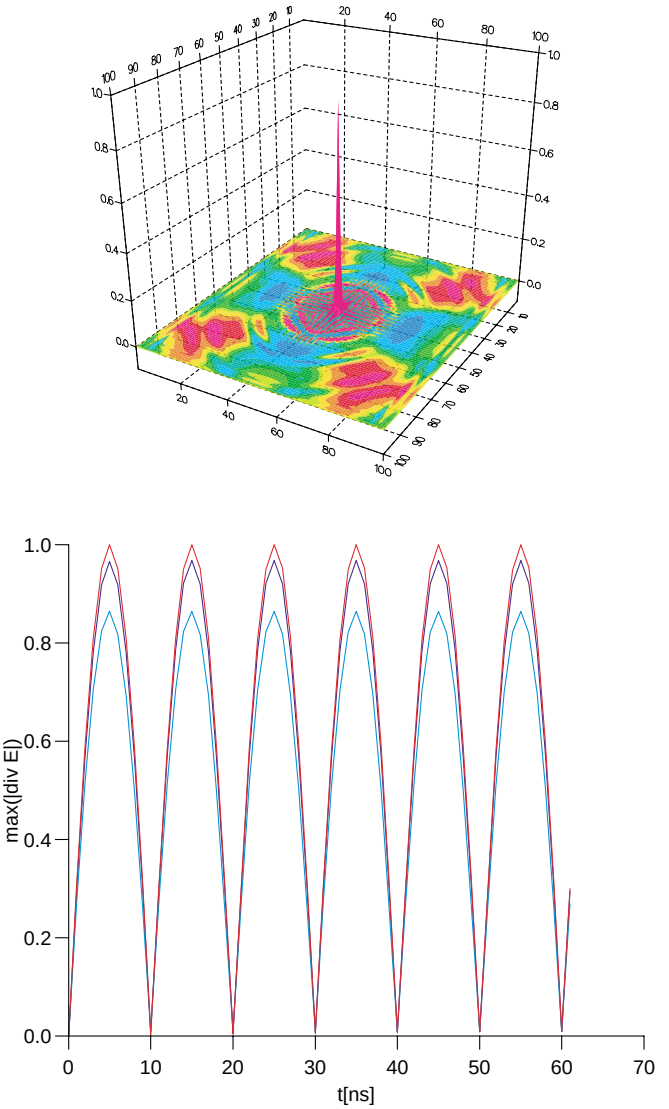


FIG. 10. *The upper picture shows the absolute value of* $\nabla \cdot \mathbf{E}$ *at* $t = 5\,ns$. *A quantitative comparison between the exact and numerically obtained maximum of* $\nabla \cdot \mathbf{E}$ *as function of time is seen in the lower part of the figure. The different curves are obtained for the values* $\gamma = 30$ (- - -), *and* $\gamma = 20$ ($\cdots$), (——) *denotes the exact solution.*

effort. For the Yee scheme [37] the effort is at about a factor of three less, compared with a second-order accurate TVD scheme. Hence, for problems with simple geometries and without strong gradients in the solution the finite-difference methods seem to be superior. Their extensions to more general grids using the "staggered" finite-volume approach [18, 19, 36] require also less computational effort than the characteristics based approach on a collocated grid. But besides their robustness a further advantage of our approach is the possibility to increase the order of accuracy without loosing their robustness by use of sophisticated ENO interpolation schemes combined with Runge–Kutta time stepping procedures. For calculations on a Carte-

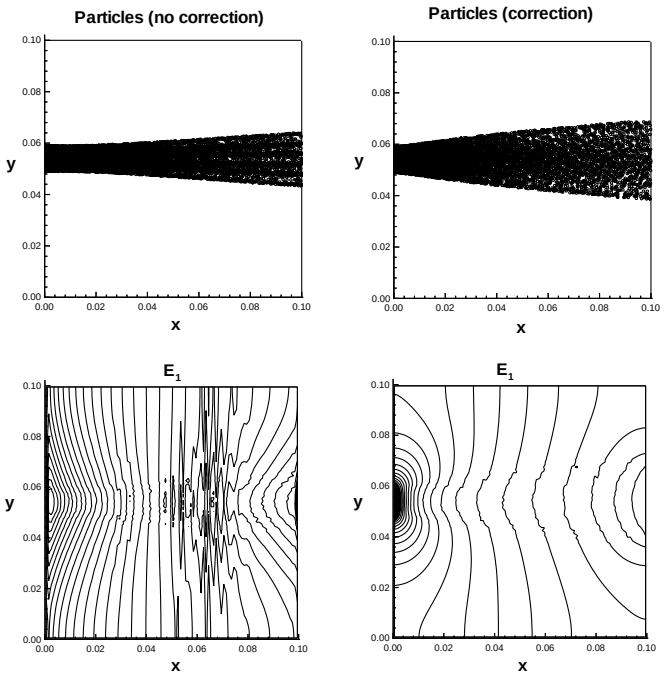


FIG. 11. *A typical finite-volume PIC application to a self-consistent beam simulation for a simple diode configuration demonstrates clearly the importance and consequence of electrical field correction. The charge conservation is enforced by the hyperbolic correction technique with* $\gamma = 3$.

sian grid we demonstrated the possibility of reducing significantly the number of grid cells by higher order accurate methods while the resolution of waves is preserved. Especially, in three-dimensional calculations this fact is interesting to save storage as well as computer time. Of course, the ENO interpolation needs more computational effort, especially for curvilinear grids, and the third- or fourth-order accurate schemes may be of practical interest only.

A class of problems where our finite-volume Maxwell solver seems to be very attractive is the self-consistent simulation of the motion of charged particles. Especially in the case where the Maxwell solver is one part of an electromagnetic PIC code, the fields have to be calculated at the particle locations. A staggering in space and time needs additional interpolation which usually increases the approximation errors. A new correction method for the electric field enforcing the validity of Gauß's law is formulated in the finite-volume framework, which seemed to be very important in the context of PIC calculations.

REFERENCES

[1] F. ASSOUS, P. DEGOND, E. HEINTZE, P.-A. RAVIART, AND J. SEGRE, *On a finite-element method for solving the three-dimensional Maxwell equations*, J. Comput. Phys., 109 (1993), pp. 222–237.

[2] C. K. BIRDSALL AND A. B. LANGDON, *Plasma Physics via Computer Simulation*, MacGraw Hill, New York, 1985.

[3] J. P. BORIS, *Relativistic plasma simulation-optimization of a hybrid code*, in Proceedings Fourth Conference on the Numerical Simulation of Plasmas, Naval Research Laboratory,

Washington, D.C., 1970, pp. 3–67.
[4] J. Cioni, L. Fezoui, and H. Steve, *A parallel time-domain Maxwell solver using upwind schemes and triangular meshes*, Impact Comput. Sci. Engrg., 5 (1993), pp. 215–247.
[5] A. J. Chorin, *A numerical method for solving incompressible flow problems*, J. Comput. Phys., 2 (1967), pp. 12–26.
[6] P. Colella, *Multidimensional upwind methods for hyperbolic conservation laws*, J. Comput. Phys., 87 (1990), pp. 171–200.
[7] R. Courant, E. Isaacson, and M. Rees, *On the solution of nonlinear hyperbolic differential equations*, Comm. Pure Appl. Math., 5 (1953), pp. 243–255.
[8] L. Durlofsky, B. Engquist, and S. Osher, *Triangle based adaptive stencils for the solution of hyperbolic conservation laws*, J. Comput. Phys., 98 (1992), pp. 64–73.
[9] J. W. Eastwood, *The virtual particle electromagnetic particle-mesh method*, Comp. Phys. Comm., 64 (1991), pp. 252–266.
[10] S. K. Godunov, *Finite difference method for numerical computation of discontinuous solutions of the equations of fluid dynamics*, Mat. Sb., 47 (1959), pp. 271–306 (in Russian).
[11] A. Harten and S. R. Chakravarthy, *Multi-Dimensional ENO Schemes for General Geometries*, Tech. Rep. 91-76, ICASE, Hampton, VA, 1991.
[12] A. Harten and S. Osher, *Uniformly high-order accurate nonoscillatory schemes* I, SIAM J. Numer. Anal., 24 (1987), pp. 279–309.
[13] G. Hedstrom, *Non-reflecting boundary conditions for nonlinear hyperbolic systems*, J. Comput. Phys., 30 (1979), pp. 222–237.
[14] R. W. Hockney and J. W. Eastwood, *Computer Simulation Using Particles*, MacGraw Hill, New York, 1981.
[15] R. Holland, *Finite-difference solution of Maxwell's equations in generalized nonorthogonal coordinates*, IEEE Trans. Nucl. Sci., NS-30 (1983), pp. 4589–4591.
[16] J. D. Jackson, *Classical Electrodynamics*, John Wiley, New York, 1975.
[17] R. J. LeVeque, *Numerical Methods for Conservation Laws*, Birkhäuser, Basel, 1990.
[18] N. K. Madsen and R. W. Ziolkowski, *Numerical solution of Maxwell's equations in the time domain using irregular nonorthogonal grids*, Wave Motion, 10 (1988), pp. 583–596.
[19] N. K. Madsen and R. W. Ziolkowski, *A three-dimensional modified finite volume technique for Maxwell's equations*, Electromagnetics, 10 (1990), pp. 147–161.
[20] G. I. Marchuk, *Methods of Numerical Mathematics*, Springer-Verlag, New York, Heidelberg, Berlin, 1975.
[21] B. Marder, *A method for incorporating Gauß's law into electromagnetic PIC codes*, J. Comput. Phys., 68 (1987), pp. 48–55.
[22] A. H. Mohammadian, V. Shankar, and W. F. Hall, *Computation of electromagnetic scattering and radiation using a time-domain finite-volume discretization procedure*, Comp. Phys. Comm., 68 (1991), pp. 175–196.
[23] C.-D. Munz, *On the numerical dissipation of high resolution schemes for hyperbolic conservation laws*, J. Comput. Phys., 77 (1988), pp. 18–39.
[24] C.-D. Munz, P. Omnes, R. Schneider, E. Sonnendrücker and U. Voß , *Divergence Correction Techniques for Maxwell Solvers Based on a Hyperbolic Model*, J. Comput. Phys., 161 (2000), to appear.
[25] P. L. Roe, *Modern shock-capturing schemes*, in Shock Waves, Proceedings of the 18th International Symposium on Shock Waves, K. Takayama, ed., Springer-Verlag, Berlin, Heidelberg, New York, 1991, pp. 29–40.
[26] J. S. Shang, *A fractional-step method for solving* 3*d time-domain Maxwell equations*, J. Comput. Phys., 118 (1995), pp. 109–119.
[27] J. S. Shang and D. Gaitonde, *Characteristic-based, time-dependent Maxwell equation solvers on a general curvilinear frame*, AIAA J., 33 (1995), pp. 491–498.
[28] C.-W. Shu and S. Osher, *Efficient implementation of essentially non-oscillatory shock capturing schemes* I, J. Comput. Phys., 83 (1988), pp. 32–78.
[29] G. Starke, *Multilevel preconditioning for the time-harmonic Maxwell equations*, in 12th Annual Review of Progress in Applied Computational Electromagnetics, Naval Postgraduate School, Monteray, CA, 1996, pp. 630–637.
[30] J. L. Steger and R. F. Warming, *Flux vector splitting of the inviscid gasdynamics equations with applications to finite-difference methods*, J. Comput. Phys., 40 (1981), pp. 263–293.
[31] G. Strang, *On the construction and comparison of difference schemes*, SIAM J. Numer. Anal., 5 (1968), pp. 506–517.
[32] A. Taflove, *Re-inventing Electromagnetics: Supercomputing Solution of Maxwell's Equations Via Direct Time Integration on Space Grids*, AIAA paper 92–0333, New York, 1992.

[33] K. W. Thompson, *Time dependent boundary conditions for hyperbolic systems*, J. Comput. Phys., 68 (1987), pp. 1–24.
[34] B. van Leer, *Towards the ultimate conservative difference scheme. A second order sequel to Godunov's method*, J. Comput. Phys., 32 (1979), pp. 101–136.
[35] H. X. Vu and J. U. Brackbill, *Accurate numerical solution of charged particle motion in a magnetic field*, J. Comput. Phys., 116 (1995), pp. 384–387.
[36] T. Weiland, *Eine Methode zur Lösung der Maxwellschen Gleichungen für sechskomponentige Felder auf diskreter Basis*, AEÜ, 14 (1977), pp. 116–119.
[37] K. S. Yee, *Numerical solution of initial boundary value problems involving Maxwell's equations in isotropic media*, IEEE Trans. Antennas and Propagation, AP-14 (1966), pp. 302–307.

SIAM J. SCI. COMPUT.
Vol. 22, No. 2, pp. 476–502
© 2000 Society for Industrial and Applied Mathematics

COMPARISON OF HIGH-ACCURACY FINITE-DIFFERENCE METHODS FOR LINEAR WAVE PROPAGATION*

DAVID W. ZINGG†

Abstract. This paper analyzes a number of high-order and optimized finite-difference methods for numerically simulating the propagation and scattering of linear waves, such as electromagnetic, acoustic, and elastic waves. The spatial operators analyzed include compact schemes, noncompact schemes, schemes on staggered grids, and schemes which are optimized to produce specific characteristics. The time-marching methods include Runge–Kutta methods, Adams–Bashforth methods, and the leapfrog method. In addition, the following fully-discrete finite-difference methods are studied: a one-step implicit scheme with a three-point spatial stencil, a one-step explicit scheme with a five-point spatial stencil, and a two-step explicit scheme with a five-point spatial stencil. For each method, the number of grid points per wavelength required for accurate simulation of wave propagation over large distances is presented. The results provide a clear understanding of the relative merits of the methods compared, especially the trade-offs associated with the use of optimized methods. A numerical example is given which shows that the benefits of an optimized scheme can be small if the waveform has broad spectral content.

Key words. finite-difference methods, wave propagation, electromagnetics, acoustics, Maxwell equations

AMS subject classifications. 65M05, 78A40, 78M20, 76Q05

PII. S1064827599350320

Introduction. Numerical simulation can play an important role in the context of engineering design and in improving our understanding of complex systems. The simulation of wave phenomena, including electromagnetic, elastic, and acoustic waves, is an area of active research. Lighthill [24] and Taflove [40] discuss the prospects for computational aeroacoustics and electromagnetics, respectively. The computational requirements for accurate simulations of the propagation and scattering of waves can be high, particularly if the size of the geometry under study is much larger than the wavelength. Consequently, there has been considerable recent effort directed towards improving the efficiency of numerical methods for simulating wave phenomena.

In electromagnetics, the most popular approach to the numerical solution of the time-domain Maxwell equations for numerous applications has been the algorithm of Yee [50], which was named the finite-difference time-domain (FDTD) method by Taflove [41]. This algorithm combines second-order centered differences on a staggered grid in space with second-order staggered leapfrog time marching. Its main attributes are its very low cost per grid node and lack of dissipative error. Yee's method is often applied using Cartesian grids, with a special treatment of curved boundaries [20]. Extension to curvilinear grids was carried out by Fusco [11]. Madsen and Ziolowski [28] put the method into a finite-volume framework applicable to unstructured grids. Vinokur and Yarrow [47, 48] developed a related finite-surface method with advantages at boundaries and grid singularities.

Other methods which have been successfully applied to the time-domain Maxwell equations include the upwind Lax–Wendroff approach used by Shankar, Mohamma-

*Received by the editors January 28, 1999; accepted for publication(in revised form) December 10, 1999; published electronically July 13, 2000.
http://www.siam.org/journals/sisc/22-2/35032.html

†University of Toronto Institute for Aerospace Studies, 4925 Dufferin St., Downsview, Ontario, Canada M3H 5T6 (dwz@oddjob.utias.utoronto.ca).

dian, and Hall [38], the characteristic-based fractional-step method of Shang [36], and the finite-element method of Cangellaris, Lin, and Mei [5]. Although all of these second-order methods have been used with a great deal of success, they are efficient only for geometries of moderate electrical size, on the order of 20 wavelengths or less. For wave propagation over longer distances, the grid resolution requirements of second-order methods can become excessive, leading to impractical CPU and memory requirements. This has motivated the development of higher-order methods which produce smaller errors for a given grid resolution, such as the extension of Yee's method to fourth-order in space [40] and the methods of Liu [25], Shang and Gaitonde [37], Zingg et al. [52, 55, 18, 19], Petropoulos [31], Turkel and Yefet [44], and Young, Gaitonde, and Shang [51], many of which use staggered grids.

In seismology, the need for higher-order methods has been recognized for some time. Alford, Kelly, and Boore [2], Marfurt [29], Dablain [9], and Sei [34] present higher-order algorithms for the elastic wave equation. Similarly, higher-order finite-difference methods have been developed for acoustic applications by Gottlieb and Turkel [13], Cohen and Joly [8], and Davis [10], for example.

It can be advantageous to modify the coefficients of a potentially higher-order method, thereby lowering the order of accuracy, to produce reduced errors over a range of wavenumbers. We refer to such schemes as optimized schemes [52]. This approach was first proposed by Vichnevetsky and De Schutter [45] and later studied in more detail by Holberg [16] and Lele [23]. Haras and Ta'asan [15] and later Kim and Lee [21] further refined the optimization technique used by Lele. The papers by Holberg and Lele spawned a number of optimized schemes, including those presented by Hu, Hussaini, and Manthey [17], Lockard, Brentner, and Atkins [27], Sguazzero, Kindelan, and Kamel [35], Tam [42], Tam and Webb [43], Zingg and Lomax [53], and Zingg, Lomax, and Jurgens [52, 55]. See Wells and Renaut [49] for a discussion of several of these schemes in the context of computational aeroacoustics.

In addition to the accuracy of the interior differencing scheme, there are several other important issues in simulating wave phenomena. High-order and optimized methods often have a large spatial stencil which cannot be used near boundaries. This necessitates the use of numerical boundary schemes which must be suitably accurate relative to the interior scheme [14] and stable. These can be difficult to obtain, and this represents a significant obstacle to the use of higher-order methods. Recent progress is reported by Olsson [30] and Carpenter, Gottlieb, and Abarbanel [6, 7]. Zingg and Lomax [54] present a fifth-order numerical boundary scheme which produces a stable scheme in conjunction with sixth-order centered differences. An improved version, which is stable for curvilinear grids, is given by Jurgens and Zingg [19]. Another important consideration is the boundary condition at the outer boundary of the domain, which inevitably causes spurious reflection. As a result of recent developments in this area, such as perfectly matched absorbing layers [1, 4, 12, 32, 57], such reflections can be substantially reduced. Consequently, the numerical errors introduced by the interior differencing scheme are often dominant, and the choice of the interior scheme is critical to the efficient simulation of wave phenomena.

An efficient discretization of a partial differential equation governing a linear wave phenomenon reliably maintains the numerical error below an acceptable threshold, which is problem dependent, at the lowest possible cost. The cost includes processing time and memory, although development cost can be considered as well. Numerical errors arise from both the spatial and the temporal discretization. They include both phase and amplitude errors, which depend on the wavenumber, the grid spacing,

the Courant number, and the direction of propagation relative to the grid. The dependence of the phase speed on the wavenumber results in numerical dispersion [46], while the amplitude error results in numerical dissipation. The dependence of these errors on the direction of wave propagation relative to the grid leads to numerical anisotropy. Fourier analysis provides a straightforward means of calculating these errors [46, 26]. Although this simplified analysis excludes errors associated with nonuniform grids and boundaries, it is a very useful tool for scheme evaluation and development. Good performance under the conditions of Fourier analysis, i.e., uniform grids and periodic boundary conditions, is a necessary condition for good performance under more general conditions.

Through Fourier analysis, one can calculate the phase and amplitude error of a given method as a function of the wavenumber. However, this information can be difficult to interpret. A more useful approach used by Lockard, Brentner, and Atkins [27] is to plot the grid resolution in terms of grid points per wavelength (PPW) required to maintain global phase and amplitude errors below a specified threshold as a function of the distance of propagation expressed in terms of the number of wavelengths traveled. This is not only an excellent metric for scheme development and comparison, but it also provides useful information for applying the methods to specific problems.

In this paper, we study and compare the grid resolution requirements of several high-order and optimized schemes, including spatial discretizations, time-marching methods, and combined time-space discretizations. Emphasis is on schemes requiring under 30 PPW for accurate simulations with propagation distances of up to 200 wavelengths. The purpose is twofold. First, the results can aid in the evaluation and application of high-accuracy methods and are especially helpful in clarifying the behavior of optimized schemes. Second, the framework presented can be used in the assessment of new methods.

Finite-difference schemes compared. In this section, we present the various finite-difference schemes in the context of the linear advection equation given by

$$\frac{\partial u}{\partial t} + a\frac{\partial u}{\partial x} = 0, \tag{1}$$

where u is a scalar quantity propagating with speed a, which is real and positive. The schemes under study can be divided into two distinct groups. In the first group, the spatial and temporal discretizations are independent, i.e., a discretization is applied to the spatial derivative to produce a system of ordinary differential equations (ODEs), which is solved numerically using a time-marching method. Hence we first study spatial discretizations and time-marching methods separately and later consider specific combinations. The spatial difference operators are approximations to $\partial/\partial x$. The time-marching methods are presented as applied to a scalar ODE of the form

$$\frac{du}{dt} = f(u,t). \tag{2}$$

The second group of methods involves the simultaneous discretization of space and time; there is no intermediate semidiscrete form. These are presented as applied to (1) itself. The coefficients of all of the schemes studied are given in the appendix.

The methods can also be classified as dissipative or nondissipative. A spatial discretization is nondissipative, i.e., produces no amplitude error, if it is skew symmetric. When combined with certain time-marching methods, such as the leapfrog

method, the resulting full discretization is nondissipative, that is, the amplitude of the solution will neither grow nor decay except in the presence of boundaries. Combined space-time discretizations can be nondissipative as well. This is generally a useful property in that it is consistent with the physics in preserving certain energy norms. Thus nondissipative schemes are widely used with great success, especially in solving the Maxwell equations. However, dissipative schemes can be effective as well, as long as the numerical dissipation is carefully controlled. For example, in the scheme of Zingg, Lomax, and Jurgens [55], the dissipative error is smaller than the phase error for all wavenumbers. Therefore, any mode for which the numerical dissipation is excessive already suffers from excessive phase error. Dissipative schemes have the advantage that high-frequency spurious waves, arising from boundaries, for example, are damped [13]. Furthermore, dissipative schemes can, at least in principle, be applied to nonlinear problems.

Spatial difference operators. On a uniform grid with $x_j = j\Delta x$ and $u_j = u(x_j)$, compact centered-difference schemes of up to tenth order can be represented by the following formula:

$$\begin{aligned} &\beta(\delta_x u)_{j-2} + \alpha(\delta_x u)_{j-1} + (\delta_x u)_j + \alpha(\delta_x u)_{j+1} + \beta(\delta_x u)_{j+2} \\ &= \frac{1}{2\Delta x}\left[\frac{c}{3}(u_{j+3} - u_{j-3}) + \frac{b}{2}(u_{j+2} - u_{j-2}) + a(u_{j+1} - u_{j-1})\right], \end{aligned} \tag{3}$$

where $(\delta_x u)_j$ is an approximation to $\partial u/\partial x$ at node j. These schemes produce no amplitude error. Noncompact schemes of up to sixth order are obtained with $\beta = \alpha = 0$.

Haras and Ta'asan [15] revisited the compact schemes of Lele [23] based on (3) using a more sophisticated optimization procedure. They developed tridiagonal schemes ($\beta = 0$) and tridiagonal schemes with a five-point stencil ($\beta = c = 0$) as well. In each case, Haras and Ta'asan present several methods which result from different parameters in the optimization procedure. In the comparisons below, we will include only those schemes which are best according to our criterion, i.e., those schemes which require the fewest grid points per wavelength for accurate wave propagation over a distance of 200 wavelengths.

The spatial operator of Zingg, Lomax, and Jurgens [52, 55] is noncompact with a seven-point stencil. The operator is divided into a skew-symmetric part given by

$$(\delta_x^a u)_j = \frac{1}{\Delta x}\left[a_3\left(u_{j+3} - u_{j-3}\right) + a_2\left(u_{j+2} - u_{j-2}\right) + a_1\left(u_{j+1} - u_{j-1}\right)\right] \tag{4}$$

and a symmetric part given by

$$(\delta_x^s u)_j = \frac{1}{\Delta x}\left[d_3\left(u_{j+3} + u_{j-3}\right) + d_2\left(u_{j+2} + u_{j-2}\right) + d_1\left(u_{j+1} + u_{j-1}\right) + d_0 u_j\right]. \tag{5}$$

The symmetric part provides dissipation of spurious high-wavenumber components of the solution. The magnitude of the dissipative component is chosen such that the amplitude error produced is less than the phase error. [52, 55] include a maximum-order scheme and an optimized scheme. The optimized scheme was designed to minimize the maximum phase and amplitude errors for waves resolved with at least 10 PPW. It is superior for distances of travel of up to 330 wavelengths. Hence we do not consider the maximum-order scheme here. Tam and Webb [43] have also developed an optimized scheme based on the skew-symmetric operator given in (4).

Lockard, Brentner, and Atkins [27] present an optimized upwind-biased noncompact spatial difference operator based on an eight-point stencil. It can be written in the form

$$(\delta_x u)_j = \frac{1}{\Delta x} \sum_{m=-4}^{3} a_m u_{j+m}. \tag{6}$$

The operator is optimized for waves resolved with at least 7 PPW and provides excellent accuracy up to about 340 wavelengths of travel. Note that this operator can also be written as the sum of a skew-symmetric and a symmetric operator. A nine-point stencil is required for systems of equations with wave speeds of opposite sign.

Differencing schemes on staggered grids are well suited to problems in which the time derivative of one variable depends on the spatial derivative of the other, and vice-versa. This is the case for the time-domain Maxwell equations or the Euler equations linearized about a reference state with zero velocity, for example. Centered staggered differencing schemes of up to sixth order can be written in the form

$$(\delta_x u)_j = \frac{1}{\Delta x} \left[b_1 \left(u_{j+1/2} - u_{j-1/2} \right) + b_2 \left(u_{j+3/2} - u_{j-3/2} \right) + b_3 \left(u_{j+5/2} - u_{j-5/2} \right) \right]. \tag{7}$$

Time-marching methods. There are many considerations involved in selecting a time-marching method, including efficiency, i.e., accuracy per unit computational effort, stability, and memory use. Since wave propagation problems are generally not particularly stiff, explicit methods are appropriate. Thus the Adams–Bashforth and Runge–Kutta families are suitable candidates. Although other methods, such as predictor-corrector methods, can be used, we analyze only a few popular options, as well as some recently introduced methods.

Since a large portion of the computational effort is generally associated with evaluation of the derivative function, one can approximately assess the efficiency of a time-marching method by accounting for the number of derivative function evaluations per time step. Runge–Kutta methods require one derivative function evaluation per stage. Zingg and Chisholm [56] have shown that for linear ODEs with constant coefficients, Runge–Kutta methods of up to sixth order can be derived with a number of stages equal to the order. However, the memory requirements of the methods of orders five and six are high. For nonlinear ODEs and linear ODEs with nonconstant coefficients, six stages are required for fifth-order accuracy and seven stages for sixth-order accuracy [22].

An alternative approach is to use low-storage multistage methods which are high-order for homogeneous linear ODEs but second-order otherwise [55]. Haras and Ta'asan [15] and Zingg, Lomax, and Jurgens [52, 55] present five- and six-stage methods of this type, respectively. For example, when applied to (2), the six-stage method of Zingg, Lomax, and Jurgens [52, 55] is given by

$$\begin{aligned}
u^{(1)}_{n+\alpha_1} &= u_n + h\alpha_1 f_n, \\
u^{(2)}_{n+\alpha_2} &= u_n + h\alpha_2 f^{(1)}_{n+\alpha_1}, \\
u^{(3)}_{n+\alpha_3} &= u_n + h\alpha_3 f^{(2)}_{n+\alpha_2}, \\
u^{(4)}_{n+\alpha_4} &= u_n + h\alpha_4 f^{(3)}_{n+\alpha_3},
\end{aligned} \tag{8}$$

$$u_{n+\alpha_5}^{(5)} = u_n + h\alpha_5 f_{n+\alpha_4}^{(4)},$$
$$u_{n+1} = u_n + h f_{n+\alpha_5}^{(5)},$$

where $h = \Delta t$ is the time step, $t_n = nh$, $u_n = u(t_n)$, and

$$f_{n+\alpha}^{(k)} = f(u_{n+\alpha}^{(k)}, t_n + \alpha h).$$

The five-stage method of [15] is analogous. In their maximum-order form (for homogeneous ODEs), these methods are not stable for pure central (skew-symmetric) differencing in space. Consequently, Haras and Ta'asan modify the coefficients of the five-stage scheme to produce stability, while Zingg, Lomax, and Jurgens [52, 55] add a dissipative (symmetric) component to their spatial operator, as given in (5).

Adams–Bashforth methods require only one derivative function evaluation per time step, and thus can be more efficient than Runge–Kutta methods of equal order. Since they are *multistep* methods, however, they require extra storage. Furthermore, Adams–Bashforth methods of order higher than four have extremely restrictive stability bounds. Thus the time-marching methods we consider here are the fourth-order Adams–Bashforth method, the low-storage fourth-order Runge–Kutta method for linear ODEs given by Zingg and Chisholm [56], and the low-storage five- and six-stage methods described above. In terms of stability and Fourier error analysis, the fourth-order Runge–Kutta method of [56] is identical to the classical fourth-order Runge–Kutta method. With respect to memory requirements, the fourth-order Adams–Bashforth method requires five memory locations per dependent variable, while the other methods considered require only two.

A natural choice of time-marching method for use with staggered spatial differencing is the second-order staggered leapfrog method, given by

$$u_{n+1} = u_n + h f_{n+1/2}. \tag{9}$$

When used with a nondissipative spatial scheme, as in the FDTD method, the resulting fully-discrete operator produces no amplitude error. Furthermore, the staggered leapfrog method generally produces a leading phase error, which can offset the phase lag usually produced by centered spatial differences. At specific Courant numbers and angles of propagation, the perfect-shift property[1] can be obtained, leading to exact propagation for all wavenumbers. Although this has little practical significance, fully-discrete methods which possess this property are generally most accurate when used near the perfect-shift conditions.

Simultaneous space-time discretizations. The scheme of Davis [10] is a nondissipative implicit scheme with a three-point spatial stencil. When applied to the linear advection equation, it is given by

$$a_0 u_j^{n+1} + a_1 u_{j-1}^{n+1} + a_2 u_{j+1}^{n+1} = b_0 u_j^n + b_1 u_{j-1}^n + b_2 u_{j+1}^n \tag{10}$$

with $a_0 = b_0$, $a_1 = b_2$, and $a_2 = b_1$, where $u_j^n = u(x_j, t_n)$. Being implicit, this scheme requires more computational expense than explicit schemes. However, it achieves fourth-order accuracy in time and space with a three-point spatial stencil.

The scheme of Gottlieb and Turkel [13] is an extension of the Lax–Wendroff approach to fourth-order in space, remaining second-order in time. It is an explicit

[1] This refers to the situation when the error from the spatial discretization precisely cancels that from the temporal discretization.

one-step scheme but requires a five-point stencil. When applied to the linear advection equation, it is given by

$$u_j^{(1)} = u_j^n + \frac{ah}{6\Delta x}(u_{j+2}^n - 8u_{j+1}^n + 7u_j^n),$$
$$u_j^{n+1} = \frac{1}{2}(u_j^n + u_j^{(1)}) + \frac{ah}{12\Delta x}(-u_{j-2}^{(1)} + 8u_{j-1}^{(1)} - 7u_j^{(1)}). \tag{11}$$

The two-stage form is motivated by nonlinear problems. This method is dissipative and has received considerable use in nonlinear problems. The justification for having a lower order of accuracy in time than in space is based on the idea that the system is at least mildly stiff. Hence the time-step restriction based on the parasitic modes, i.e., the stability limit, is often much more stringent than that for the driving modes, i.e., the accuracy limit. Note, however, that many wave propagation problems are not stiff at all.

It is straightforward to extend this approach to fourth-order in time without increasing the stencil. For example, one can derive the following dissipative one-step fourth-order method:

$$u_j^{n+1} = u_j^n - ah(\delta_x^{(4)}u)_j^n + \frac{h^2a^2}{2}(\delta_{xx}^{(4)}u)_j^n - \frac{h^3a^3}{6}(\delta_{xxx}^{(2)}u)_j^n + \frac{h^4a^4}{24}(\delta_{xxxx}^{(2)}u)_j^n, \tag{12}$$

where $\delta_x^{(4)}$ is a fourth-order centered difference approximation to a first derivative, $\delta_{xx}^{(4)}$ is a fourth-order centered difference approximation to a second derivative, $\delta_{xxx}^{(2)}$ is a second-order centered difference approximation to a third derivative, and $\delta_{xxxx}^{(2)}$ is a second-order centered approximation to a fourth derivative. All of the operators on the right-hand side require a five-point stencil, as follows:

$$(\delta_x^{(4)}u)_j = \frac{1}{12\Delta x}\left(u_{j-2} - 8u_{j-1} + 8u_{j+1} - u_{j+2}\right),$$
$$(\delta_{xx}^{(4)}u)_j = \frac{1}{12\Delta x^2}\left(-u_{j-2} + 16u_{j-1} - 30u_j + 16u_{j+1} - u_{j+2}\right),$$
$$(\delta_{xxx}^{(2)}u)_j = \frac{1}{2\Delta x^3}\left(-u_{j-2} + 2u_{j-1} - 2u_{j+1} + u_{j+2}\right),$$
$$(\delta_{xxxx}^{(2)}u)_j = \frac{1}{\Delta x^4}\left(u_{j-2} - 4u_{j-1} + 6u_j - 4u_{j+1} + u_{j+2}\right). \tag{13}$$

This scheme is quite complicated to apply in multidimensions. Hence we will consider a simpler scheme which is fourth-order in time and space, the following nondissipative two-step explicit scheme:

$$u_j^{n+1} = u_j^{n-1} - 2ah(\delta_x^{(4)}u)_j^n - \frac{h^3a^3}{3}(\delta_{xxx}^{(2)}u)_j^n, \tag{14}$$

where $\delta_x^{(4)}$ and $\delta_{xxx}^{(2)}$ are given in (13) above. Related methods for the second-order wave equation are presented by Cohen and Joly [8] and Shubin and Bell [39].

Fourier error analysis. Consider the linear convection equation, (1), on an infinite domain. A solution initiated by a harmonic function with wavenumber κ is

$$u(x,t) = f(t)e^{i\kappa x}, \tag{15}$$

where $f(t)$ satisfies the ODE

$$\frac{df}{dt} = -ia\kappa f. \tag{16}$$

If the spatial derivative is approximated by a finite-difference formula, then the ODE becomes

$$\frac{df}{dt} = -ia\kappa^* f, \tag{17}$$

where κ^* is the modified wavenumber. For example, the modified wavenumber associated with the spatial operator given in (3) is obtained from

$$\kappa^* \Delta x = \frac{a\sin(z) + (b/2)\sin(2z) + (c/3)\sin(3z)}{1 + 2\alpha\cos(z) + 2\beta\cos(2z)}, \tag{18}$$

where $z = \kappa\Delta x$, i.e., $z = 2\pi/\text{PPW}$. The numerical phase speed, a^*, is the speed at which a harmonic function propagates numerically. It is related to the modified wavenumber by

$$\frac{a^*}{a} = \frac{\kappa^*}{\kappa}. \tag{19}$$

The dependence of a^* on κ introduces numerical dispersion. For the spatial operator given in (4) and (5), the modified wavenumber is obtained from

$$\begin{aligned} i\kappa^*\Delta x = {} & d_0 + 2(d_1\cos\kappa\Delta x + d_2\cos 2\kappa\Delta x + d_3\cos 3\kappa\Delta x) \\ & + 2i(a_1\sin\kappa\Delta x + a_2\sin 2\kappa\Delta x + a_3\sin 3\kappa\Delta x). \end{aligned} \tag{20}$$

The real part of the modified wavenumber determines the phase (or dispersive) error, while the imaginary part determines the amplitude (or dissipative) error.

The accuracy of a time-marching method can be assessed through the characteristic polynomial of the ODE resulting from application of the time-marching method to the following ODE:

$$\frac{df}{dt} = \lambda f. \tag{21}$$

The accuracy analysis is based on the principal root of the characteristic polynomial, denoted by $\sigma(\lambda h)$, which is an approximation to $e^{\lambda h}$. In order to assess the accuracy of time-marching methods for wave propagation, we consider only pure imaginary values of λ, i.e., $\lambda = i\omega$ with ω real. The normalized local amplitude and phase errors are determined from $\sigma(\lambda h)$, as follows:

$$er_a = |\sigma| - 1, \tag{22}$$

$$er_p = -\frac{\phi}{\omega h} + 1, \tag{23}$$

where $\phi = \arctan(\sigma_i/\sigma_r)$, and σ_r and σ_i are the real and imaginary parts of σ. Equation (16) is in the form of (21) with $\lambda = i\omega = -ia\kappa$. Substituting $\omega = -a\kappa$ into (23), we obtain

$$er_p = \frac{\phi}{zC} + 1, \tag{24}$$

where $\sigma = \sigma(-izC)$, and $C = ah/\Delta x$ is the Courant number. To analyze a time-marching method in combination with a spatial discretization, $\sigma = \sigma(-i\kappa^*\Delta x C)$,

where κ^* is the modified wavenumber associated with the spatial operator. For simultaneous space-time discretizations, the complex amplification factor is calculated directly by substituting a solution of the form $u(x,t) = \sigma^n e^{i\kappa x}$ into the fully discrete operator [3].

Our criterion for comparing schemes is based on the magnitude of the global amplitude and phase errors, which are

$$\begin{aligned} Er_a &= \left||\sigma|^N - 1\right| \\ &= \left||\sigma|^{PPWn_w/C} - 1\right|, \end{aligned} \tag{25}$$

$$\begin{aligned} Er_p &= N|\omega h - \phi| \\ &= n_w \left|\frac{PPW\phi}{C} + 2\pi\right|, \end{aligned} \tag{26}$$

where $N = PPWn_w/C$ is the number of time steps and n_w is the number of wavelengths traveled. Using these formulas with an accurate time-marching method and a very small Courant number gives the errors for the spatial operator alone; the time advance is effectively exact. In the results section, the various methods are compared in terms of the PPW required to keep both global phase and amplitude errors below 0.1 as a function of the number of wavelengths traveled. This error threshold is, of course, arbitrary and corresponds to that used by Lockard, Brentner, and Atkins [27]. The relative performance of the methods is similar for other choices, and thus our conclusions are independent of the specific value of the error threshold. For some schemes, especially optimized schemes, the error does not increase monotonically with the nondimensional wavenumber z. Hence the error can actually become larger as the grid resolution is increased. In such cases, we use the value of PPW for which all higher values of PPW also satisfy the error threshold.

Since practical problems involve multidimensional systems of equations, the use of one-dimensional scalar analysis requires some justification. A hyperbolic system of equations is diagonalizable with real eigenvalues, or wave speeds. These wave speeds can vary significantly. Consequently, the effective Courant number for the different waves can also vary significantly. A nonuniform grid further contributes to the variation in the Courant number. For the fastest wave, the Courant number must be sufficiently small such that the scheme is both accurate and stable. For the slower waves, the Courant number is then much smaller. A scheme which produces poor accuracy at low Courant numbers is thus inappropriate for such problems. This situation can arise if a scheme relies on cancellation of time and space errors to achieve high accuracy. For example, several schemes which combine the spatial and temporal discretization produce the perfect-shift property at specific Courant numbers. Often this perfect cancellation of temporal and spatial errors is achieved at a Courant number of unity. For such schemes, the error *increases* as the Courant number is reduced since the temporal error decreases and no longer cancels the spatial error. Hence it is important to assess schemes over a range of Courant numbers. Note that this is not an issue with schemes which combine a high-accuracy spatial discretization with a high-accuracy time-marching method. Since such schemes generally do not rely on cancellation to achieve high accuracy, the error does not increase as the Courant number is reduced.

The situation is similar in multidimensions. For most spatial discretizations, the error is largest for waves propagating at 0 or 90 degrees to the grid [46, 23, 53]. The

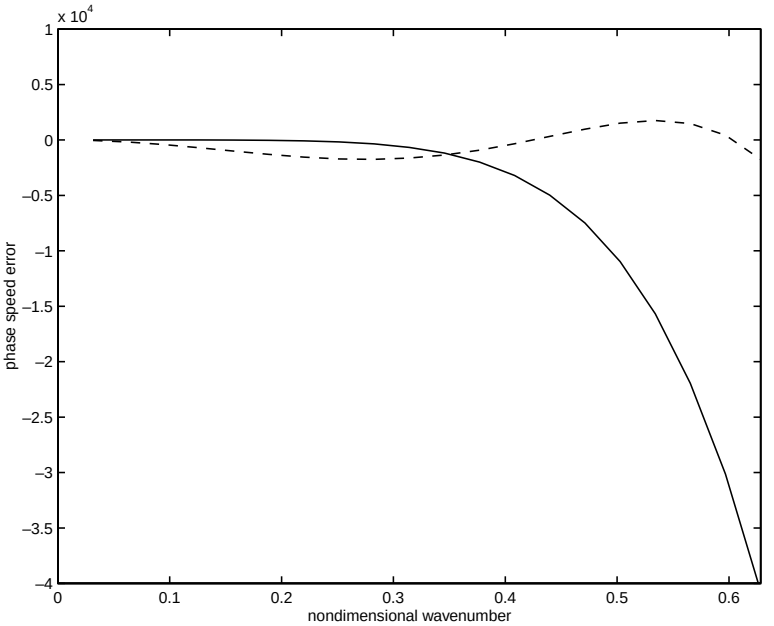


FIG. 1. *Phase speed error for the maximum-order (—) and optimized (- - -) schemes of Zingg, Lomax, and Jurgens* [55].

error from the time-marching method is isotropic. Therefore, if there is no cancellation of spatial and temporal errors, the one-dimensional analysis is conservative. In contrast, if a scheme relies on such cancellation, then the error at arbitrary angles of propagation can be much higher than that in one dimension. For example, a scheme with the perfect shift property at a Courant number of unity in one dimension can produce large errors at the same Courant number in multidimensions and can even be unstable. This occurs because the spatial error is anisotropic while the temporal error is isotropic. Hence they cannot cancel at all angles. Fortunately, this situation can be revealed by a one-dimensional analysis as long as a wide range of Courant numbers is considered. As the Courant number tends to zero, so does the temporal error, and only the spatial error remains. Hence the one-dimensional analysis at a low Courant number can represent the worst case for some schemes, since there is no cancellation of temporal and spatial errors. This is confirmed by the results presented below.

Before proceeding to the results, we briefly discuss the concept of an optimized scheme, using the schemes of Zingg, Lomax, and Jurgens [55] as an example. Figure 1 shows the phase speed error for the maximum-order and optimized spatial discretizations given in [55] for a nondimensional wavenumber z between 0 and $\pi/5$, i.e., for waves resolved with at least 10 PPW. This optimized scheme trades increased errors at low wavenumbers in return for greatly reduced errors for $0.4 \leq z \leq \pi/5$. The idea is to extend the range of wavenumbers for which the scheme is sufficiently accurate. Note that, for the optimized scheme, the error at $z = 0.29$ (22 PPW) is very close to that at $z = \pi/5$ (10 PPW). Consequently, there is no benefit in increasing the grid resolution from 10 PPW to 22 PPW. If a grid resolution of 10 PPW is insufficient for a given propagation distance, then greater than 22 PPW will be required. This explains the sudden jumps in the grid resolution requirements of optimized schemes which will be seen below.

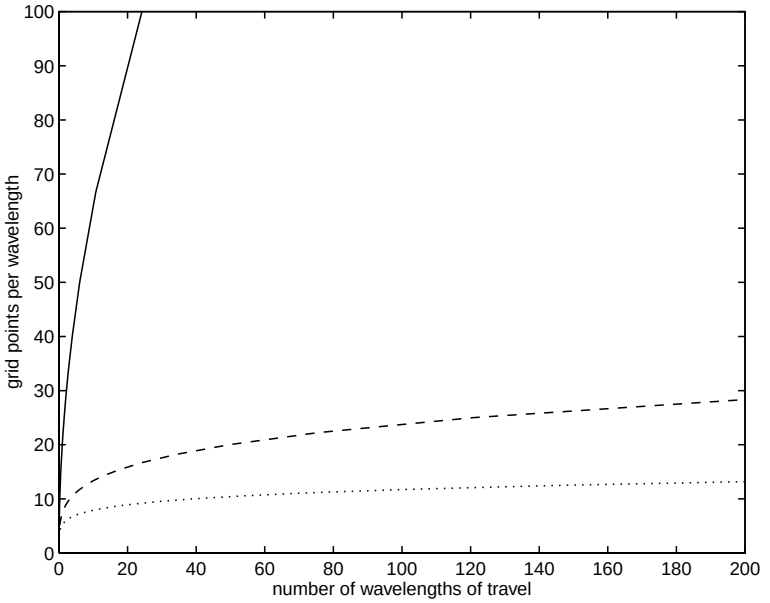


FIG. 2. *Grid resolution requirements for second-order (—), fourth-order (- - -), and sixth-order (· · ·) noncompact centered spatial differences.*

Results.

Spatial difference operators. In this section, we consider the errors produced by the spatial operators alone. The figures show the PPW required to keep both global phase and amplitude errors below 0.1 as a function of the number of wavelengths traveled. Figure 2 shows the grid resolution requirements for second-, fourth-, and sixth-order noncompact centered difference schemes. Since these spatial operators are nondissipative, the grid resolution requirements are determined from the global phase error. The requirements of the second-order scheme are clearly excessive. While more accurate second-order discretizations are available, such as the FDTD scheme or the upwind leapfrog scheme [33], none of these meets our criterion of 30 PPW for 200 wavelengths of travel except under specific conditions.

The PPW requirements for compact centered difference schemes of up to tenth order are shown in Figure 3. The tenth-order scheme requires the solution of a pentadiagonal system of equations. The remaining schemes lead to tridiagonal systems. The compact schemes are considerably more accurate than the noncompact schemes of equivalent order.

We next demonstrate the tradeoffs associated with optimization, using the tridiagonal five-point operators ($\beta = c = 0$) of Haras and Ta'asan as an example. Figure 4 shows the behavior of three different optimized schemes given in [15] as well as the maximum-order scheme which can be obtained using this operator, which is sixth-order. Note the presence of sudden jumps in the grid resolution requirements of the optimized schemes. These are associated with the fact that for an optimized scheme the error does not increase monotonically with increasing wavenumber, as discussed earlier. Their location is dependent on the error threshold used. Reducing the error allowed moves the jumps to the left. With the present error threshold, the scheme denoted "A" is superior up to a distance of travel of about 60 wavelengths. Scheme B

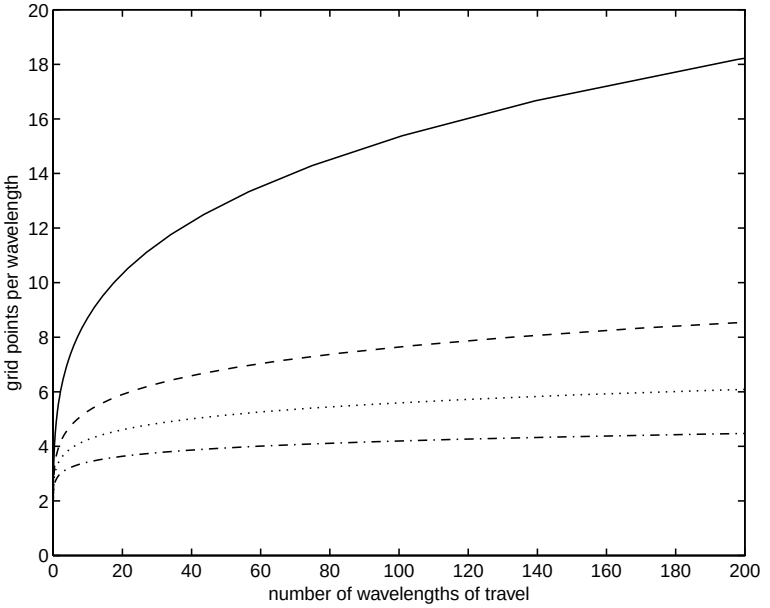


FIG. 3. *Grid resolution requirements for fourth-order (——), sixth-order (- - -), eighth-order (· · ·), and tenth-order (— · —) compact centered spatial differences.*

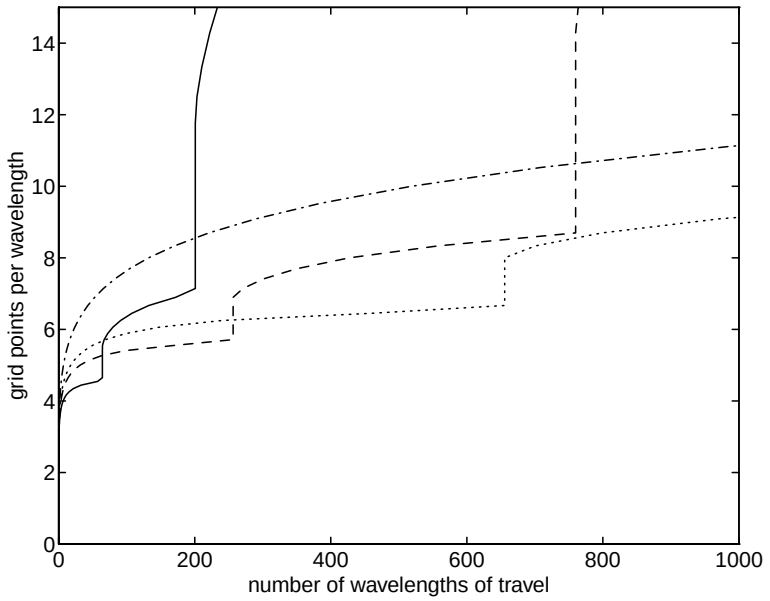


FIG. 4. *Grid resolution requirements for the tridiagonal spatial operators with* 5*-point right-hand stencil of Haras and Ta'asan* [15]: *scheme A (——), scheme B (- - -), scheme C (· · ·), sixth-order scheme (— · —).*

is superior for distances up to 250 wavelengths while scheme C is preferable for even longer distances. Such behavior is typical of optimized schemes. Aggressive optimization (as in the case of scheme A) leads to excellent performance for small distances

of travel but poor performance for longer distances. In this paper we concentrate on distances of travel up to 200 wavelengths. Hence we consider only scheme B further.

Figure 5 shows the PPW required by the optimized schemes of Haras and Ta'asan for distances of travel up to 200 wavelengths. In each case, the best scheme presented by Haras and Ta'asan for this distance of travel is shown, as discussed above. The pentadiagonal scheme requires about 3.7 PPW for 200 wavelengths of travel while the tridiagonal seven-point scheme requires 4.5 and the tridiagonal five-point scheme requires 5.7. The computational effort is roughly proportional to PPW^{d+1}, where d is the number of dimensions, since the number of grid nodes is proportional to PPW^d, and the additional factor of PPW is associated with the decrease in the time step required to maintain a constant Courant number as PPW is increased, i.e., $N = PPWn_w/C$. Therefore the increased cost per grid node of the pentadiagonal scheme is not justified. However, the tridiagonal seven-point scheme is more efficient than the tridiagonal five-point scheme.

The noncompact optimized schemes of Lockard, Brentner, and Atkins [27], Zingg, Lomax, and Jurgens [55], and Tam and Webb [43] are compared with sixth-order centered differences in Figure 6. For the scheme of Lockard, Brentner, and Atkins [27], the PPW requirements are determined by the amplitude error. For the scheme of Zingg, Lomax, and Jurgens [55], the phase and amplitude errors produce roughly equivalent grid resolution requirements over this distance. Since Tam and Webb's scheme [43] is nondissipative, its PPW requirements are determined by the phase error. The figure shows that Tam and Webb's scheme [43] has been optimized for fairly small distances of travel, leading to excellent performance for less than roughly 14 wavelengths of travel. The scheme of Lockard, Brentner, and Atkins [27] requires roughly 8 PPW for 200 wavelengths of travel while that of Zingg, Lomax, and Jurgens [55] requires over 9 PPW.

Figure 7 shows the PPW requirements of the staggered spatial operators. In each case, the staggered schemes are much more accurate than their nonstaggered counterparts of equivalent order. The grid requirements of the second-order scheme are again excessive. However, when used with staggered leapfrog time marching (the FDTD scheme), better results can be obtained. Also shown in Figure 7 is an optimized scheme with $b_3 = 103/19200$, $b_2 = -1315/19200$, and $b_1 = 22630/19200$, which produces excellent performance for up to 200 wavelengths of travel.

Time-marching methods. Figures 8 and 9 show the amplitude and phase errors produced by the four time-marching methods under consideration. The five- and six-stage methods shown are the maximum-order versions rather than the optimized methods, which are discussed in the next subsection. In order to account for the number of stages in the Runge–Kutta methods, the errors are plotted versus $\omega h/p$, where p is the number of stages. Hence the errors shown are for approximately equal computational effort. Since the time step of a p-stage scheme is thus p times larger than that of a single-stage scheme, the amplitude error shown is $|\sigma|^{1/p} - 1$.

The figures show that the phase errors produced by these methods are larger than the amplitude errors. Each increase in the order of the Runge–Kutta method produces an increase in accuracy even though the extra work has been accounted for. The fourth-order Adams–Bashforth method is much more accurate than the fourth-order Runge–Kutta method per unit cost. It produces the lowest amplitude error of the methods considered and phase error comparable to the five-stage Runge–Kutta method.

In order to compare time-marching methods properly, one must consider the

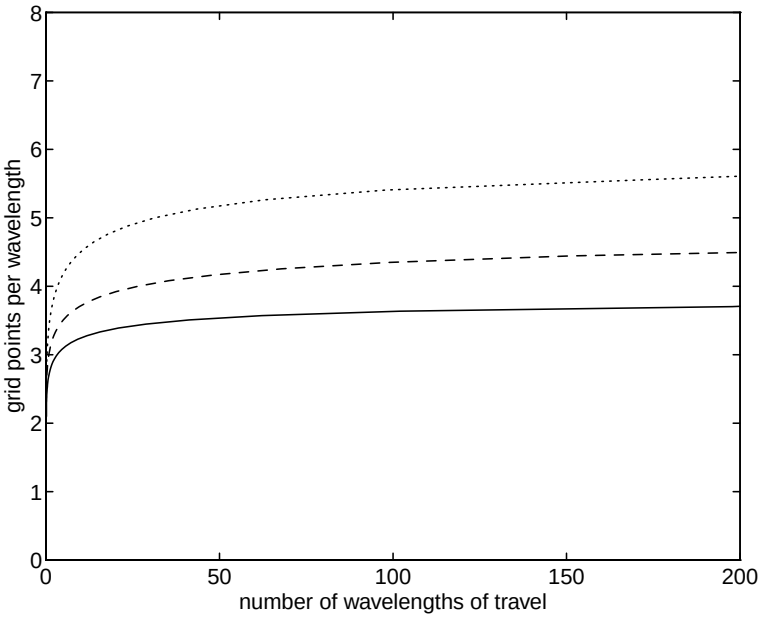


FIG. 5. *Grid resolution requirements for the spatial operators of Haras and Ta'asan* [15]: *pentadiagonal operator with seven-point right-hand stencil (——), tridiagonal operator with seven-point right-hand stencil (- - -), tridiagonal operator with five-point right-hand stencil (· · ·).*

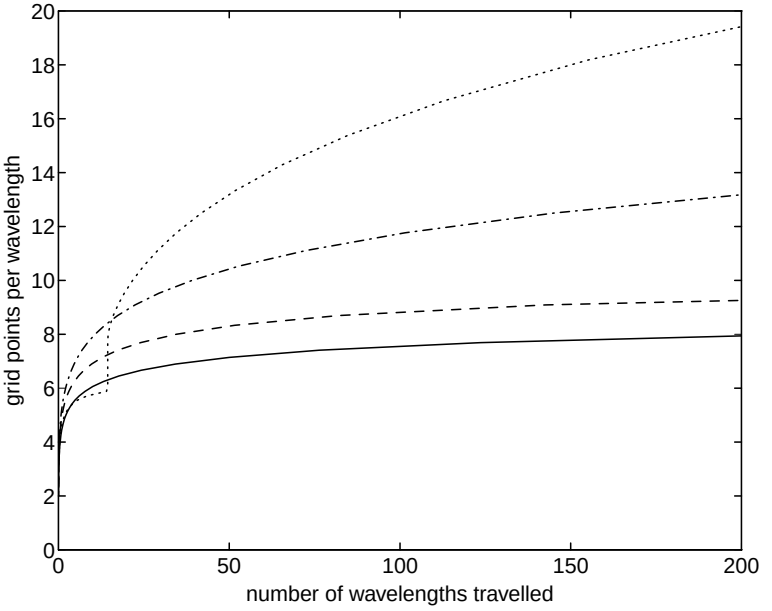


FIG. 6. *Grid resolution requirements for the spatial operators of Lockard, Brentner, and Atkins* [27] *(——), Zingg, Lomax, and Jurgens* [55] *(- - -), Tam and Webb* [43] *(· · ·), and sixth-order centered differences (— · —).*

spatial discretization to be used. For each combination of a spatial discretization and a time-marching method, there is a Courant number which minimizes the computational

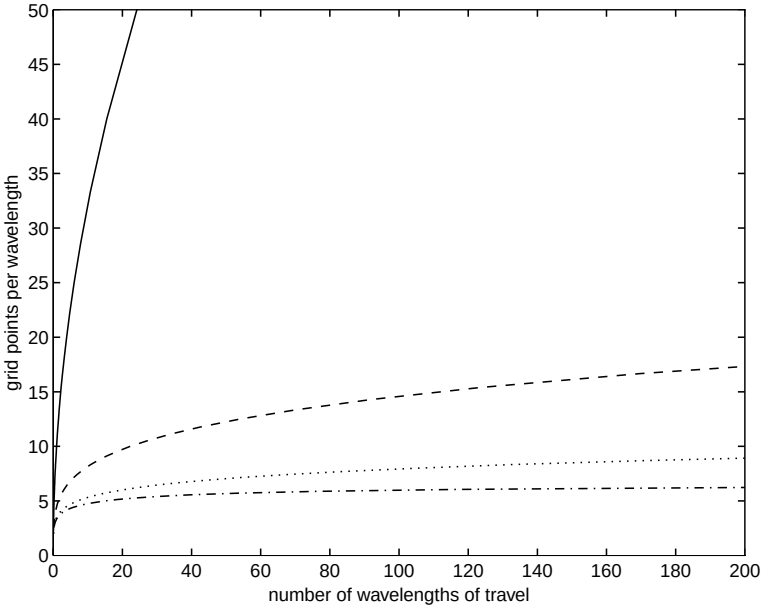


FIG. 7. *Grid resolution requirements for second-order (——), fourth-order (- - -), sixth-order (· · ·), and optimized (— · —) centered spatial differences on a staggered grid.*

work to achieve a given standard of accuracy for a specified distance of travel. The computational work is proportional to the product of the number of derivative function evaluations and the work per function evaluation. The former is given by $pN = pPPWn_w/C$, while the latter is proportional to the number of grid nodes, PPW^d, where p is again the number of stages (which gives the number of function evaluations per time step for the schemes under consideration) and d the number of dimensions. Hence the computational work for a given distance of travel, n_w, is proportional to

$$F = \frac{p}{C} PPW^{d+1}. \tag{27}$$

For example, with fourth-order centered differences in space the optimum Courant number for the fourth-order Runge–Kutta method in two dimensions (determined by trial and error) is about 1.25 for 200 wavelengths of travel and global errors less than 0.1. With this spatial operator, the optimum Courant number for the Adams–Bashforth method is determined by its stability bound, which for pure imaginary λ is roughly 0.43. With fourth-order centered differences in two dimensions, the resulting Courant number is about 0.21. The value of F produced at this Courant number is slightly higher than that produced using the fourth-order Runge–Kutta method at its optimum Courant number. Since the fourth-order Runge–Kutta method also requires less memory than the fourth-order Adams–Bashforth method, it is clearly the preferred method for use with fourth-order centered differences in space.

With sixth-order centered differences in space, the comparison changes. The optimum Courant number for the fourth-order Runge–Kutta method is reduced to roughly 2/3 while that for the fourth-order Adams–Bashforth method remains stability limited, leading to a Courant number of roughly 0.19. In this case, the value of F produced by the Adams–Bashforth method is less than 80% of that for the Runge–Kutta method. Thus the comparison is more complicated, as one must weigh

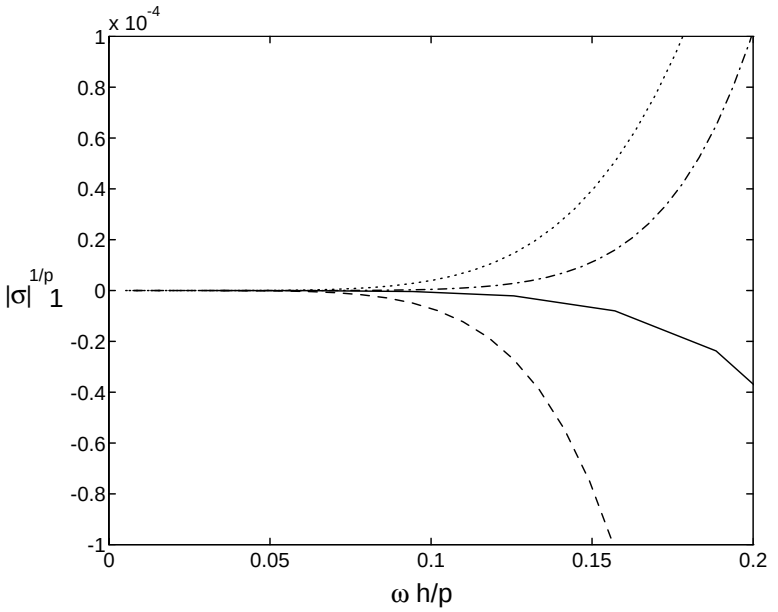


FIG. 8. *Amplitude error produced by the fourth-order Adams–Bashforth method (——), the fourth-order Runge–Kutta method (- - -), the five-stage Runge–Kutta method (· · ·), and the six-stage Runge–Kutta method (— · —).*

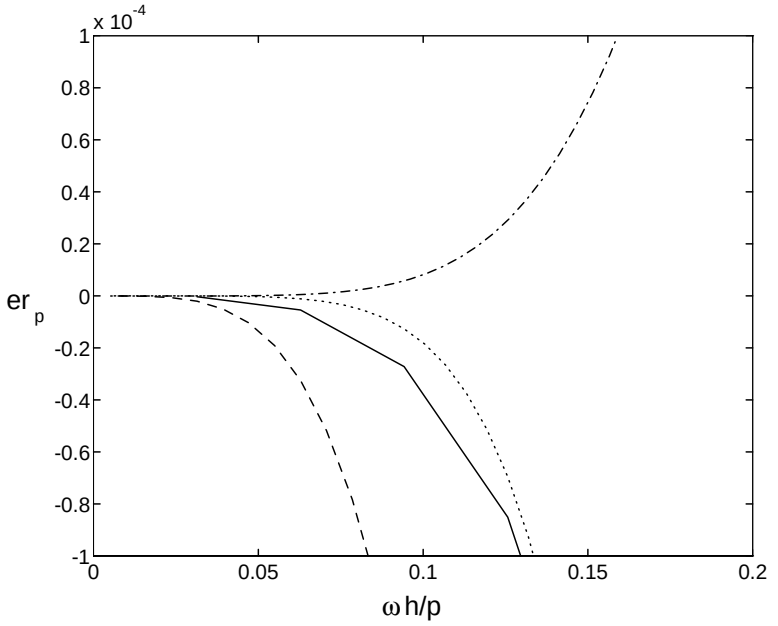


FIG. 9. *Phase error produced by the fourth-order Adams–Bashforth method (——), the fourth-order Runge–Kutta method (- - -), the five-stage Runge–Kutta method (· · ·), and the six-stage Runge–Kutta method (— · —).*

the increased efficiency of the Adams–Bashforth method against the reduced memory requirements of the Runge–Kutta method.

The difficulty with the maximum-order five- and six-stage Runge–Kutta methods

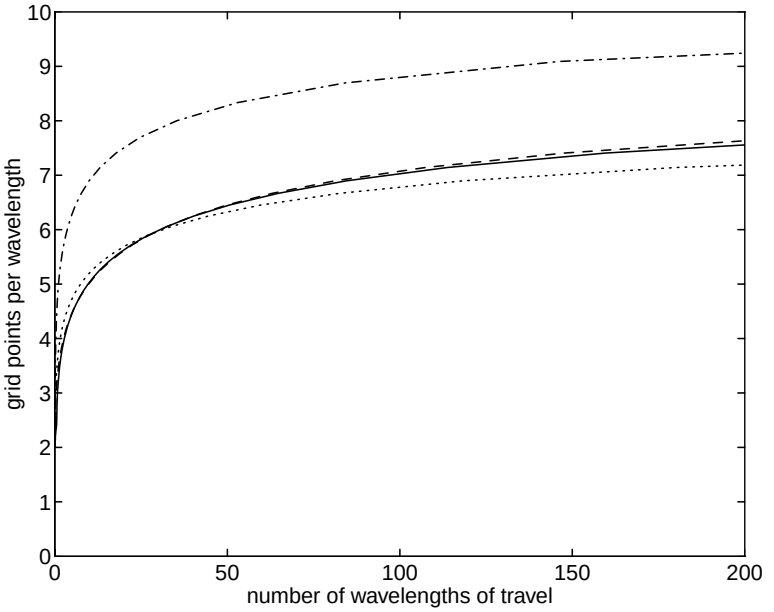


FIG. 10. *Grid resolution requirements for the spatial and temporal operators of Haras and Ta'asan* [15] *at a Courant number of* 0.9: *pentadiagonal operator with seven-point right-hand stencil (——), tridiagonal operator with seven-point right-hand stencil (- - -), tridiagonal operator with five-point right-hand stencil (· · ·); spatial and temporal operators of Zingg, Lomax, and Jurgens* [55] *at a Courant number of* 1 *(— · —).*

is that they are unstable for pure imaginary λ, as shown in Figure 8. We consider these schemes further below.

Combined space-time discretizations. Haras and Ta'asan modified the coefficients of the five-stage Runge–Kutta method to produce stability for pure imaginary λ while maintaining second-order accuracy and optimized error behavior. The methods are designed for $C = 0.9$. Figure 10 shows the grid resolution requirements of the three spatial operators of Haras and Ta'asan compared in Figure 5 in combination with their five-stage time-marching method at a Courant number of 0.9. All three schemes require between 7 and 8 PPW for 200 wavelengths of travel. The advantage of the more accurate spatial operators has been lost. Either a lower Courant number or a more accurate time-marching method should be used.

As a result of the dissipation in the spatial operator, the six-stage time-marching method of Zingg, Lomax, and Jurgens [55] is stable up to a Courant number a little greater than unity in two dimensions. The grid requirements for the combined space-time discretization at a Courant number of unity are shown in Figure 10. Comparison with Figure 6 shows that the time-marching method introduces very little error compared to the spatial differencing. Tam and Webb [43] use an optimized four-step Adams–Bashforth method in conjunction with their spatial operator. It produces little error for Courant numbers less than about 0.3. In both cases, optimization of the time-marching method has a much smaller impact than optimization of the spatial operator.

Figure 11 shows the PPW requirements of the fourth-order staggered spatial difference operator combined with staggered leapfrog time marching. As the Courant number is increased from 0.001 to 0.1, the PPW requirements decrease, since the

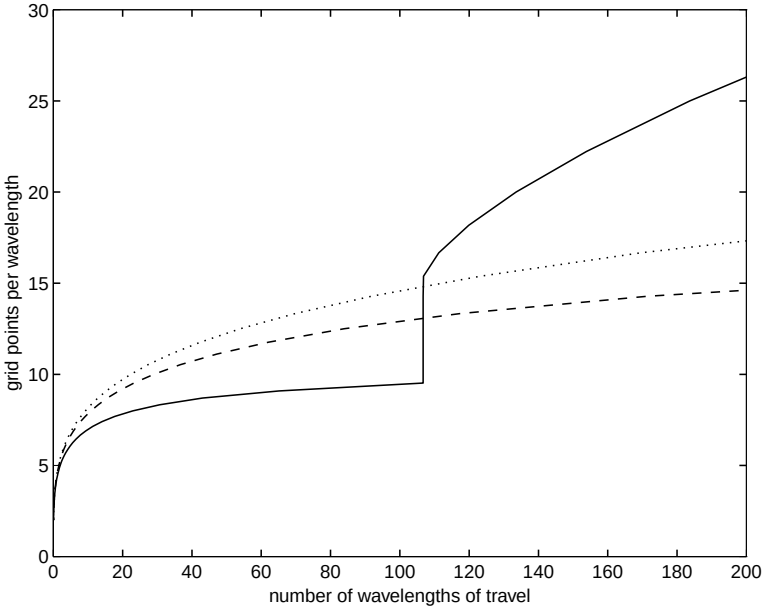


FIG. 11. *Grid resolution requirements for fourth-order centered differences on a staggered grid coupled with staggered leapfrog time marching at Courant numbers of* 0.2 *(——),* 0.1 *(- - -), and* 0.001 *(· · ·).*

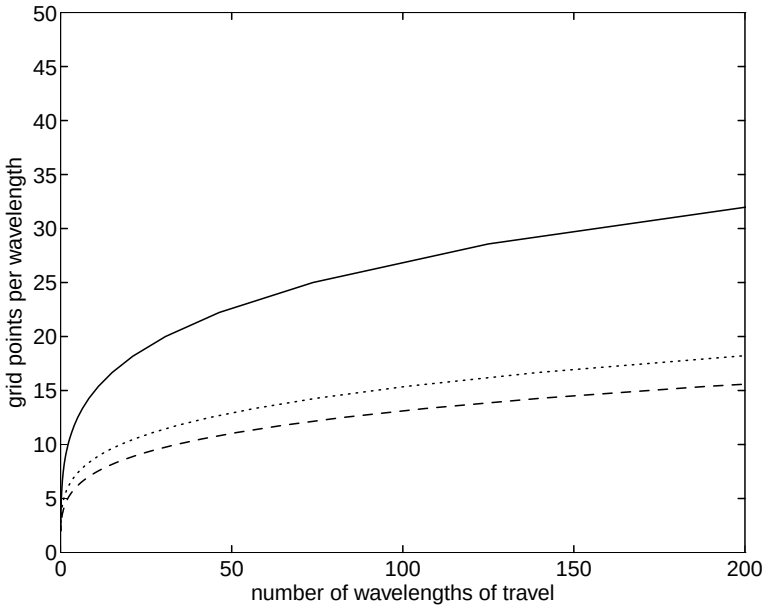


FIG. 12. *Grid resolution requirements for the scheme of Davis* [10] *at Courant numbers of* 3 *(——),* 1.5 *(- - -), and* 0.01 *(· · ·).*

error from the time-marching method is of opposite sign to that of the spatial operator. However, as the Courant number is further increased, the error from the second-order time-marching method begins to dominate. Excellent performance for

200 wavelengths of travel is obtained for Courant numbers up to 0.1.

Results for the method of Davis [10] are shown in Figure 12. Since the method is nondissipative, the PPW requirements are determined from the phase error. This scheme is unconditionally stable. For Courant numbers under 1.5, less than 19 PPW are required for 200 wavelengths of travel. While this is quite good, much better than many schemes, it is not sufficient to justify the additional computational effort associated with an implicit scheme.

Figure 13 shows the grid resolution requirements for the Gottlieb–Turkel scheme, which is stable up to a Courant number of 2/3 in one dimension. This scheme is very robust and has received extensive use in nonlinear applications. Since the method is only second-order in time, high accuracy is obtained only for Courant numbers less than about 0.13. Less than 29 PPW are required to propagate a wave 200 wavelengths with Courant numbers below this value. Although the cost per grid node is reasonably low, these PPW requirements are too high for efficient simulation over such distances.

Finally, the grid requirements of the nondissipative two-step explicit scheme, (14), are shown in Figure 14. This scheme is stable up to a Courant number of unity in one dimension. It has the perfect shift property at this Courant number. The phase error increases as the Courant number is reduced. Less than 29 PPW are required for all stable Courant numbers. Since it provides reasonably low error at Courant numbers up to unity, this scheme has a very low cost per grid node, substantially lower than the Gottlieb–Turkel scheme. However, the PPW requirements are again much higher than some of the other schemes considered.

A numerical example. In this section, we present a numerical simulation of the propagation and reflection of an electromagnetic wave which demonstrates the applicability of the previous analysis in multidimensional cases using nonuniform curvilinear grids. Further details, including the treatment of the boundary conditions, are available in [19]. The governing equations are the transverse magnetic set of the two-dimensional time-domain Maxwell equations. The simulation consists of a pulsed plane wave incident on a perfectly-conducting cylinder. A grid containing 5,400 nodes is shown in Figure 15. Figure 16 shows a snapshot of the electric field intensity computed on a grid with 21,600 nodes using the maximum-order version of the method of Zingg, Lomax, and Jurgens [52, 55]. The dashed contours indicate negative values of the electric field intensity. This solution is visually indistinguishable from that computed using the same method on a grid with 16 times as many nodes, which is used as a baseline to estimate numerical errors. The reflected wave has not yet reached the outer boundary, so spurious reflections are not an issue.

Figures 17 and 18 show the L_2 norm of the numerical error versus the number of grid nodes and the CPU effort, respectively. Four methods are included, the maximum-order (MO) and optimized (O10) schemes of Zingg, Lomax, and Jurgens [52, 55], as well as second- (C2) and fourth-order (C4) centered differences. The second- and fourth-order difference schemes are combined with fourth-order Runge–Kutta time marching and include a very small amount of artificial dissipation for stability. These figures show that the higher-order and optimized methods lead to substantial reductions in both memory and CPU time. The relative grid resolution requirements of the methods are consistent with the predictions obtained from Fourier analysis. Furthermore, the results show that the increased cost per grid node of the higher-order methods is insignificant compared to the reduction in grid resolution required. The benefits of the optimized scheme over the maximum-order scheme are fairly modest due to the nature of the waveform, which is a Gaussian and thus has

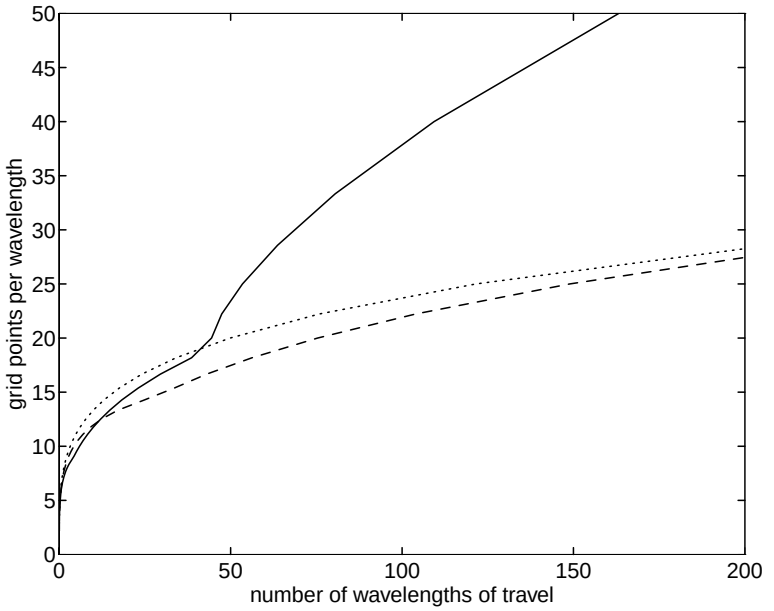


FIG. 13. *Grid resolution requirements for the scheme of Gottlieb and Turkel* [13] *at Courant numbers of* 0.2 *(——),* 0.13 *(- - -), and* 0.01 *(· · ·).*

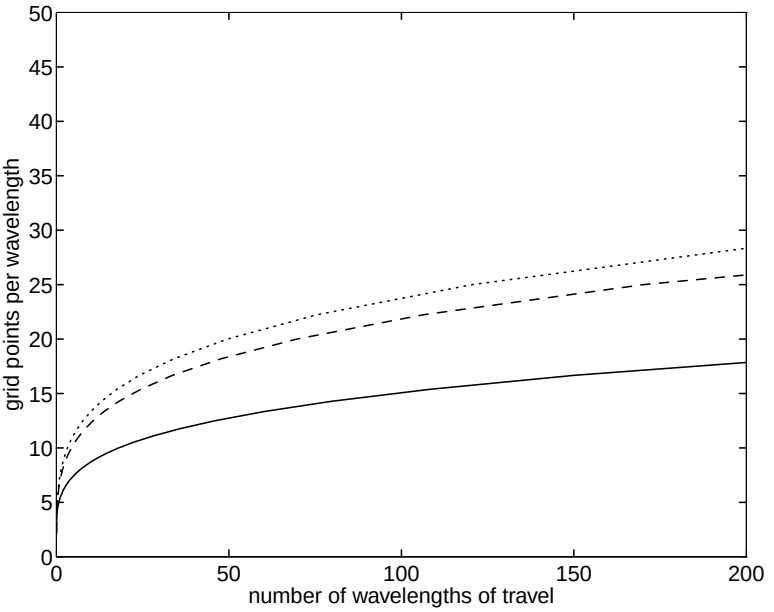


FIG. 14. *Grid resolution requirements for the two-step explicit scheme* (10) *at Courant numbers of* 0.9 *(——),* 0.5 *(- - -), and* 0.01 *(· · ·).*

considerable low-wavenumber content.

Discussion and conclusions. Fourier analysis shows that optimized schemes can provide a significant advantage over their maximum-order counterparts. How-

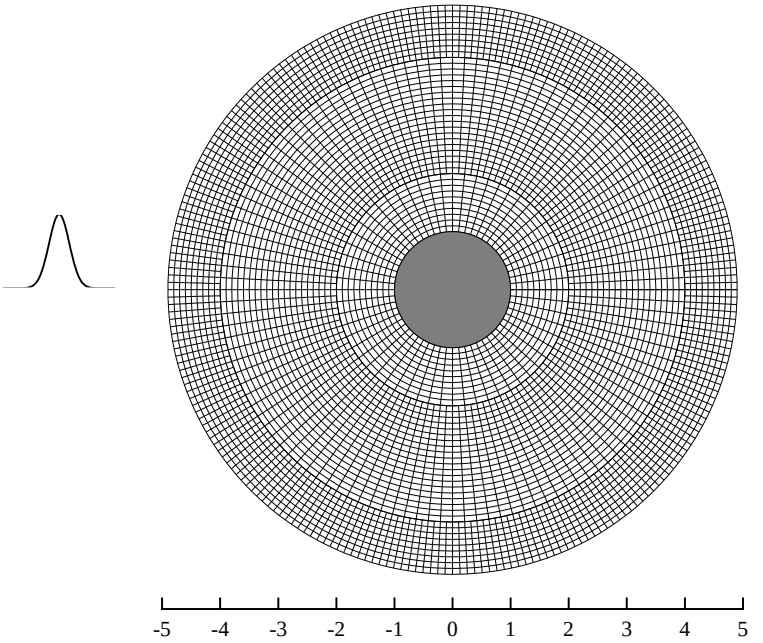


FIG. 15. *Grid for the perfectly-conducting cylinder.*

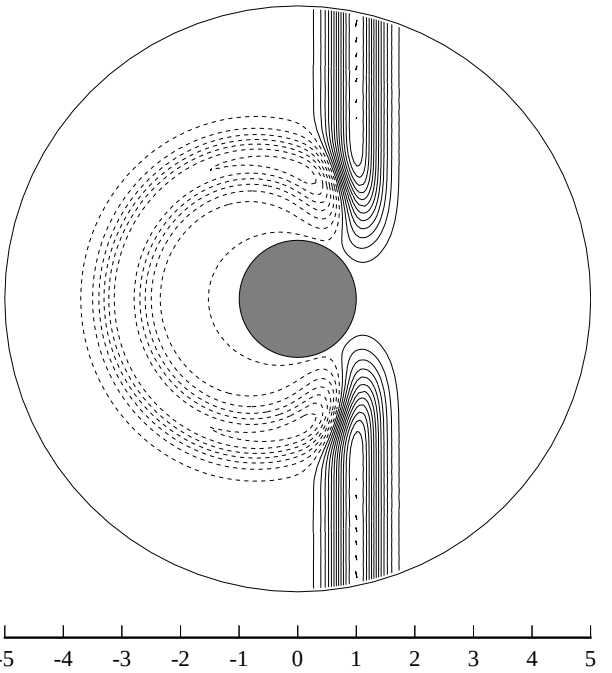


FIG. 16. *Computed contours of electric field intensity.*

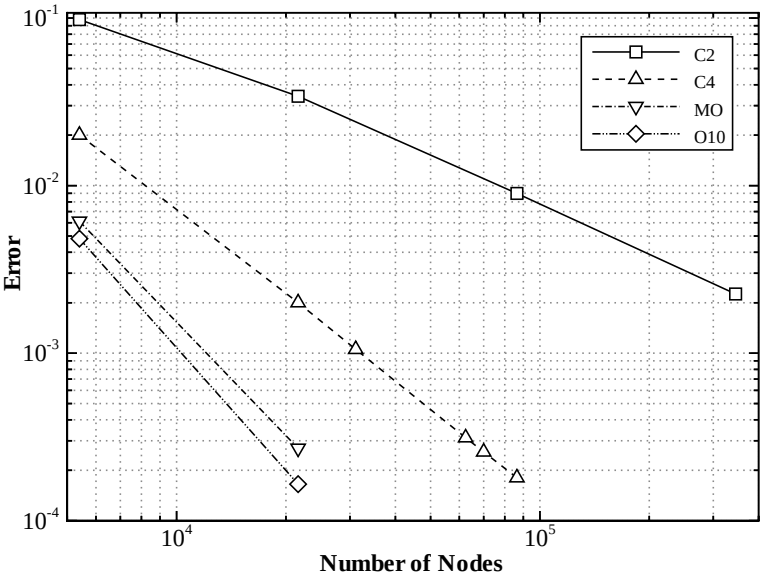


FIG. 17. *Error in the electric field intensity as a function of the number of grid nodes.*

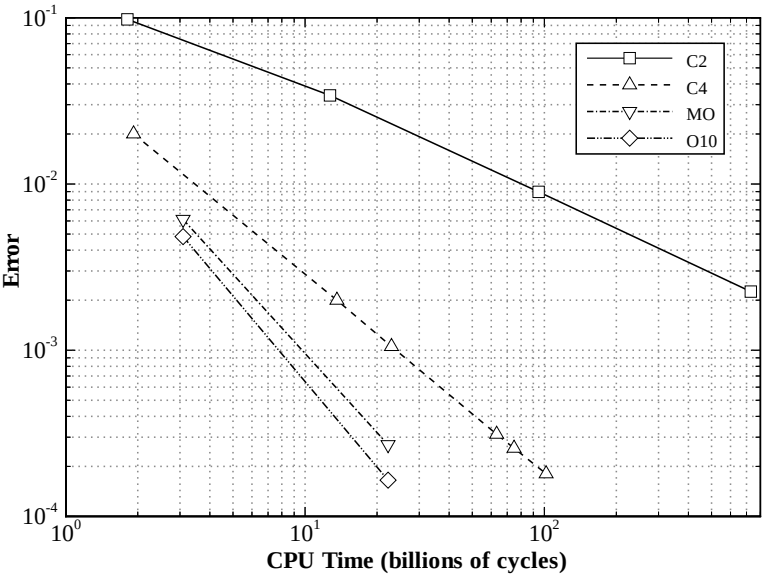


FIG. 18. *Error in the electric field intensity as a function of the CPU time.*

ever, if optimized too aggressively, they perform poorly for longer distances of travel. Furthermore, if a waveform has significant low wavenumber content, as in the case of a Gaussian pulse, the benefits of an optimized scheme can be minimal, and maximum-order schemes can even be superior. The spatial operators of Haras and Ta'asan [15], Lockard, Brentner, and Atkins [27], and Zingg, Lomax, and Jurgens [52, 55] all provide adequate accuracy for 200 wavelengths of travel. The selection of a time-marching method is not quite as critical, since the computational work varies linearly with the time-step size. Consequently, the cost of reducing the time step is much less than the

cost of reducing the PPW, and, furthermore, has no memory implications. Adams–Bashforth and low-storage Runge–Kutta methods can be used, with the latter often preferable because of their reduced memory requirements.

For appropriate problems, such as those involving electromagnetic waves or acoustic waves in a quiescent medium, staggered spatial schemes perform very well. These can be combined with either a high-order time-marching method or the staggered leapfrog method. In the latter case, a low Courant number should be used. For large propagation distances, higher-order time-marching methods are more efficient.

The grid requirements of the explicit fourth-order methods involving simultaneous space-time discretization are reasonably low considering the low cost of these schemes, especially the two-step explicit scheme. This suggests that sixth-order and optimized extensions of these schemes are worthy of investigation.

Based on the results presented, it is clear that high-order and optimized finite-difference methods will play an important role in the simulation of high-frequency linear wave phenomena. Several of the methods studied have the potential to substantially reduce the computational requirements for accurate simulations, including both CPU time and memory. The choice of an optimization strategy can be correlated with a specific distance of propagation. Among the various optimized schemes of Haras and Ta'asan, for example, specific choices can be made if the propagation distance can be estimated. This suggests the use of optimized schemes which are specifically tailored to the spectral content and distance of propagation of a given simulation. In addition to providing a useful reference for scheme evaluation and comparison, our results can be used as a basis for selecting an appropriate grid resolution when applying these schemes.

Appendix. The following are the coefficients of the finite-difference schemes studied. The number of significant figures given is based on the number given in the cited references.

The spatial operator of Haras and Ta'asan is given in (3). The schemes shown in Figure 4 have $\beta = c = 0$. The remaining coefficients are

Scheme A:

$$\begin{aligned} \alpha &= 0.3534620453, \\ a &= 1.566965775, \\ b &= 0.1399583152; \end{aligned}$$

Scheme B:

$$\begin{aligned} \alpha &= 0.3461890571, \\ a &= 1.5633098070, \\ b &= 0.1290683071; \end{aligned}$$

Scheme C:

$$\begin{aligned} \alpha &= 0.3427812069, \\ a &= 1.5614141543, \\ b &= 0.124148259. \end{aligned}$$

In Figure 5, the pentadiagonal seven-point scheme has

$$\alpha = 0.5801818925,$$

$$\begin{aligned}\beta &= 0.0877284887,\\ a &= 1.3058941939,\\ b &= 0.9975884963,\\ c &= 0.0323380724.\end{aligned}$$

The tridiagonal seven-point scheme has

$$\begin{aligned}\alpha &= 0.3904091387,\\ \beta &= 0,\\ a &= 1.5638887738,\\ b &= 0.2348222711,\\ c &= -0.0178927675.\end{aligned}$$

The tridiagonal five-point operator is scheme B above.

In Figure 6, the scheme of Lockard, Brentner, and Atkins is the average of two schemes with the following coefficients, as defined in (6):

$$\begin{aligned}a_{-4} &= 0.0207860419,\\ a_{-3} &= -0.1500704734,\\ a_{-2} &= 0.5234309723,\\ a_{-1} &= -1.34207332539,\\ a_0 &= 0.574548248808,\\ a_1 &= 0.4090357053658,\\ a_2 &= -0.035657169508\end{aligned}$$

and

$$\begin{aligned}a_0 &= 0,\\ a_1 &= -a_{-1} = 0.763289242273,\\ a_2 &= -a_{-2} = -0.160631393818,\\ a_3 &= -a_{-3} = 0.019324515121.\end{aligned}$$

The optimized scheme of Zingg, Lomax, and Jurgens [52, 55] is given by (4) and (5) with the following coefficients:

$$\begin{aligned}a_1 &= 0.75996126,\\ a_2 &= -0.15812197,\\ a_3 &= 0.018760895,\\ d_0 &= 0.1,\\ d_1 &= -0.076384622,\\ d_2 &= 0.032289620,\\ d_3 &= -0.0059049989.\end{aligned}$$

Tam and Webb's scheme is obtained from (4) with (coefficients are from [42])

$$\begin{aligned}a_1 &= 0.770882380518,\\ a_2 &= -0.166705904415,\\ a_3 &= 0.0208431427703.\end{aligned}$$

The five-stage Runge–Kutta method shown in Figures 8 and 9 has the following characteristic polynomial:

$$\sigma(\lambda h) = 1 + \lambda h + \frac{(\lambda h)^2}{2} + \frac{(\lambda h)^3}{6} + \frac{(\lambda h)^4}{24} + \frac{(\lambda h)^5}{120}.$$

The six-stage method is obtained from (8) with $\alpha_5 = 1/2$, $\alpha_4 = 1/3$, $\alpha_3 = 1/4$, $\alpha_2 = 1/5$, $\alpha_1 = 1/6$, leading to the following characteristic polynomial:

$$\sigma(\lambda h) = 1 + \lambda h + \frac{(\lambda h)^2}{2} + \frac{(\lambda h)^3}{6} + \frac{(\lambda h)^4}{24} + \frac{(\lambda h)^5}{120} + \frac{(\lambda h)^6}{720}.$$

The characteristic polynomial of the optimized five-stage Runge–Kutta method of Haras and Ta'asan used in Figure 10 is

$$\sigma(\lambda h) = 1 + \lambda h + \frac{(\lambda h)^2}{2} + 0.166407(\lambda h)^3 + 0.0409525(\lambda h)^4 + 0.0074510(\lambda h)^5.$$

The optimized temporal operator of Zingg, Lomax, and Jurgens [52, 55] is obtained from (8) with the following coefficients:

$$\begin{aligned} \alpha_1 &= 0.168850, \\ \alpha_2 &= 0.197348, \\ \alpha_3 &= 0.250038, \\ \alpha_4 &= 0.333306, \\ \alpha_5 &= 0.5. \end{aligned}$$

The resulting characteristic polynomial is

$$\begin{aligned} \sigma(\lambda h) = 1 + \lambda h + \frac{(\lambda h)^2}{2} + 0.16665295(\lambda h)^3 + 0.041669557(\lambda h)^4 \\ + 0.0082233848(\lambda h)^5 + 0.0013885169(\lambda h)^6. \end{aligned}$$

The scheme of Davis is obtained from (10) with

$$\begin{aligned} a_0 &= b_0 = -2(C-2)(C+2), \\ a_1 &= b_2 = (C-1)(C-2), \\ a_2 &= b_1 = (C+1)(C+2), \end{aligned}$$

where $C = ah/\Delta x$ is the Courant number.

REFERENCES

[1] S. Abarbanel, D. Gottlieb, and J.S. Hesthaven, *Well-posed perfectly matched layers for advective acoustics*, J. Comput. Phys., 154 (1999), pp. 266–283.

[2] R.M. Alford, K.R. Kelly, and D.M. Boore, *Accuracy of finite-difference modelling of the acoustic wave equation*, Geophysics, 39 (1974), pp. 834–842.

[3] D.A. Anderson, J.C. Tannehill, and R.H. Pletcher, *Computational Fluid Mechanics and Heat Transfer*, McGraw-Hill, New York, 1984, p. 73.

[4] J.-P. Berenger, *A perfectly matched layer for the absorption of electromagnetic waves*, J. Comput. Phys., 114 (1994), pp. 185–200.

[5] A.C. Cangellaris, C.-C. Lin., and K.K. Mei, *Point-matched time-domain finite element methods for electromagnetic radiation and scattering*, IEEE Trans. Antennas and Propagation, 35 (1987), pp. 1160–1173.

[6] M.H. Carpenter, D. Gottlieb, and S. Abarbanel, *Stable and accurate boundary treatments for compact, high-order finite-difference schemes*, Appl. Numer. Math., 12 (1993), pp. 55–87.

[7] M.H. Carpenter, D. Gottlieb, and S. Abarbanel, *Time-stable boundary conditions for finite-difference schemes solving hyperbolic systems: Methodology and application to high-order compact schemes*, J. Comput. Phys., 111 (1994), pp. 220–236.

[8] G. Cohen and P. Joly, *Fourth order schemes for the heterogeneous acoustics equation*, Comput. Methods Appl. Mech. Engrg., 80 (1990), pp. 397–407.

[9] M.A. Dablain, *The application of high-order differencing to the scalar wave equation*, Geophysics, 51 (1986), pp. 54–66.

[10] S. Davis, *Low-dispersion finite difference methods for acoustic waves in a pipe*, J. Acoust. Soc. Amer., 90 (1991), pp. 2775–2781.

[11] M. Fusco, *FDTD algorithm in curvilinear coordinates*, IEEE Trans. Antennas and Propagation, 38 (1990), pp. 78–89.

[12] S.D. Gedney, *An anisotropic perfectly matched layer absorbing medium for the truncation of FDTD lattices*, IEEE Trans. Antennas and Propagation, 44 (1996), pp. 1630–1639.

[13] D. Gottlieb and E. Turkel, *Dissipative two-four methods for time-dependent problems*, Math. Comp., 30 (1976), pp. 703–723.

[14] B. Gustafsson, *The convergence rate for difference approximations to mixed initial boundary value problems*, Math. Comp., 29 (1975), pp. 396–406.

[15] Z. Haras and S. Ta'asan, *Finite-difference schemes for long-time integration*, J. Comput. Phys., 114 (1994), pp. 265–279.

[16] O. Holberg, *Computational aspects of the choice of operator and sampling interval for numerical differentiation in large-scale simulation of wave phenomena*, Geophysical Prospecting, 35 (1987), pp. 629–655.

[17] F.Q. Hu, M.Y. Hussaini, and J.L. Manthey, *Low-dissipation and low-dispersion Runge-Kutta schemes for computational acoustics*, J. Comput. Phys., 124 (1996), pp. 177–191.

[18] H.M. Jurgens and D.W. Zingg, *Implementation of a high-accuracy finite-difference scheme for linear wave phenomena*, Proceedings of the Third International Conference on Spectral and High Order Methods, Houston, TX, Houston J. Math. (1995).

[19] H.M. Jurgens and D.W. Zingg, *Numerical solution of the time-domain Maxwell equations using high-accuracy finite-difference methods*, SIAM J. Sci. Comput., to appear.

[20] T.G. Jurgens, A. Taflove, K.R. Umashankar, and T.G. Moore, *Finite-difference time-domain modeling of curved surfaces*, IEEE Trans. Antennas and Propagation, 40 (1992), pp. 357–366.

[21] J.W. Kim and D.J. Lee, *Optimized compact finite difference schemes with maximum resolution*, AIAA J., 34 (1996), pp. 887–893.

[22] J.D. Lambert, *Computational Methods in Ordinary Differential Equations*, Wiley, New York, 1973.

[23] S.K. Lele, *Compact finite difference schemes with spectral-like resolution*, J. Comput. Phys., 103 (1992), pp. 16–42.

[24] J. Lighthill, *The final panel discussion*, in Computational Aeroacoustics, J.C. Hardin and M.Y. Hussaini, eds., Springer-Verlag, New York, 1993, pp. 499–513.

[25] Y. Liu, *A generalized finite-volume algorithm for solving the Maxwell equations on arbitrary grids*, Conference Proceedings of the 10th Annual Review of Progress in Applied Computational Electromagnetics, 1994, pp. 487–494.

[26] Y. Liu, *Fourier analysis of numerical algorithms for the Maxwell equations*, J. Comput. Phys., 124 (1996), pp. 396–416.

[27] D.P. Lockard, K.S. Brentner, and H.L. Atkins, *High accuracy algorithms for computational aeroacoustics*, AIAA J., 33 (1994), pp. 246–251.

[28] N.K. Madsen and R.W. Ziolowski, *A three-dimensional modified finite volume technique for Maxwell's equations*, Electromagnetics, 10 (1990), pp. 147–161.

[29] K.J. Marfurt, *Accuracy of finite-difference and finite-element modeling of the scalar and elastic wave equations*, Geophysics, 49 (1984), pp. 533–549.

[30] P. Olsson, *Summation by parts, projections, and stability* I, Math. Comp., 64 (1995), pp. 1035–1065.

[31] P.G. Petropoulos, *Phase error control for FD-TD methods of second and fourth order accuracy*, IEEE Trans. Antennas and Propagation, 42 (1994), pp. 859–862.

[32] P.G. Petropoulos, L. Zhao, and A.C. Cangellaris, *A reflectionless sponge layer absorbing boundary condition for the solution of Maxwell's equations with high-order staggered finite difference schemes*, J. Comput. Phys., 139 (1998), pp. 184–208.

[33] P.L. Roe, *Linear bicharacteristic schemes without dissipation*, SIAM J. Sci. Comput., 19 (1998), pp. 1405–1427.
[34] A. Sei, *A family of numerical schemes for the computation of elastic waves*, SIAM J. Sci. Comput., 16 (1995), pp. 898–916.
[35] P. Sguazzero, M. Kindelan, and A. Kamel, *Dispersion-bounded numerical integration of the elastodynamic equations with cost-effective staggered schemes*, Comput. Methods Appl. Mech. Engrg., 80 (1990), pp. 165–172.
[36] J.S. Shang, *A fractional-step method for the time domain Maxwell equations*, J. Comput. Phys., 118 (1995), pp. 109–119.
[37] J.S. Shang and D. Gaitonde, *On High Resolution Schemes for Time-Dependent Maxwell Equations*, AIAA Paper 96-0832, 1996.
[38] V. Shankar, A.H. Mohammadian, and W.F. Hall, *A time-domain finite-volume treatment for the Maxwell equations*, Electromagnetics, 10 (1990), pp. 127–145.
[39] G.R. Shubin and J.B. Bell, *A modified equation approach to constructing fourth order methods for acoustic wave propagation*, SIAM J. Sci. Statist. Comput., 8 (1987), pp. 135–151.
[40] A. Taflove, *Computational Electrodynamics: The Finite-Difference Time-Domain Method*, Artech House, Boston, 1995.
[41] A. Taflove, *Application of the finite difference time-domain method to sinusoidal steady-state electromagnetic penetration problems*, IEEE Trans. Electromagnetic Compatibility, 22 (1980), pp. 191–202.
[42] C.K.W. Tam, *Computational aeroacoustics: Issues and methods*, AIAA J., 33 (1995), pp. 1788–1796.
[43] C.K.W. Tam and J.C. Webb, *Dispersion-relation-preserving finite difference schemes for computational acoustics*, J. Comput. Phys., 107 (1993), pp. 262–281.
[44] E. Turkel and A. Yefet, *Fourth order method for Maxwell equations on a staggered mesh*, IEEE Antennas and Propagation Society International Symposium 1997 Digest, 4 (1997), pp. 2156–2159.
[45] R. Vichnevetsky and F. De Schutter, *A Frequency Analysis of Finite Difference and Finite-Element Methods for Initial Value Problems*, in Advances in Computer Methods for Partial Differential Equations, R. Vichnevetsky, ed., AICA/IMACS, Rutgers University, New Brunswick, NJ, 1975, pp. 46–52.
[46] R. Vichnevetsky and J.B. Bowles, *Fourier Analysis of Numerical Approximations of Hyperbolic Equations*, SIAM Stud. Appl. Math. 5, SIAM, Philadelphia, 1982.
[47] M. Vinokur and M. Yarrow, *Finite-Surface Method for the Maxwell Equations in Generalized Coordinates*, AIAA Paper 93-0463, 1993.
[48] M. Vinokur and M. Yarrow, *Finite-Surface Method for the Maxwell Equations with Corner Singularities*, AIAA Paper 94-0233, 1994.
[49] V.L. Wells and R.A. Renaut, *Computing aerodynamically generated noise*, Annu. Rev. Fluid Mech., 29 (1997), pp. 161–199.
[50] K.S. Yee, *Numerical solution of initial boundary value problems involving Maxwell's equations in isotropic media*, IEEE Trans. Antennas and Propagation, 14 (1966), pp. 302–307.
[51] J.L. Young, D. Gaitonde, and J.S. Shang, *Toward the construction of a fourth-order difference scheme for transient EM wave simulation: Staggered grid approach*, IEEE Trans. Antennas and Propagation, 45 (1997), pp. 1573–1580.
[52] D.W. Zingg, H. Lomax, and H.M. Jurgens, *An Optimized Finite-Difference Scheme for Wave Propagation Problems*, AIAA Paper 93-0459, 1993.
[53] D.W. Zingg and H. Lomax, *Finite-difference schemes on regular triangular grids*, J. Comput. Phys., 108 (1993), pp. 306–313.
[54] D.W. Zingg and H. Lomax, *On the eigensystems associated with numerical boundary schemes for hyperbolic equations*, in Numerical Methods for Fluid Dynamics, M.J. Baines and K.W. Morton, eds., Clarendon Press, Oxford, UK, 1993, pp. 471–480.
[55] D.W. Zingg, H. Lomax, and H.M. Jurgens, *High-accuracy finite-difference schemes for linear wave propagation*, SIAM J. Sci. Comput., 17 (1996), pp. 328–346.
[56] D.W. Zingg, and T.T. Chisholm, *Runge-Kutta methods for linear ordinary differential equations*, Appl. Numer. Math., 31 (1999), pp. 227–238.
[57] R.W. Ziolkowski, *Time-derivative Lorentz-material model based absorbing boundary conditions*, IEEE Trans. Antennas and Propagation, 45 (1997), pp. 1530–1535.

SIAM J. SCI. COMPUT.
Vol. 22, No. 2, pp. 503–516
© 2000 Society for Industrial and Applied Mathematics

HIGH-ORDER TOTAL VARIATION-BASED IMAGE RESTORATION*

TONY CHAN†, ANTONIO MARQUINA‡, AND PEP MULET‡

Abstract. The total variation (TV) denoising method is a PDE-based technique that preserves edges well but has the sometimes undesirable *staircase effect*, namely, the transformation of smooth regions (*ramps*) into piecewise constant regions (*stairs*). In this paper we present an improved model, constructed by adding a nonlinear fourth order diffusive term to the Euler–Lagrange equations of the variational TV model. Our technique substantially reduces the staircase effect, while preserving sharp jump discontinuities (edges). We show numerical evidence of the power of resolution of this novel model with respect to the TV model in some 1D and 2D numerical examples.

Key words. image denoising, total variation, fourth order PDE

AMS subject classifications. 68U10, 65F10, 65K10

PII. S1064827598344169

1. Introduction.

The degradation of an image is usually unavoidable during its acquisition and at the early stages of its processing, and it renders the later phases difficult and inaccurate. The classical algorithms for image denoising have been mainly based on least squares, and, consequently, their outputs may be contaminated by Gibbs phenomena and do not approximate images containing edges well. To overcome this difficulty a technique based on the minimization of the total variation (TV) norm is proposed in [12]. This technique preserves edges well, but the images resulting from the application of this technique in the presence of noise are often piecewise constant; thus the finer details in the original image may not be recovered satisfactorily, and *ramps* (affine regions) will give *stairs* (piecewise constant regions); see Figure 8.2. In this paper we present an improved model, constructed by adding a nonlinear fourth order diffusive term to the Euler–Lagrange equations of the variational TV model. This technique substantially reduces the staircase effect, while preserving sharp jump discontinuities (edges).

The paper is organized as follows. In section 2, we describe the original TV model and the nonlinear equations associated with it. In section 3, we describe the staircase effect caused by the TV model and briefly review several techniques proposed in the literature to deal with it. In section 4 we aim to reduce the staircase effect by using a variational formulation for the denoising problem with a functional obtained by adding a nonlinear second order term to the TV functional. In the next section we analyze the fourth order nonlinear Euler–Lagrange equation associated with the latter variational problem in the 1D case and notice that their discretizations can be ill-posed. We then propose to drop the term that might cause the ill-posedness, thus deviating from the variational formulation to obtain the nonlinear fourth order diffusion equation that we propose in this paper. We next present a fixed point scheme for the solution of

*Received by the editors September 1, 1998; accepted for publication (in revised form) February 1, 2000; published electronically July 25, 2000.

http://www.siam.org/journals/sisc/22-2/34416.html

†Department of Mathematics, University of California, 405 Hilgard Avenue, Los Angeles, CA 90024-1555 (chan@math.ucla.edu, http://www.math.ucla.edu/~chan). This author was supported by grants ONR-N00014-96-1-0277, NSF DMS-96-26755, and NSF INT9602089.

‡Departament de Matemàtica Aplicada, Universitat de València, Dr. Moliner, 50, 46100 Burjassot, Valencia, Spain (marquina@uv.es, mulet@uv.es, http://gata.uv.es/~marquina, http://gata.uv.es/~mulet). These authors were supported by NSF grant INT9602089 and DGICYT grant PB97-1402.

this nonlinear equation. In section 6, a straightforward 2D extension of the 1D PDE is explored and more sophisticated possibilities are hinted at. We comment on the choice of parameters in section 7. Finally, in section 8, we discuss some numerical techniques to solve the problem and present numerical experiments comparing the results of the TV denoising technique and the new model.

2. TV denoising. An image can be interpreted as either a real function defined on Ω, a bounded and open domain of $\mathbb{R}^2$ (for simplicity we will assume Ω to be the unit square henceforth), or as a suitable discretization of this continuous image.

Our interest is to restore an image which is contaminated with noise in such a way that the process should recover the edges of the image. Let us denote by z the observed image and by u the real image. The model of degradation we assume is $u + n = z$, where n is a Gaussian white noise; i.e., the values n_i of n at the pixels i are independent random variables, each with a Gaussian distribution of zero mean and variance σ^2.

Our objective is to estimate u from statistics of the noise and some a priori knowledge of the image (smoothness, existence of edges). This knowledge is incorporated into the formulation by using a functional R that measures the quality of the image u in the sense that smaller values of $R(u)$ correspond to better images. The process, in other words, consists in the choice of the best quality image among those matching the constraints imposed by the statistics of the noise.

The usual approach consists in solving the following constrained optimization problem:

$$\begin{aligned} &\min_u R(u) \\ &\text{subject to } ||u - z||^2_{\mathcal{L}^2} = |\Omega|\sigma^2, \end{aligned} \tag{2.1}$$

since $n = z - u$ and $E(\int_\Omega n^2) = |\Omega|\sigma^2$ ($E(X)$ denotes the expectation of the random variable X) give that $||u - z||^2_{\mathcal{L}^2} = \int_\Omega (u - z)^2 \approx |\Omega|\sigma^2$. There are known techniques (see [3]) for solving the constrained optimization problem (2.1) by exploiting solvers for the corresponding unconstrained problem. Therefore, for the sake of clarity, we will assume the Lagrange multiplier λ associated with (2.1) to be known throughout the exposition. For $\alpha = \frac{1}{\lambda}$, we can then write the equivalent unconstrained problem for image denoising

$$\min_u \alpha R(u) + \frac{1}{2}\|u - z\|^2_{\mathcal{L}^2}. \tag{2.2}$$

Here α represents the trade-off between smoothness and fidelity to the original data.

Examples of regularization functionals that can be found in the literature are $R(u) = \|u\|_{\mathcal{L}^2}, \|\Delta u\|_{\mathcal{L}^2}, \|\nabla u\|_{\mathcal{L}^2} = \left(\int_\Omega |\nabla u|^2\right)^{\frac{1}{2}}$, where ∇ is the gradient, $|\nabla u| = \sqrt{u_x^2 + u_y^2}$, and Δ is the Laplacian; see [13, 10]. The main disadvantage of using these functionals is that they do not allow discontinuities in the solution; therefore the edges cannot be satisfactorily recovered.

In [12], the TV is proposed as the regularization functional for the image restoration problem:

$$TV(u) = \int_\Omega |\nabla u|.$$

The TV functional does not penalize discontinuities in u and thus allows us to recover the edges of the original image. The restoration problem can be thus written as

$$\min_u \int_\Omega \left(\alpha |\nabla u| + \frac{1}{2}(u-z)^2 \right). \tag{2.3}$$

The functional of problem (2.3) is strictly convex, coercive, and lower semicontinuous, but it is not differentiable. Therefore the optimality conditions cannot be stated with the usual gradient but with a subgradient. Nevertheless, to overcome this technical difficulty it is common to slightly perturb the TV functional to become

$$\int_\Omega \sqrt{|\nabla u|^2 + \beta},$$

where β is a small positive parameter, or

$$\int_\Omega |\nabla u|_\beta,$$

with the notation ($x \in \mathbb{R}$, $v \in \mathbb{R}^2$)

$$|x|_\beta = \sqrt{x^2 + \beta}, \quad |v|_\beta = \sqrt{|v|^2 + \beta}. \tag{2.4}$$

In [1] it is shown that the solutions of the perturbed problems

$$\min_u \int_\Omega \left(\alpha |\nabla u|_\beta + \frac{1}{2}(u-z)^2 \right) \tag{2.5}$$

converge to the solution of (2.3) when $\beta \to 0$. The Euler–Lagrange equation of (2.5) is

$$0 = -\alpha \nabla \cdot \left(\frac{\nabla u}{|\nabla u|_\beta} \right) + u - z, \tag{2.6}$$

with homogeneous Neumann boundary conditions.

3. The staircase effect. The image restoration model based on the TV tends to yield piecewise constant images, i.e., "blocky" images. Whereas this is certainly useful for many applications, it can be a serious drawback for many others. In particular, when applying this denoising technique to an affine image degraded with noise, the result will invariably be a "staircase," thus creating oversharpening at smooth transitions; see Figure 8.2a.

To prevent the TV oversharpening, one could penalize jumps more. This can be achieved by taking second derivatives into account: intuitively, in discrete form, the first derivative at a height 1 jump of a step function is $\frac{1}{h}$, whereas the second derivative is $\frac{1}{h^2} \gg \frac{1}{h}$ when $h \approx 0$. Somehow, the TV functional does not distinguish between jumps and smooth transitions; therefore we want to analyze functionals that take into account second order derivatives. Successful application of functionals with second (or higher) order derivatives can be found in the works by Geman and Reynolds [9] and Chambolle and Lions [6]. In [4], Blomgren, Chan, and Mulet propose a "TV-H^1 interpolation" approach that avoids the use of second order derivatives to address the staircase problem of the TV technique.

We briefly review the inf-convolution technique of Chambolle and Lions and refer the reader to [6] for further information. Chambolle and Lions propose to use as a

regularizing functional the inf-convolution of the TV functional F_1, and a functional based on second order derivatives

$$F_2 = \alpha \int_\Omega |d^2 u|,$$

where d^2u is the second differential of u and α is a parameter that balances F_1 and F_2. The problem can then be written as finding $u = u_1 + u_2$ such that

$$(u_1, u_2) = \operatorname*{argmin}_{u_1, u_2} F_1(u_1) + F_2(u_2)$$
$$\text{subject to } \frac{1}{2}(\|u_1 + u_2 - z\|_{\mathcal{L}^2} - \sigma^2) = 0.$$

In other words, this method decomposes u into a sum of a smooth function u_2 and a function u_1 that contains the jumps, so that the corresponding Euler–Lagrange equations are solved by u_1, u_2.

4. A second order functional for staircase reduction. In this section we perform a qualitative study of image denoising models aiming to reduce the staircase effect and consisting in the solution of variational problems

$$\min_u R(u) + \frac{1}{2}\|u - z\|^2_{\mathcal{L}^2}, \tag{4.1}$$

where

$$R(u) = \int_\Omega f(\nabla u, \mathcal{L}(u)) \tag{4.2}$$

for a real function f and an elliptic operator $\mathcal{L}(u)$. Note that in (4.1) we have included α in the functional $R(u)$. The latter functionals should retain the good properties of the TV functional near the "true" edges and penalize "wrong" edges created in regions which "should" be smooth. The naïve choice of, e.g.,

$$\int_\Omega (\alpha |\nabla u|_\beta + \mu(\mathcal{L}(u))^2) \tag{4.3}$$

results in a high global penalization of jumps that smooths the data excessively. Bearing this in mind, we consider an *adaptive* functional, in which the action of the second order term is lessened where the gradient is large:

$$\int_\Omega (\alpha |\nabla u|_\beta + \mu \Phi(|\nabla u|)(\mathcal{L}(u))^2), \tag{4.4}$$

where Φ is a real function that indicates the presence of edges in the sense that its value approaches 0 when the gradient $|\nabla u|$ is large, i.e., near edges.

The motivation for the choice of this precise form for the additional second order term is that it is quadratic in $\mathcal{L}(u)$; hence, the numerical treatment is easier than merely choosing $|\mathcal{L}(u)|$. The additional complexity caused by the term $\Phi(|\nabla u|)$ in (4.4) does not render it much more complicated than (4.3), for this equation is already nonquadratic.

A family of functions that meet the above requirements is

$$\Phi(x) = 1/(\sqrt{x^2 + \gamma})^p, \quad p > 0, \tag{4.5}$$

which can be written more compactly as $1/|x|^p_\gamma$, using the notation introduced in (2.4). A balance between the two differential terms of (4.4) can be achieved by the choice of exponent $p = 3$, for then both depend on the scale h as $\mathcal{O}(\frac{1}{h})$, thus making the functional $R(u)$ independent of the scale h: intuitively,

$$\alpha|\nabla u|_\beta + \mu\frac{\mathcal{L}(u)^2}{|\nabla u|^p_\gamma} \approx \mathcal{O}(h^{-1}) + \frac{\mathcal{O}((\frac{1}{h^2})^2)}{\mathcal{O}(\frac{1}{h^p})} \approx \mathcal{O}(h^{-1}) + \mathcal{O}(h^{p-4}),$$

which gives $p = 3$, by equating the exponents of both terms. With this choice of exponent, the functional in (4.4) now reads

$$\int_\Omega \left(\alpha|\nabla u|_\beta + \mu\frac{\mathcal{L}(u)^2}{|\nabla u|^3_\gamma}\right), \tag{4.6}$$

and the denoising variational problem is thus

$$\min_u \int_\Omega \left(\alpha|\nabla u|_\beta + \mu\frac{\mathcal{L}(u)^2}{|\nabla u|^3_\gamma} + \frac{1}{2}(u-z)^2\right). \tag{4.7}$$

Furthermore, concerning the energy of monotone 1D signals, let us consider a function u that is monotone on $[x_i, x_{i+1}]$, $0 = x_1 < \cdots < x_{n+1} = 1$; then its TV is $\sum_{i=1}^n |u(x_{i+1}) - u(x_i)|$. Applying this to $u_1 = \chi_{(\frac{1}{2},1)}$ and $u_2 = x$, we get $TV(u_1) = TV(u_2) = 1$. Let us examine the second term of the energy in (4.6) for the natural choice $\mathcal{L}(u) = u''$. Since $u_2' = 1$ and $u_2'' = 0$, the contribution of the second order term is null; therefore the energy of u_2 is α. A suitable discretization of $I_2 = \int_0^1 \frac{(u'')^2}{((u')^2+\gamma)^{\frac{3}{2}}}$ for $u = u_1$ gives $I_2 \approx h\frac{(\frac{1}{h^2})^2}{(\frac{1}{h^2}+\gamma)^{\frac{3}{2}}} = \frac{1}{(1+\gamma h^2)^{\frac{3}{2}}} \to 1$ when $h \to 0$. The energy, computed from (4.7), of u_1 is then $\alpha + \mu \cdot 1 = \alpha + \mu$; therefore, it is bigger than the energy of u_2. Thus, this functional "prefers ramps to stairs."

The functionals in (4.6) are not convex; therefore some good mathematical properties of (strictly) convex functionals, as uniqueness of solutions, are not present. Use of nonconvex functionals in the image processing field can be found in [9, 14, 2].

5. The 1D model. For the reasonable choice of $\mathcal{L}(u) = u''$, the Euler–Lagrange equation for the 1D denoising variational problem (4.7) reads:

$$\begin{aligned} 0 &= (\kappa_2 u'')'' - (\kappa_1^* u')' + u - z, \\ \kappa_1^* &= \frac{\alpha}{|u'|_\beta} - 3\mu\frac{(u'')^2}{|u'|^5_\gamma}, \\ \kappa_2 &= \frac{2\mu}{|u'|^3_\gamma}, \end{aligned} \tag{5.1}$$

where the natural boundary conditions, obtained by integration by parts, are

$$u'(x) = (\kappa_2(u')u'')'(x) = 0, x = 0, 1.$$

In this 1D case the latter boundary conditions can be simplified to

$$u'(x) = u'''(x) = 0, x = 0, 1. \tag{5.2}$$

The coefficient κ_2 is strictly positive, whereas κ_1^* may be negative. This can be problematic when solving discretizations of these equations, for then it may cause

numerical difficulties due to the lack of maximum principles for this fourth order equation. Near the jump discontinuities of u, κ_2 becomes $O(h^3)$ and κ_1^* is about $O(h)$, where h is the spatial step-size of the representation of u. Thus the second order term dominates the equation, and therefore the positivity of the coefficient κ_1^* is an unavoidable requirement to prevent the increasing in the TV of u that would otherwise make the solution of (5.1) ill-conditioned. Therefore, we introduce a regularization, consisting in redefining κ_1^* to be $\kappa_1 = \frac{\alpha}{|u'|_\beta} > 0$, which can be interpreted as adding a *nonlinear viscosity* to the original equation. To recap, the regularized equations that we propose are

$$\begin{aligned} 0 &= (\kappa_2 u'')'' - (\kappa_1 u')' + u - z, \\ \kappa_1 &= \frac{\alpha}{|u'|_\beta}, \\ \kappa_2 &= \frac{2\mu}{|u'|_\gamma^3}, \end{aligned} \tag{5.3}$$

with boundary conditions given by (5.2). Note that if $\alpha = 0$ we still have a fourth order anisotropic diffusion equation with denoising effect.

This regularization makes the equation satisfy a local maximum principle in the sense that suitable discretizations give positive definite frozen Jacobians. Near the jumps the model approaches the TV model, since the coefficient $\kappa_2 = \frac{2\mu}{|u'|_\gamma^3}$ of the additional fourth order term behaves as $\mathcal{O}(h^3)$. Furthermore, (5.3) can be regarded as adding the fourth order nonlinear term $(\kappa_2 u'')''$ to the Euler–Lagrange equations (2.6). It is worthwhile to point out that, due to the lack of symmetry, (5.3) is not the Euler–Lagrange equation of any variational problem.

For the numerical solution of (5.3), we could consider the steady-state solution of the time evolving equation

$$u_t = -(\kappa_2 u_{xx})_{xx} + (\kappa_1 u_x)_x - u + z, \tag{5.4}$$

where u is now a function of t and x. The time evolution equation is parabolic since κ_2 is strictly positive, and we know that the constant coefficient parabolic equation

$$u_t = -a\, u_{xxxx} + b\, u_{xx} - u + z, \tag{5.5}$$

with $a > 0$ and initial data in $\mathcal{L}^2$, is a well-posed problem (see [11, p. 271]). Equation (5.4) is, of course, much more complicated than (5.5), but this can serve as a model to understand the local behavior of the linear equation obtained after freezing the coefficients of (5.4).

In our case, we have nonlinear coefficients that change with u' so that time marching procedures are usually stiff and inefficient, as practice has shown in that kind of problem. The CFL stability restriction for explicit schemes (see [11]) requires the time step to be $\mathcal{O}(h^4)$, which precludes their use in any realistic computation.

The *lagged diffusivity fixed point* iteration has proven lately to be a quite popular linearization method for solving the TV restoration problem; see [15]. In the same fashion, we can consider an iterative procedure that aims to converge to a solution of (5.3), starting from the initial approximation $u_0 = z$ and solving for u^{n+1} the following the linear fourth order elliptic equation at every step:

$$0 = (\kappa_2(u_n) u''_{n+1})'' - (\kappa_1(u_n) u'_{n+1})' + u_n - z, \quad n = 0, 1 \ldots. \tag{5.6}$$

Remark 1. We have observed in our experiments that our model with $\alpha = 0$ and sound choices of μ, i.e.,

$$\begin{aligned} 0 &= (\kappa_2 u'')'' + u - z, \\ \kappa_2 &= \frac{2\mu}{|u'|_\gamma^3}, \end{aligned}$$

can be used for denoising purposes. It recovers sharp edges and does not produce staircasing, but it may create artificial nonnull gradients at flat regions and some overshooting that do not appear for sufficiently big α, since those features then have a TV penalty.

6. The 2D model. In the 2D case there are many possible choices for the elliptic operator $\mathcal{L}(u)$. One could choose the *directional Laplacian*

$$\mathcal{L}(u) = \left(\frac{\nabla u}{|\nabla u|}\right)^T H(u) \left(\frac{\nabla u}{|\nabla u|}\right) = \frac{u_{xx}u_x^2 + 2u_{xy}u_x u_y + u_{yy}u_y^2}{u_x^2 + u_y^2},$$

where $H(u)$ denotes the Hessian matrix of u. However, the Euler–Lagrange equations for the isotropic Laplacian $\mathcal{L}(u) = \Delta u = u_{xx} + u_{yy}$ are easier to tackle numerically; therefore we will restrict ourselves to treat this case in this paper.

With the latter choices of elliptic operator, the Euler–Lagrange equation of (4.7) is the following:

$$\begin{aligned} 0 &= \Delta(\kappa_2 \Delta u) - \nabla \cdot (\kappa_1^* \nabla u) + u - z, \\ \kappa_1^* &= \frac{\alpha}{|\nabla u|_\beta} - 3\mu \frac{(\Delta u)^2}{|\nabla u|_\gamma^5}, \\ \kappa_2 &= \frac{2\mu}{|\nabla u|_\gamma^3}, \\ 0 &= \frac{\partial u}{\partial n}, \\ 0 &= \frac{\partial\big(\kappa_2 \Delta u\big)}{\partial n}, \end{aligned} \tag{6.1}$$

where n is the outward unit normal vector to the boundary.

The same stability issues as in the 1D case apply here. We drop the negative term in κ_1 to add appropriate artificial viscosity to (6.1). Thus, the resulting equations are

$$\begin{aligned} 0 &= \Delta(\kappa_2 \Delta u) - \nabla \cdot (\kappa_1 \nabla u) + u - z, \\ \kappa_1 &= \frac{\alpha}{|\nabla u|_\beta}, \\ \kappa_2 &= \frac{2\mu}{|\nabla u|_\gamma^3}, \end{aligned} \tag{6.2}$$

with the same boundary conditions as in (6.1).

As in the 1D case, we could use the fixed point scheme

$$0 = \Delta(\kappa_2(u_n)\Delta u_{n+1}) - \nabla \cdot (\kappa_1(u_n)\nabla u_{n+1}) + u_{n+1} - z, \tag{6.3}$$

solving at each iteration a linear fourth order elliptic equation. Nevertheless, in our experiments we have found that it does not show a strongly stable convergence behavior in the sense that the norm of the right-hand side of (6.2) does not decrease

monotonically towards 0. However, a suitable modification of it has shown smoother convergence in our experiments. The modified scheme consists in solving equation (6.3) for $\hat{u}_{n+1}$ and then underrelaxing by setting u_{n+1} to be the average $\frac{1}{2}(u_n + \hat{u}_{n+1})$, i.e.,

$$\begin{aligned} &\Delta(\kappa_2(u_n)\Delta\hat{u}_{n+1}) - \nabla\cdot(\kappa_1(u_n)\nabla\hat{u}_{n+1}) + \hat{u}_{n+1} = z, \\ &u_{n+1} = \frac{u_n + \hat{u}_{n+1}}{2}. \end{aligned} \tag{6.4}$$

Underrelaxation is a classical procedure used to avoid weak instability.

The same issues as in the 1D model about using (6.2) with $\kappa_1 = 0$ apply here as well.

7. Choice of parameters. We comment briefly on the choice of the various parameters in the algorithm described above. The easiest parameter to choose is β. If d is the dynamic range of the image (i.e., u takes values from the interval $[0, d]$), then we recommend from our experience using $\beta = 10^{-5}d^2$, so β scales properly with the term $|\nabla u|^2$ that appears in (2.4). For the same scaling reasons and to get a satisfactory removal of staircasing, we strongly recommend $\gamma = d^2$, i.e., $\gamma = 1$ if the picture takes values in $[0, 1]$.

The other two parameters, α, μ, are more complicated to choose. We describe now a heuristic procedure that has proven to be quite successful in our experiments. We make the usual assumption that the noise variance σ^2 is known, as is, e.g., the setting of [12].

We have observed that the fixed point scheme ((5.6) in 1D and (6.4) in 2D) with $\kappa_1 = 0$ converges nicely for μ less than some threshold value μ_0, but it gives oscillations of the norm of the equations and bad conditioning of the frozen Jacobians for μ sufficiently bigger than μ_0 ($\mu > 1.5\mu_0$, say). We recommend selecting $\mu = \mu_0$ and then adjusting α so that u matches the noise variance, i.e., $\|u - z\|^2_{\mathcal{L}^2} = \sigma^2$. This matching can be accomplished by using the techniques of [3] or by taking into account that $\|u_\alpha - z\|_{\mathcal{L}^2}$ typically is an increasing function of α, where u_α is the solution of the problem for that α.

8. Numerical results. In this section, we perform some numerical experiments in 1D and 2D with the new model. Throughout this section, the parameters are chosen as described in section 7 so that a match of the noise level is achieved. The linear PDEs in (5.6) and (6.4) are discretized by standard forward differences, except that a wider stencil is used for the computation of κ_2. This discretization is obtained by applying standard forward differences to the smoothed image $g * u$, where

$$g = G^{(k)} = G *^{(k)} \cdots * G, \quad G = \frac{1}{4}\begin{bmatrix} 1 & 2 & 1 \end{bmatrix},$$

in the 1D case and

$$g = G^{(k)} = G *^{(k)} \cdots * G, \quad G = \frac{1}{8}\begin{bmatrix} 0 & 1 & 0 \\ 1 & 4 & 1 \\ 0 & 1 & 0 \end{bmatrix}, \tag{8.1}$$

in the 2D case. This technique is inspired by [5] and results in the expression

$$\kappa_2 = \frac{2\mu}{|\nabla(g * u)|^3_\gamma} = \frac{2\mu}{|(\nabla g) * u|^3_\gamma}. \tag{8.2}$$

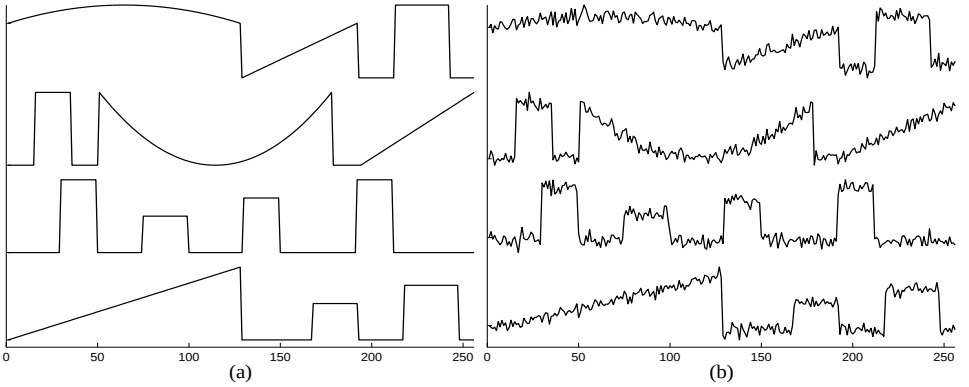


FIG. 8.1. (a) *Original* 1*D images;* (b) *noisy* 1*D images, SNR*≈ 5.

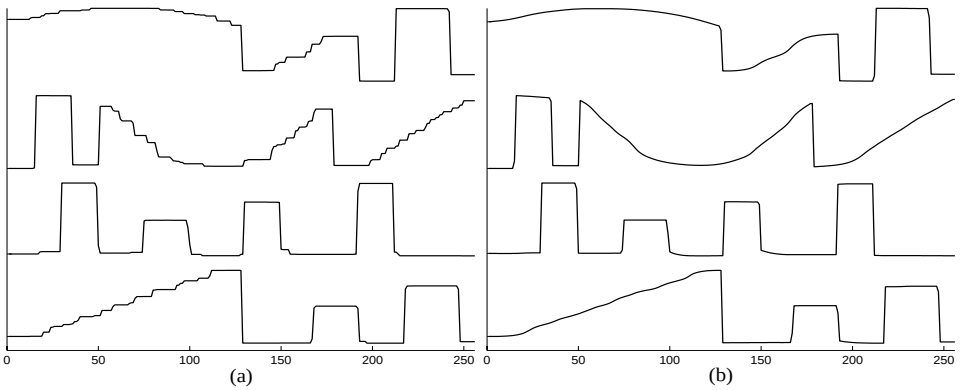


FIG. 8.2. (a) *Results of TV restoration;* (b) *results of our model.*

We have observed that using standard differences for the computation of κ_2, i.e., $k = 0$ in (8.2), does not give convergent schemes.

The resulting systems of linear equations (with symmetric and positive definite matrices) are solved by Gaussian elimination in 1D and the conjugate gradient method preconditioned by SSOR in 2D.

In the first experiment we compare the result of the restoration using the new model with the result of the constrained TV restoration for a number of test 1D images, including piecewise constant, piecewise linear, and piecewise parabolic 1D images. The original 1D images are shown in Figure 8.1a, and the (artificially) degraded 1D images in Figure 8.1b. The results of the TV denoising method are shown in Figure 8.2a. We show in Figure 8.2b, right, the result of our model. We make a final comparison by overlaying the original, TV restoration and the results of our model for some of the test 1D images, and we show it in Figure 8.3. We observe that the results obtained by the new model have sharp gradients at the "true" edges but smooth transitions at ramps, whereas the TV model shows the typical staircase effect.

In the next experiment, we compare the results of our model to Chambolle–Lions's inf-convolution approach. In Figure 8.4a, we plot several solutions of our model for some μ's differing by power of 2 factors from the parameter $\mu = 13.89$ selected by our algorithm. We point out that the solution corresponding to $\mu = 27.78$ (topmost

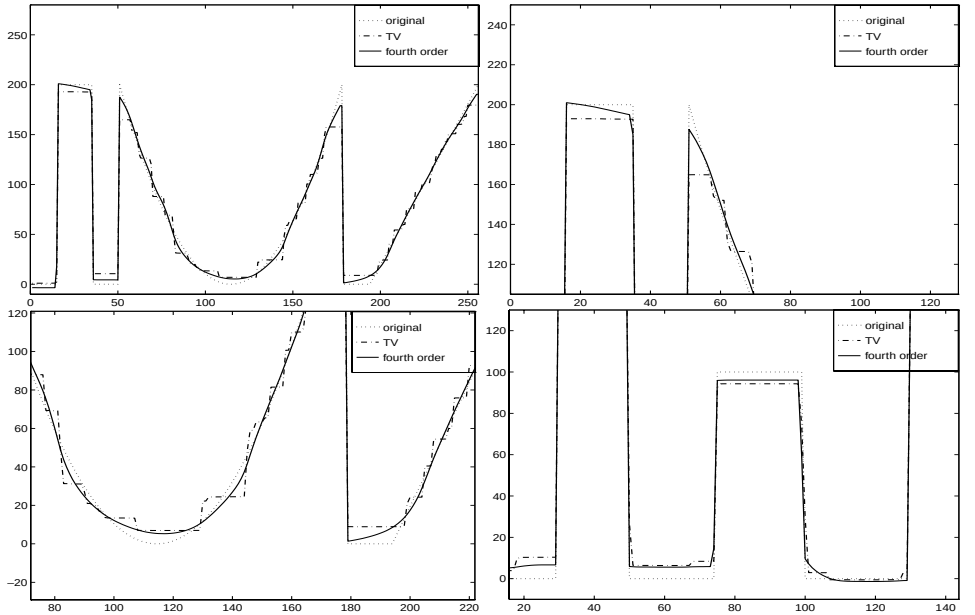


FIG. 8.3. *Details of comparison of TV denoising and our model.*

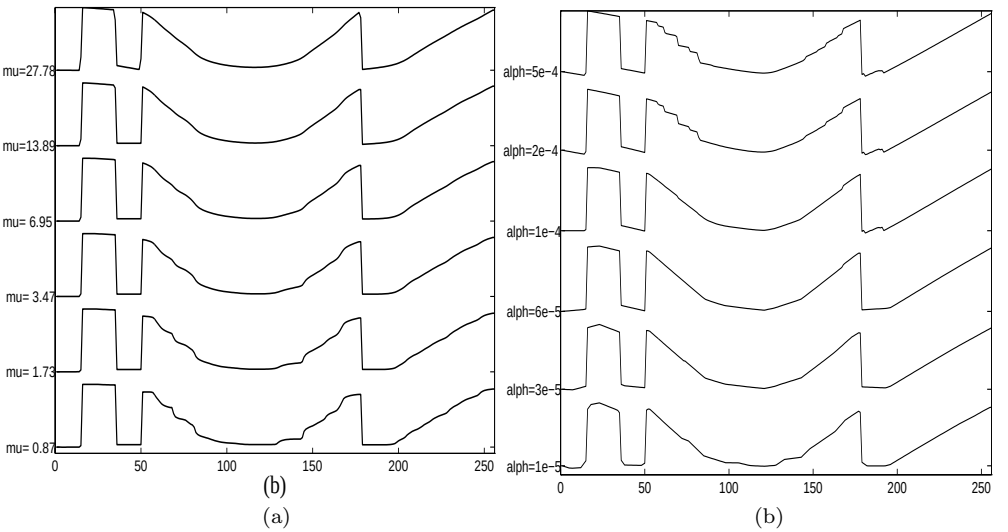


FIG. 8.4. *Comparison of our model and* inf*-convolution models for several parameters.*

one) does not match the noise level due to the nature of the selection procedure for μ expounded in section 7. In Figure 8.4b, we display some solutions of the inf-convolution technique for the values of the parameter $\alpha = 2^{-16}, 2^{-15}, \ldots, 2^{-11}$, where a match with the noise level has been enforced in our algorithms. The conclusion we can draw from this experiment is that the model we present in this paper is quite less sensitive to the parameter choice than the inf-convolution approach, since, for the same range of variation of the parameter that controls the intensity of the fourth order term of the equation (μ in our model and α in Chambolle–Lions'), our results look

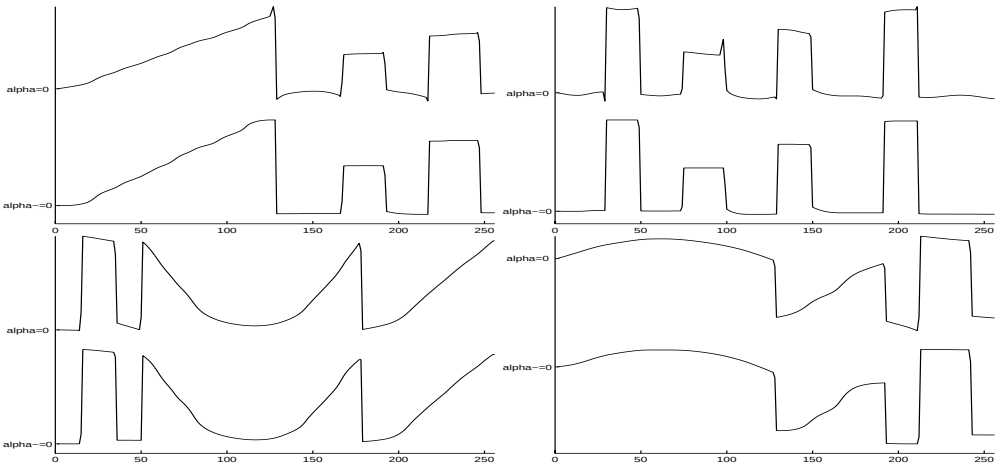


FIG. 8.5. *Comparison of our model with* $\alpha = 0$ *and* $\alpha \neq 0$.

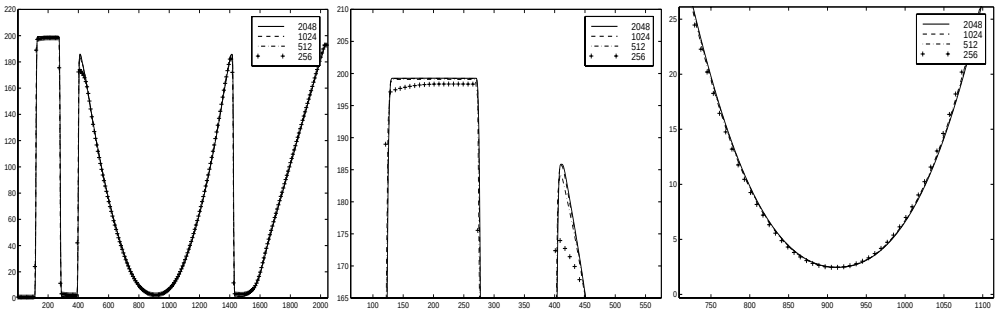


FIG. 8.6. *Details of comparison of solutions for* 256, 512, 1024, 2048 *grid points.*

much more uniform. Furthermore, our results exhibit no staircasing in the topmost three and sharp edges in all plots, whereas the best visual result of the inf-convolution technique, corresponding to $\alpha = 2^{-14} \approx 6 \times 10^{-5}$, does not contain staircasing, but some spurious gradients have been created in zones that should be flat. Actually, for $\mu = 0.87$ (at the bottom, $\mu = \frac{1}{16} \times 13.89$) only a mild staircase effect appears in the results of our model. Furthermore, the heuristics applied to the choice of the parameter μ explained in section 7 apparently give good results.

In Figure 8.5 we compare the results of our model with $\alpha = 0$ and $\alpha \neq 0$. This comparison backs the statements in Remark 1.

In the next experiment we analyze the dependence of the solution of the new model with respect to the grid size. For this, we use the signal that appears in the second place from the top of Figure 8.1a, whose size is $256 = 2^8$. We interpolate it to sizes $2^9, \ldots, 2^{15}$ and apply our model, with the parameters chosen as in the first experiment, to each of these interpolated signals as "noisy data" z. We overlay some of these solutions in Figure 8.6 and plot in Figure 8.7 the 2-, 1-, and ∞-norms of the difference of each solution with the solution at the finest grid. The errors thus computed behave as $\mathcal{O}(h^p)$, where $p \approx 1.5$ for the 2-norm, $p \approx 1$ for the 1- and ∞-norms. As can be seen from the pictures, the difference between these signals is very small, which seems to suggest that the PDE that we propose is well-posed.

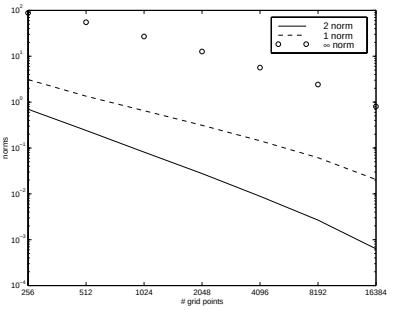


FIG. 8.7. *Plot of the* 1-, 2-, *and* ∞*-norms of the error between the solution of our scheme in a fine grid of* 2^{15} *points to the solutions in coarser grids of* $2^8, \ldots, 2^{14}$ *points.*

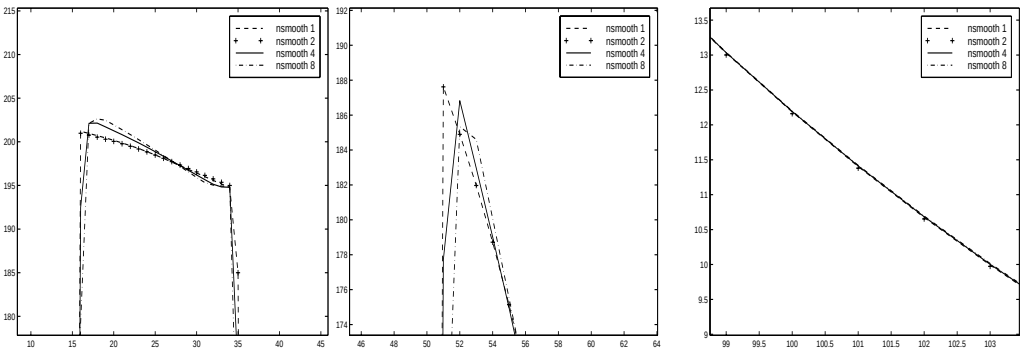


FIG. 8.8. *Details of comparison of solutions for degree of smoothing* $k = 1, 2, 4, 8$.

FIG. 8.9. *Left: original image. Right: noisy image, SNR* ≈ 4.

We analyze in the next experiment the dependence of the solution of our new model on the degree of smoothing k for the calculation of κ_2. We solve the denoising problem using convolution kernels given by (8.1) with $k = 1, 2, 4, 8$ and with optimal parameters for the noisy signal of the previous experiment. The results are overlayed in Figure 8.8 and suggest that the impact of smoothing is very little: it is imperceptible in smooth zones and edges are captured quite independently of the degree of smoothing.

The image of Figure 8.9 is used in the 2D experiment. The SNR of the noisy image is approximately 4. Details of the solutions of the denoising problems by TV and our model are shown in Figure 8.10. The contour plots of these details are shown

FIG. 8.10. *Left: image obtained by TV restoration model. Right: image obtained by our model.*

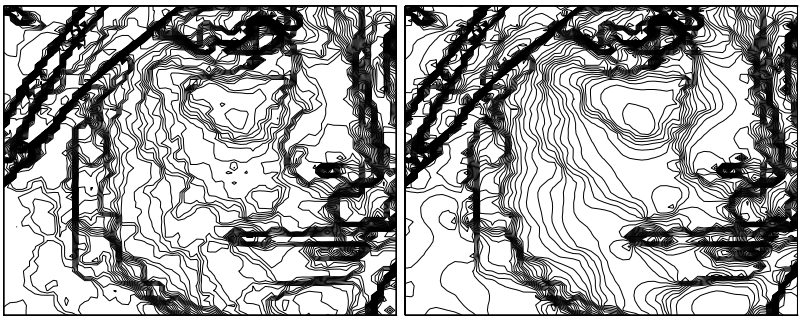


FIG. 8.11. *Left: contours of part of the image obtained by TV restoration model. Right: contours of part of the image obtained by new restoration model.*

in Figure 8.11. We can deduce from these contours that the image obtained by the new model is significantly smoother at smooth regions than the one obtained by the TV model, while the "true" edges are preserved similarly by both models. Furthermore, we can see from the contours of the TV restoration that it contains many regions of constant gray level, whereas the new model tends to preserve some small details better, thus giving the restored image a less cartoon-like aspect.

9. Concluding remarks. The main advantage of the fourth order PDE model that we have presented in this paper with respect to the original TV model is that it allows smooth transitions without penalizing sharp edges. It also shows significant robustness to the choice of the parameter μ. A disadvantage of the new model is that it has two parameters instead of one, although some heuristics have been applied to choose the additional parameter μ (see section 7). Another disadvantage is that the mathematical problem is much more challenging. In principle, the uniqueness of the solution of the nonlinear fourth order PDE that we propose is not ensured. Nevertheless, we have observed that for sound choices of the parameter μ, the fixed point method that we use to solve this equation converges to the same solution, regardless of the initial approximation. However, although this fixed point scheme has shown significant convergence robustness in our experiments, it does not show the monotonic decay of the norm of the equation that appears when applied to the TV restoration problem (see [8]). We will pursue more sophisticated and faster alternatives to this method in our future research.

REFERENCES

[1] R. Acar and C. R. Vogel, *Analysis of total variation penalty methods for ill-posed problems*, Inverse Problems, 10 (1994), pp. 1217–1229.

[2] G. Aubert and L. Vese, *A variational method in image recovery*, SIAM J. Numer. Anal., 34 (1997), pp. 1948–1979.

[3] P. Blomgren and T. Chan, *Modular Solvers for Constrained Image Restoration Problems*, CAM report 97-52, Math Department, University of California at Los Angeles, Los Angeles, CA, November 1997.

[4] P. Blomgren, T. F. Chan, and P. Mulet, *Extensions to total variation denoising*, in Proceedings Society of Photo-Optical Instrumentation Engineers (SPIE) 97, San Diego, 1997.

[5] F. Catté, P.-L. Lions, J.-M. Morel, and T. Coll, *Image selective smoothing and edge detection by nonlinear diffusion*, SIAM J. Numer. Anal., 29 (1992), pp. 182–193.

[6] A. Chambolle and P.-L. Lions, *Image recovery via total variation minimization and related problems*, Numer. Math., 76 (1997), pp. 167–188.

[7] T. Chan, G. Golub, and P. Mulet, *A nonlinear primal-dual method for total variation-based image restoration*, SIAM J. Sci. Comput., 20 (1999), pp. 1964–1977.

[8] T. F. Chan and P. Mulet, *On the convergence of the lagged diffusivity fixed point method in image restoration*, SIAM J. Numer. Anal., 36 (1999), pp. 354–367.

[9] D. Geman and G. Reynolds, *Constrained restoration and the recovery of discontinuities*, IEEE Trans. Pattern Anal. Mach. Intel., 14 (1992), pp. 367–383.

[10] C. W. Groetsch, *The Theory of Tikhonov Regularization for Fredholm Integral Equations of the First Kind*, Pitman, Boston, 1984.

[11] B. Gustafsson, H. O. Kreiss, and J. Oliger, *Time Dependent Problems and Difference Methods*, Wiley-Interscience, New York, 1995.

[12] L. Rudin, S. Osher, and E. Fatemi, *Nonlinear total variation based noise removal algorithms*, Phys. D, 60 (1992), pp. 259–268.

[13] A. N. Tikhonov and V. Y. Arsenin, *Solutions of Ill-Posed Problems*, John Wiley, New York, 1977.

[14] L. Vese, *Variational Problems and PDE's for Image Analysis and Curve Evolution*, Ph.D. thesis, University of Nice, France, 1996.

[15] C. R. Vogel and M. E. Oman, *Iterative methods for total variation denoising*, SIAM J. Sci. Comput., 17 (1996), pp. 227–238.

SIAM J. SCI. COMPUT.
Vol. 22, No. 2, pp. 517–534
© 2000 Society for Industrial and Applied Mathematics

A ROBUST ALGORITHM FOR OPTIMIZATION WITH GENERAL EQUALITY AND INEQUALITY CONSTRAINTS*

XIN-WEI LIU† AND YA-XIANG YUAN†

Abstract. An algorithm for general nonlinearly constrained optimization is presented, which solves an unconstrained piecewise quadratic subproblem and a quadratic programming subproblem at each iterate. The algorithm is robust since it can circumvent the difficulties associated with the possible inconsistency of QP subproblem of the original SQP method. Moreover, the algorithm can converge to a point which satisfies a certain first-order necessary optimality condition even when the original problem is itself infeasible, which is a feature of Burke and Han's methods [*Math. Programming*, 43 (1989), pp. 277–303]. Unlike Burke and Han's methods, our algorithm does not introduce additional bound constraints. The algorithm solves the same subproblems as the Han–Powell SQP algorithm at feasible points of the original problem. Under certain assumptions, it is shown that the algorithm coincides with the Han–Powell method when the iterates are sufficiently close to the solution. Some global convergence results are proved and locally superlinear convergence results are also obtained. Preliminary numerical results are reported.

Key words. SQP algorithm, constrained optimization, convergence

AMS subject classifications. 65K10, 95C30, 90C45

PII. S1064827598334861

1. Introduction.

We consider the optimization problem with general equality and inequality constraints

$$\min f(x) \tag{1.1}$$

$$\text{subject to (s.t.) } c_i(x) = 0, \quad i \in E, \tag{1.2}$$

$$c_i(x) \geq 0, \quad i \in I, \tag{1.3}$$

where $f(x) : R^n \to R$ and $c_i(x) : R^n \to R (i \in E \cup I)$ are continuously differentiable functions, $E = \{1, 2, \ldots, m_e\}$, $I = \{m_e + 1, \ldots, m\}$, m_e and m are two positive integers, and $m \geq m_e$.

SQP algorithms for constrained optimization are iterative and generate a new approximate to the solution by the procedure

$$x^+ = x + sd, \tag{1.4}$$

where x is the current point, d is a search direction which minimizes a quadratic model subject to linearized constraints, and s is the step-length along the direction such as [8, 13, 21]. For $k \geq 1$, the original SQP method developed by Wilson, Han, and Powell solves the following QP subproblem

$$\min g_k^T d + \frac{1}{2} d^T B_k d \tag{1.5}$$

$$\text{s.t. } c_i(x_k) + \nabla c_i(x_k)^T d = 0, \quad i \in E, \tag{1.6}$$

$$c_i(x_k) + \nabla c_i(x_k)^T d \geq 0, \quad i \in I, \tag{1.7}$$

*Received by the editors March 2, 1998; accepted for publication (in revised form) March 14, 2000; published electronically July 25, 2000. This research was partially supported by Chinese NSF grants 19525101, 19731010, and by National 9-5 key project 96-221-04-02-02.
http://www.siam.org/journals/sisc/22-2/33486.html

†LSEC, Institute of Computational Mathematics and Scientific/Engineering Computing, Chinese Academy of Sciences, P.O. Box 2719, Beijing, 100080, China (lxw@lsec.cc.ac.cn, yyx@lsec.cc.ac.cn).

at the kth iterate, where $g_k = \nabla f(x_k)$ is the gradient of the objective function and B_k is an estimate of the Hessian of the Lagrangian function

$$L(x, \lambda) = f(x) - \sum_{i=1}^{m} \lambda_i c_i(x), \tag{1.8}$$

and $(\lambda_1, \lambda_2, \ldots, \lambda_m)^T$ is a multiplier vector approximation.

Because of its nice convergence properties (for example, see [8, 13, 14] and [1]), the SQP method has been attracting the attention of many researchers. It has been extended to problems other than optimization [12, 20].

The requisite consistency of the linearized constraints of the QP subproblem (1.5)–(1.7) is a serious limitation of the SQP method. Within the framework of the SQP method, Powell suggests solving a modified subproblem at each iterate (see [13, 19]):

$$\min\ g_k^T d + \frac{1}{2} d^T B_k d + \frac{1}{2}\delta_k (1-\mu)^2 \tag{1.9}$$

$$\text{s.t. } \mu c_i(x_k) + \nabla c_i(x_k)^T d = 0, \quad i \in E, \tag{1.10}$$

$$\mu_i c_i(x_k) + \nabla c_i(x_k)^T d \geq 0, \quad i \in I, \tag{1.11}$$

where

$$\mu_i = \begin{cases} 1, & c_i(x_k) > 0 \\ \mu, & c_i(x_k) \leq 0 \end{cases}$$

and $0 \leq \mu \leq 1$, $\delta_k > 0$ is a penalty parameter. With some other technique, the computational investigation provided by Schittkowski [17, 18] shows that this modification works very well.

A simple example presented by Burke and Han [3] and Burke [2] indicates this approach may not be the best one. Assume there are two constraints on R:

$$c_1(x) = 1 - e^x = 0, \tag{1.12}$$

$$c_2(x) = x = 0, \tag{1.13}$$

with any objective function $f(x)$ on R. For any infeasible point $x \neq 0$, the linearized constraints are inconsistent, and the only solution of the modified constraints (1.10)–(1.11) is $\mu = 0$ and $d = 0$. Although this example is too specialized to make a general claim, it shows that the problem caused by the inconsistency of the linearized constraints can not always be solved by using (1.10)–(1.11).

Based on a trust region strategy, Fletcher [6, 7] developed the Sl_1QP method for (1.1)–(1.3). Fletcher's approach solves the following QP subproblem at the kth iteration:

$$\min\ g_k^T d + \frac{1}{2} d^T B_k d + \delta_k ||(c(x_k) + \nabla c(x_k)^T d)_-||_1 \tag{1.14}$$

$$\text{s.t. } ||d||_\infty \leq \beta_k, \tag{1.15}$$

where $c(x_k) = (c_1(x_k), \ldots, c_m(x_k))^T$, $(c(x_k) + \nabla c(x_k)^T d)_- \in R^m$ with

$$(c_i(x_k) + \nabla c_i(x_k)^T d)_- = c_i(x_k) + \nabla c_i(x_k)^T d, \quad i \in E, \tag{1.16}$$

$$(c_i(x_k) + \nabla c_i(x_k)^T d)_- = \min(0, c_i(x_k) + \nabla c_i(x_k)^T d), \quad i \in I, \tag{1.17}$$

δ_k is a penalty parameter, β_k is a positive constant. It has been shown that under certain assumptions the search direction generated by (1.14)–(1.15) is locally identical to that by (1.5)–(1.7).

Burke and Han [3] show that Fletcher's approach is still incomplete. One of the reasons is that the search direction may point away from the optimal point.

Similar to the method of Sahba [16], Burke and Han [3] and Burke [2] present an approach to overcome difficulties associated with the inconsistency of the QP subproblem (1.5)–(1.7). Their methods are also similar to the methods of Powell [13] and Fletcher [6, 7]. A feature different from the other methods is that even when (1.1)–(1.3) is itself infeasible their methods can converge to a point which meets a certain first-order necessary optimality condition. However, Burke and Han's method is conceptual.

In this paper, we describe an implementable algorithm which is a modification to the SQP method. Our motivation is to explore further techniques for overcoming the inconsistency of the QP subproblem to derive an efficient reliable SQP algorithm. The line search direction of our algorithm consists of two directions: one is computed by solving a special nonsmooth l_1QP subproblem that depends on only active constraints defined by an active technique; another is obtained by solving a simplified QP problem which is always feasible even when the QP subproblem of the standard SQP method is infeasible. Our algorithm is a generalization of the algorithm presented by Liu and Yuan [10], which is also similar to Burke and Han's method [3]. However, unlike their method, we do not introduce additional bound constraints. Our algorithm obtains a direction which can be a nonzero descent direction of the merit function even if (1.5)–(1.7) is infeasible. At a feasible point of (1.1)–(1.3), the algorithm solves the same subproblem as (1.5)–(1.7). Moreover, under certain assumptions, our algorithm generates the same iterates as the Han–Powell method. Some global convergence results are proved and locally superlinear convergence is derived.

Our algorithm can be easily combined with the trust region approach. Thus, the algorithm can be extended to a trust region algorithm for optimization with general constraints.

The paper is organized as follows. We present our algorithm in section 2. The stationary properties of the algorithm are given in section 3. In section 4 some global convergence results are proved. We discuss the local properties of the algorithm in section 5. In section 6, some preliminary numerical results are reported.

2. The algorithm. Define the penalty function associated with (1.1)–(1.3),

$$\phi(x, r) = f(x) + r||c(x)_-||, \tag{2.1}$$

where $||\cdot||$ is any given convex norm on R^m, $r > 0$ is a penalty parameter, and $c(x)_- \in R^m$ with

$$c_i(x)_- = c_i(x), \quad i \in E, \tag{2.2}$$

$$c_i(x)_- = \min(0, c_i(x)), \quad i \in I. \tag{2.3}$$

It is straightforward to see that $||c(x)_-|| = 0$ if and only if x is a feasible point of (1.1)–(1.3). If the norm $||\cdot||$ is the l_1 norm, (2.1) is the l_1 exact penalty function, which is also a merit function employed by Han [8] and Powell [13, 14]. Throughout this paper if the norm is not specified, it is the same as that used in (2.1).

Define the index sets

$$I_k = \{i \in I : c_i(x_k) \le 0\}, \tag{2.4}$$

$$\bar{I}_k = \{i \in I : c_i(x_k) > 0\}, \tag{2.5}$$

$$J_k = I_k \cup E. \tag{2.6}$$

These index sets are related to the current iterate x_k and can be identified easily. Under some assumptions, we will show that J_k tends to be the index set of the active constraints of (1.1)–(1.3).

Our algorithm solves two subproblems at each iterate: one is an unconstrained piecewise quadratic subproblem (see [13, 21]), and the other is a quadratic programming subproblem. At the kth iteration the unconstrained subproblem has the following form:

$$\min_{d \in R^n} \psi_k(d) = \frac{1}{2} d^T B_k d + r_k ||(c_{J_k}(x_k) + \nabla c_{J_k}(x_k)^T d)_-||, \tag{2.7}$$

where B_k positive definite is an estimate of the Lagrangian Hessian of (1.1)–(1.3), $c_{J_k}(x_k) \in R^{|J_k|}$ is a vector whose components are $c_i(x_k)(i \in J_k)$, $|J_k|$ is the cardinality of the index set J_k, and r_k is the penalty parameter. Let d_{k1} be the solution of (2.7). If x_k is feasible, we have $d_{k1} = 0$. If $d_{k1} \neq 0$, d_{k1} is a descent direction of $\phi(x_k, r_{k+1})$ for sufficiently large r_{k+1}. Moreover, there is a $\tau_k \in (0, 1]$ such that $c_i(x_k) + \delta \nabla c_i(x_k)^T d_{k1} \geq 0$ for all $\delta \in [0, \tau_k]$ and $i \in \bar{I}_k$. In fact, we can let $\tau_k = \min\{1, \hat{\tau}_k\}$, where

$$\hat{\tau}_k = \min\{-c_i(x_k)/(\nabla c_i(x_k)^T d_{k1}) : \quad i \in \bar{I}_k \text{ and } \nabla c_i(x_k)^T d_{k1} < 0\}. \tag{2.8}$$

Let $\hat{c}_i(x_k) = c_i(x_k) + \nabla c_i(x_k)^T \tau_k d_{k1}$ for $i \in \bar{I}_k$. We generate d_{k2} by solving the QP subproblem

$$\min\ g_k^T d + \frac{1}{2} d^T B_k d \tag{2.9}$$

$$\text{s.t. } \nabla c_i(x_k)^T d = 0, \quad i \in E, \tag{2.10}$$

$$\nabla c_i(x_k)^T d \geq 0, \quad i \in I_k, \tag{2.11}$$

$$\hat{c}_i(x_k) + \nabla c_i(x_k)^T d \geq 0, \quad i \in \bar{I}_k, \tag{2.12}$$

and let $d_k = \tau_k d_{k1} + d_{k2}$ be the search direction. It will be shown that d_k is a descent direction for the penalty function where the penalty parameter is updated automatically. Therefore, (2.1) can be employed as a merit function to force the global convergence of the algorithm.

The updating of penalty parameter for the SQP approach is important. In order to obtain the global convergence, Han [8] and Powell [13] require that

$$r \geq ||\lambda_k||_\infty \tag{2.13}$$

for all $k \geq 1$, where λ_k is an estimate of the Lagrangian multiplier vector at x_k. However, (2.13) is generally replaced by some updating procedure when practically implementing an SQP algorithm because we do not know any information about the multiplier vector of (1.8). Similar to Powell [13] and Burke and Han [3], a penalty parameter updating procedure is employed in our algorithm. Since d_{k2} is not related to the constraint violation, the object of updating the penalty parameter is to force d_{k1} to be a descent direction of (2.1). Thus, at the kth iteration we let r_k remain unchanged if d_{k1} is a descent direction; otherwise, r_k is increased in the following way:

$$r_{k+1} = \max\left\{2r_k + \rho, \quad \frac{g_k^T d_{k1} + d_{k1}^T B_k d_{k1}}{||(c_{J_k})_-|| - ||(c_{J_k} + \nabla c_{J_k}^T d_{k1})_-||}\right\}, \tag{2.14}$$

where ρ is a positive number.

Now we can state our algorithm as follows.

ALGORITHM 2.1 (a robust algorithm for optimization).

Step 1 [Step 0.] Given the initial approximate x_0, an $n\times n$ symmetric positive definite matrix B_0, an initial penalty parameter $r_0 > 0$, and some positive scalars ρ, β and μ, where $\beta < 1$ and $\mu < \frac{1}{2}$; $k = 0$;

If the stopping criterion is satisfied, stop;
Solve subproblem (2.7) *to generate d_{k1} and subproblem* (2.9)–(2.12) *to generate d_{k2};*

Step 2. *Update penalty parameter. If*

$$g_k^T d_{k1} + \frac{1}{2} d_{k1}^T B_k d_{k1} + r_k(||(c_{J_k}(x_k) + \nabla c_{J_k}(x_k)^T d_{k1})_-|| - ||(c_k(x_k))_-||) \le 0, \tag{2.15}$$

let $r_{k+1} = r_k$; Otherwise, r_k is updated by (2.14).

Step 3. $d_k = \tau_k d_{k1} + d_{k2}$. Select the smallest positive integer s such that

$$\phi(x_k + \beta^s d_k, r_{k+1}) - \phi(x_k, r_{k+1}) \le \mu\beta^s(g_k^T d_k + r_{k+1}(||(c(x_k) + \nabla c(x_k)^T d_k)_-|| - ||(c(x_k))_-||)). \tag{2.16}$$

Let $t_k = \beta^s$ and $x_{k+1} = x_k + t_k d_k$;

Step 4. Generate B_{k+1}. Set $k = k + 1$ and goto Step 1.

The stopping criterion is not given in the algorithm. Generally, $||d_k||_2 = 0$ can be used as the stopping criterion. Since no assumption on regularity of the constraints is made, it is possible that d_k does not tend to zero for $k \to \infty$. Thus, we use the condition $||x_{k+1} - x_k||_2 = 0$ as the stopping criterion. In practical implementation, a positive tolerance number will be introduced.

Algorithm 2.1 is similar to the methods proposed by Burke and Han [3] and Burke [2]. Since no additional bound constraints are employed, the algorithm can be implemented in the same way as SQP algorithms.

It should be noted that our algorithm solves the same subproblem as (1.5)–(1.7) at a feasible point of (1.1)–(1.3).

Two examples presented by Burke and Han [3] can help us to understand the above algorithm and the differences between our algorithm and Burke and Han's methods.

Example 2.2. The constraint function $c : R \to R^2$ has the form

$$c(x) = \begin{pmatrix} 1 - e^x \\ x \end{pmatrix} \tag{2.17}$$

and $m_e = m = 2$. The norm is the l_1 norm.

For this problem, (2.7) has the form

$$\min_{d\in R} \quad \frac{1}{2} B_k d^2 + r_k(|1 - e^x - e^x d| + |x + d|). \tag{2.18}$$

For any $x_k = x \neq 0$, by direct calculations, $d_{k2} = 0$ and

$$d_{k1} = \begin{cases} e^{-x} - 1 \text{ or } -\frac{r_k}{B_k}(e^x + 1) & \text{if } x > 0, \\ \frac{r_k}{B_k}(e^x + 1) \text{ or } -x & \text{if } x < 0, \\ = 0 & \text{if } x = 0. \end{cases} \tag{2.19}$$

It is easily found that d_{k1} has the following properties:

$$d_{k1} \begin{cases} > 0 & \text{for } x < 0, \\ < 0 & \text{for } x > 0, \\ = 0 & \text{for } x = 0. \end{cases} \tag{2.20}$$

From (2.20), it is easy to see that our algorithm will converge to the solution $x = 0$ from any starting point.

Example 2.3. The constraint function $c : R \to R^2$ is given by

$$c(x) = \begin{pmatrix} -x^2 - 1 \\ -x \end{pmatrix} \tag{2.21}$$

and $m_e = 0$, $m = 2$. Any problem with $c(x)$ as its constraint is infeasible as $c_1(x) = -x^2 - 1 = 0$ has no solution. Let the norm be the l_1 norm.

For constraints (2.21), we have that

$$d_{k1} = \begin{cases} \max\left(-\frac{2r_k}{B_k}x,\ -\frac{x^2+1}{2x}\right) & \text{if } x < 0, \\ -x,\ -\frac{x^2+1}{2x},\ -\frac{r_k(2x+1)}{B_k} \text{ or } -\frac{2r_k}{B_k}x & \text{if } 0 < x < 1, \\ -x,\ -\frac{x^2+1}{2x},\ -\frac{r_k}{B_k} \text{ or } -\frac{r_k}{B_k}(2x+1) & \text{if } x > 1, \\ \max\left(-\frac{3r_k}{B_k},\ -1\right) & \text{if } x = 1, \\ 0 & \text{if } x = 0, \end{cases} \tag{2.22}$$

and

$$d_{k2} \begin{cases} = 0 & \text{if } x < 0, \\ \le 0 & \text{if } x \ge 0. \end{cases} \tag{2.23}$$

Thus, the search direction generated by our algorithm always points toward the origin, of which the image under c is the closest point to R^2_+ for the l_1 norm.

Algorithm 2.1 can also solve the problem (8.1) of Burke and Han [3] successfully since $d_{k2} = 0$ and d_{k1} directs to the optimal solution for any iteration point $x \neq 0$.

3. Stationary properties of the algorithm. Examples 2.2 and 2.3 display some properties of Algorithm 2.1. These properties are favorable in practice because much information, such as consistency for (1.1)–(1.3), is not known beforehand. Since no restrictions are imposed on the constraint functions, a cluster point of the sequence generated by our algorithm can be one of three different types of points. Similar to Yuan [23], we give their definitions and their stationary properties.

DEFINITION 3.1. *$x \in R^n$ is called*

(1) *a strong stationary point of* (1.1)–(1.3) *if x is feasible and there exists a vector $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_m)^T \in R^m$ such that*

$$g(x) - \sum_{i=1}^{m} \lambda_i \nabla c_i(x) = 0, \tag{3.1}$$

$$\lambda_i \ge 0, \quad \lambda_i c_i(x) = 0, \qquad i \in I; \tag{3.2}$$

(2) *an infeasible stationary point of* (1.1)–(1.3) *if x is infeasible and*

$$\min_{d \in R^n} ||(c(x) + \nabla c(x)^T d)_-|| = ||(c(x))_-||; \tag{3.3}$$

(3) *a singular stationary point of* (1.1)–(1.3) *if* x *is feasible and there exists an infeasible sequence* $\{v_k\}$ *converging to* x *such that*

$$\lim_{k\to\infty} \frac{\min_{d\in R^n} ||(c(v_k) + \nabla c(v_k)^T d)_-||}{||(c(v_k))_-||} = 1. \tag{3.4}$$

Definition 3.1 is related to our algorithm closely. It should be noted that there are some differences between our definition and that of [23], for example, the definition on the singular stationary point.

A strong stationary point defined above is precisely a $K-T$ point of (1.1)–(1.3). If $||(c(x_k))_-|| = 0$ and $d_{k2} = 0$, by the first-order $K-T$ condition of (2.9)–(2.12), x_k is a strong stationary point of (1.1)–(1.3).

Throughout this report, we make the following assumption.

Assumption 3.2. (1) $f(x)$ and $c_i(x)$, $i \in E \cup I$, are twice continuously differentiable functions; (2) the approximation B_k of the Lagrangian Hessian is positive definite and there exists two positive constants M_1 and M_2 such that

$$M_1||d||_2^2 \le d^T B_k d \le M_2||d||_2^2 \tag{3.5}$$

holds for all $d \in R^n$ and all $k \ge 1$.

LEMMA 3.3. *The following statements hold:*

(i) *If* (3.3) *holds at* x_k, *then* $d = 0$ *solves* (2.7) *uniquely;*

(ii) *if* $\{x_k\}$ *and* $\{r_k\}$ *are bounded, then* $\{d_{k1}\}$ *is also bounded.*

Proof. (i) For any $d \ne 0$, by (3.3), there exists $t > 0$ sufficiently small such that

$$\begin{aligned} \psi_k(td) &= (1/2)t^2 d^T B_k d + r_k||(c_{J_k}(x_k) + \nabla c_{J_k}(x_k)^T (td))_-|| \\ &= (1/2)t^2 d^T B_k d + r_k||(c(x_k) + \nabla c(x_k)^T (td))_-|| \\ &\ge (1/2)t^2 d^T B_k d + r_k||(c(x_k))_-|| > \psi_k(0). \end{aligned} \tag{3.6}$$

Because $\psi_k(d)$ is convex, we can see that $d = 0$ is the unique solution of (2.7).

(ii) The definition of d_{k1} shows that

$$\begin{aligned} \psi_k(0) &\ge \psi_k(d_{k1}) \\ &\ge (1/2)M_1||d_{k1}||_2^2 + r_k||(c_{J_k}(x_k) + \nabla c_{J_k}(x_k)^T d_{k1})_-|| \\ &\ge (1/2)M_1||d_{k1}||_2^2. \end{aligned} \tag{3.7}$$

Therefore,

$$||d_{k1}||_2^2 \le (2/M_1)\psi_k(0) = (2/M_1)r_k||(c(x_k))_-||. \qquad \square \tag{3.8}$$

LEMMA 3.4. *If* $x \in R^n$ *is an infeasible stationary point or a singular stationary point as defined above, then there exist* $\lambda_0 \ge 0$ *and* $\lambda \in R^m$ *such that the first-order necessary optimality condition*

$$\lambda_0 g(x) - \sum_{i=1}^{m} \lambda_i \nabla c_i(x) = 0, \tag{3.9}$$

$$\lambda_i \ge 0, \quad i \in I, \tag{3.10}$$

holds.

Proof. Suppose that $d(x)$ minimizes the unconstrained problem

$$\min_{d\in R^n} \frac{1}{2}d^T Bd + ||(c(x) + \nabla c(x)^T d)_-|| \tag{3.11}$$

at the iteration point x, where B is any positive definite matrix. Then, the first-order optimality condition at x gives that

$$Bd + \nabla c(x)\mu(x) = 0, \tag{3.12}$$

$$\mu(x) \in \partial||u||\big|_{u=(c(x)+\nabla c(x)^T d)_-}, \tag{3.13}$$

where $\mu(x) \in R^m$. It follows directly from (3.13) that $(\mu(x))_i \le 0$ for $i \in I$.

If x is an infeasible stationary point, similar to the proof of Lemma 3.3, we have that $d(x) = 0$. Let $\lambda_0 = 0$ and $\lambda_i = -(\mu(x))_i$, which gives (3.9).

Now suppose that x is a singular stationary point, $\{x_k : k \in K\}$ is a subsequence, and $x_k \to x$ for $k \to \infty (k \in K)$. Suppose that $d(x_k)$ is a solution of (3.11) at x_k; then (3.12)–(3.13) holds at x_k and

$$\begin{aligned}\min_{d\in R^n} ||(c(x_k) + \nabla c(x_k)^T d)_-|| - ||(c(x_k))_-|| \\ \le -\frac{1}{2}d(x_k)^T Bd(x_k) \le 0.\end{aligned} \tag{3.14}$$

Combining (3.4), we have

$$\lim_{k\to\infty, k\in K} \frac{d(x_k)^T Bd(x_k)}{||c(x_k)_-||} = 0. \tag{3.15}$$

Thus, for $k \in K$,

$$\lim_{k\to\infty} ||d(x_k)|| = 0. \tag{3.16}$$

It follows from (3.16) and (3.12) that

$$\lim_{k\to\infty, k\in K} \nabla c(x_k)\mu(x_k) = 0. \tag{3.17}$$

Because $||\mu(x_k)||_0 \le 1$ for all k (where $||\cdot||_0$ is the dual norm of $||\cdot||$), there is a cluster point $\mu^* \in R^m$ with $(\mu^*)_i \le 0$ for $i \in I$. We see that (3.9) holds if we let $\lambda_0 = 0$ and $\lambda_i = -(\mu^*)_i$ for $i \in E \cup I$. This completes our proof. □

4. Global convergence. First we show that if our algorithm stops after finite many iterations, the last iterate point must be a strong stationary point or an infeasible stationary point of (1.1)–(1.3).

LEMMA 4.1. *Suppose that d_{k1} is a solution of* (2.7) *and d_{k2} solves* (2.9)–(2.12). *If $d_{k1} = 0$ and $d_{k2} = 0$, then x_k is either a strong stationary point or an infeasible stationary point of* (1.1)–(1.3).

Proof. If $d_{k1} = 0$ and $d_{k2} = 0$, it follows from the first-order Kuhn–Tucker condition of (2.9)–(2.12) that there exists $\lambda_k \in R^m$ such that

$$g_k - \nabla c(x_k)\lambda_k = 0, \tag{4.1}$$

$$(\lambda_k)_i c_i(x_k) = 0 \quad \text{for } i \in \bar{I}_k, \tag{4.2}$$

$$(\lambda_k)_i \ge 0 \quad \text{for } i \in I. \tag{4.3}$$

If $||(c(x_k))_-|| = 0$, then by (4.1)–(4.3) and Definition 3.1(1), x_k is a strong stationary point of (1.1)–(1.3).

Suppose that $||(c(x_k))_-|| \neq 0$. We want to prove that (3.3) holds for x_k. If it is not the case, then there exist $\tilde{d}_k \neq 0$ and $0 < \tau_k \leq 1$ such that

$$\min_{d \in R^n} ||(c(x_k) + \nabla c(x_k)^T d)_-|| = ||(c(x_k) + \nabla c(x_k)^T \tilde{d}_k)_-|| < ||(c(x_k))_-|| \tag{4.4}$$

and

$$||(c(x_k) + \nabla c(x_k)^T (\tau_k \tilde{d}_k))_-|| = ||(c_{J_k}(x_k) + \nabla c_{J_k}(x_k)^T (\tau_k \tilde{d}_k))_-||. \tag{4.5}$$

Let $\hat{d}_k = \tau_k \tilde{d}_k$; then it follows that

$$r_k(||(c_{J_k}(x_k))_-|| - ||(c_{J_k}(x_k) + \nabla c_{J_k}(x_k)^T \hat{d}_k)_-||) \leq \frac{1}{2} \hat{d}_k^T B_k \hat{d}_k. \tag{4.6}$$

Define

$$t_0 = \frac{r_k}{2} \frac{||(c_{J_k}(x_k))_-|| - ||(c_{J_k}(x_k) + \nabla c_{J_k}(x_k)^T \hat{d}_k)_-||}{\hat{d}_k^T B_k \hat{d}_k}; \tag{4.7}$$

then by (4.6), $0 < t_0 \leq \frac{1}{4}$ and

$$\begin{aligned} &\psi_k(t_0 \hat{d}_k) - \psi_k(d_{k1}) \\ &\quad \leq \frac{1}{2} t_0^2 \hat{d}_k^T B_k \hat{d}_k + r_k t_0 \{||(c_{J_k}(x_k) + \nabla c_{J_k}(x_k)^T \hat{d}_k)_-|| - ||(c_{J_k}(x_k))_-||\} \\ &\quad \leq \frac{3}{4} t_0 \tau_k r_k \{||(c(x_k) + \nabla c(x_k)^T \tilde{d}_k)_-|| - ||(c(x_k))_-||\} < 0, \end{aligned} \tag{4.8}$$

which gives a contradiction. □

The following result shows that the line search procedure is well defined in the algorithm.

LEMMA 4.2. *Suppose that at least one of d_{k1} and d_{k2} is nonzero; then τ_k is defined by* (2.8). *Then $\tau_k d_{k1} + d_{k2}$ is a descent direction of the penalty function* (2.1) *and the line search condition* (2.17) *is well defined.*

Proof. Let $q(x) = ||c(x)_-||$; then by Lemma 4.1 of Burke and Han [4],

$$q'(x; d) \leq ||(c(x) + \nabla c(x)^T d)_-|| - ||c(x)_-||. \tag{4.9}$$

Define $d_k = \tau_k d_{k1} + d_{k2}$; then

$$\phi'(x_k, r_{k+1}; d_k) \leq g_k^T d_k + r_{k+1}(||(c(x_k) + \nabla c(x_k)^T d_k)_-|| - ||c(x_k)_-||). \tag{4.10}$$

By (2.9)–(2.12) and the convexity of the norm,

$$\begin{aligned} &||(c(x_k) + \nabla c(x_k)^T d_k)_-|| - ||c(x_k)_-|| \\ &\quad \leq \tau_k(||(c_{J_k}(x_k) + \nabla c_{J_k}(x_k)^T d_{k1})_-|| - ||c(x_k)_-||). \end{aligned} \tag{4.11}$$

Thus,

$$\begin{aligned} \phi'(x_k, r_{k+1}; d_k) &\leq g_k^T d_{k2} + \tau_k \{g_k^T d_{k1} \\ &\quad + r_{k+1}(||(c_{J_k}(x_k) + \nabla c_{J_k}(x_k)^T d_{k1})_-|| - ||c(x_k)_-||)\}. \end{aligned} \tag{4.12}$$

It follows from Step 2 of the algorithm that

$$\phi'(x_k, r_{k+1}; d_k) \leq -\frac{1}{2} d_{k2}^T B_k d_{k2} - \frac{1}{2}\tau_k d_{k1}^T B_k d_{k1} < 0. \tag{4.13}$$

Now we prove that the line search condition (2.17) is well defined. By the mean value theorem, for any $t > 0$, there exists $\alpha \in (0, t)$ such that

$$f(x_k + td_k) - f(x_k) = tg(x_k + \alpha d_k)^T d_k. \tag{4.14}$$

Similarly there exist $\alpha_i \in (0, t)$ such that

$$c_i(x_k + td_k) - c_i(x_k) = t\nabla c_i(x_k + \alpha_i d_k)^T d_k. \tag{4.15}$$

Define $A_k = (\nabla c_1(x_k + \alpha_1 d_k), \nabla c_2(x_k + \alpha_2 d_k), \ldots, \nabla c_m(x_k + \alpha_m d_k))$; then

$$\begin{aligned}\phi(x_k + td_k, r_{k+1}) - \phi(x_k, r_{k+1}) &\leq tg(x_k + \alpha d_k)^T d_k \\ &\quad + tr_{k+1}(||(c(x_k) + A_k^T d_k)_-|| - ||c(x_k)_-||).\end{aligned} \tag{4.16}$$

Since

$$||(c(x_k) + A_k^T d_k)_-|| - ||(c(x_k) + \nabla c(x_k)^T d_k)_-|| \leq ||(A_k - \nabla c(x_k))^T d_k||, \tag{4.17}$$

it follows from the first part of the proof that there always exists a sufficiently small $t_0 > 0$ such that for all $t \in (0, t_0)$, $\alpha \in (0, t)$,

$$\begin{aligned}&(g(x_k + \alpha d_k) - g_k)^T d_k + r_{k+1}(||(A_k - \nabla c(x_k))^T d_k||) \\ &\quad + (1 - \mu)(g_k^T d_k + r_{k+1}(||(c(x_k) + \nabla c(x_k)^T d_k)_-|| - ||c(x_k)_-||)) < 0,\end{aligned} \tag{4.18}$$

which completes the proof. □

Assumption 4.3. $\{x_k\}$ and $\{d_k\}$ are uniformly bounded.

The assumption on $\{x_k\}$ is common in analyses on convergence of the algorithms. Since the objective function (2.9) is coercive, and $d = 0$ is feasible for (2.10)–(2.12), d_{k2} is bounded. If $r_k \to \infty$, in place of (2.7), we use the following subproblem:

$$\min \frac{1}{2} d^T B_k d + r_k ||(c_{J_k}(x_k) + \nabla c_{J_k}(x_k)^T d)_-|| \tag{4.19}$$

$$\text{s.t. } ||d||_2 \leq R, \tag{4.20}$$

where $R > 0$ is a constant, and all analyses still hold since the norm is convex.

If $r_k \to \infty$, by Lemma 4.2 of [23], $\lim_{k\to\infty} ||c(x_k)_-||$ exists.

LEMMA 4.4. *If $r_k \to \infty$ and $\lim_{k\to\infty} ||c(x_k)_-|| \neq 0$, then there exists a convergent subsequence of $\{x_k\}$ which converges to an infeasible stationary point of* (1.1)–(1.3).

Proof. Let S be the set of the accumulation points of $\{x_k\}$. If the lemma is not true, for any $x \in S$, $||c(x)_-|| \neq 0$ and (3.3) does not hold. Thus, there exists a $v > 0$ such that for k large enough,

$$\min_{||d||_2 \leq \delta} ||(c(x_k) + \nabla c(x_k)^T d)_-|| \leq ||c(x_k)_-|| - v, \tag{4.21}$$

where δ is a positive constant.

Let $\hat{d}_k$ be a vector such that $\|\hat{d}_k\| \leq \delta$ and that

$$||(c(x_k) + \nabla c(x_k)^T \hat{d}_k)_-|| = \min_{||d||_2 \leq \delta} ||(c(x_k) + \nabla c(x_k)^T d)_-||. \tag{4.22}$$

The fact that $\|\hat{d}_k\| \leq \delta$,

$$||(c_{J_k}(x_k) + \nabla c_{J_k}(x_k)^T \hat{d}_k)_-|| \leq ||(c(x_k) + \nabla c(x_k)^T \hat{d}_k)_-||, \tag{4.23}$$

$r_k \to \infty$, and that d_{k1} solves (2.7) implies that inequality

$$\begin{aligned} &g_k^T d_{k1} + \frac{1}{2} d_{k1}^T B_k d_{k1} + r_k(||(c_{J_k}(x_k) + \nabla c_{J_k}(x_k)^T d_{k1})_-|| - ||(c(x_k))_-||) \\ &\leq g_k^T d_{k1} + \frac{1}{2} \hat{d}_k^T B_k \hat{d}_k + r_k(||(c_{J_k}(x_k) + \nabla c_{J_k}(x_k)^T \hat{d}_k)_-|| - ||(c(x_k))_-||) \\ &\leq g_k^T d_{k1} + \frac{1}{2} M\delta^2 - r_k v < 0 \end{aligned} \tag{4.24}$$

holds for all sufficiently large k, which contradicts the parameter updating procedure. □

Similarly, we have the following result.

LEMMA 4.5. *If $r_k \to \infty$ and $\lim_{k\to\infty} ||c(x_k)_-|| = 0$, then there exists a convergent subsequence of $\{x_k\}$ which converges to a singular stationary point of* (1.1)–(1.3).

Proof. Let x be any accumulation point of $\{x_k\}$. Then x is a feasible point of (1.1)–(1.3). The condition $r_k \to \infty$ implies that there exists an infinite subsequence $\{x_k : k \in \mathcal{K}\}$ such that $||(c(x_k))_-|| \neq 0$ for $k \in \mathcal{K}$.

If the result is not true, then for any convergent subsequence $\{x_k : k \in \tilde{K}\}(\tilde{K} \subset \mathcal{K})$, (3.4) does not hold. Hence, there exists a positive number v such that (4.21) holds. Similar to Lemma 4.4, the proof can be completed. □

The above two lemmas imply that r_k is bounded if no subsequence of $\{x_k\}$ converges to an infeasible stationary point or a singular stationary point of (1.1)–(1.3).

LEMMA 4.6. *Suppose that $r_k = r$ (r is a positive constant) for all k large enough, $\{x_k\}$ is an infinite sequence, and $\{x_k : k \in \hat{K}\}$ is a convergent subsequence. Then $d_k \to 0$ for $k \in \hat{K}$ and $k \to \infty$.*

Proof. We proceed by contradiction. Without loss of generality, assume that $r_k = r$ for all k.

Suppose that there exist an infinite subset $K' \subset \hat{K}$ and a positive constant η such that $||d_k||_2 \geq \eta$ for $k \in K'$. By Lemma 4.2, there exists $\hat{\eta} > 0$ such that

$$\nabla_t \phi(x_k + td_k, r)|_{t=0} \leq -\hat{\eta} < 0. \tag{4.25}$$

Thus, there exists a constant $\sigma > 0$ and sufficiently small $t_k > 0$ such that for $k \in K'$,

$$\phi(x_k + t_k d_k, r) \leq \phi(x_k, r) - \sigma. \tag{4.26}$$

The above inequality implies that

$$\Sigma_{k\in K'}(\phi(x_k + t_k d_k, r) - \phi(x_k, r)) \leq -\Sigma_{k\in K'}\sigma = -\infty, \tag{4.27}$$

which is a contradiction. This completes the proof. □

In the following theorem, we assume that (d_{k2}, λ_k) is a Kuhn–Tucker pair of (2.9)–(2.12) at x_k, where $\lambda_k \in R^m$ is a Lagrange multiplier vector associated with d_{k2}.

THEOREM 4.7. *Suppose that $\{x_k\}$ is an infinite sequence generated by the algorithm, $\{r_k\}$ and $\{\lambda_k\}$ are bounded, and $\{x_k : k \in \hat{K}\}$ is a subsequence converging to x^*. If $||c(x^*)_-|| = 0$, then x^* is a strong stationary point of* (1.1)–(1.3).

Proof. Since (d_{k2}, λ_k) is a Kuhn–Tucker pair of (2.9)–(2.12) at x_k, we have

$$g_k + B_k d_{k2} - \nabla c(x_k)\lambda_k = 0, \tag{4.28}$$

$$(\lambda_k)_i \nabla c_i(x_k)^T d_{k2} = 0 \text{ for } i \in I_k, \tag{4.29}$$

$$(\lambda_k)_i(\hat{c}_i(x_k) + \nabla c_i(x_k)^T d_{k2}) = 0 \text{ for } i \in \bar{I}_k, \tag{4.30}$$

$$(\lambda_k)_i \geq 0 \text{ for } i \in I, \tag{4.31}$$

and (2.10)–(2.12) hold. Moreover, (d_{k1}, μ_{J_k}) satisfies that

$$B_k d_{k1} + r_k \nabla c_{J_k}(x_k)\mu_{J_k} = 0, \tag{4.32}$$

$$\mu_{J_k} \in \partial ||u|| \big|_{u=(c_{J_k}(x_k)+\nabla c_{J_k}(x_k)^T d_{k1})_-}, \tag{4.33}$$

where $\mu_{J_k} \in R^{|J_k|}$ is a vector with $(\mu_{J_k})_i (i \in J_k)$ as its components. It follows from (4.33) that $(\mu_{J_k})_i \leq 0$ for $i \in I_k$. Thus, we have

$$g_k + B_k d_k - \nabla c(x_k)u_k = 0, \tag{4.34}$$

$$(u_k)_i(c_i(x_k) + \nabla c_i(x_k)^T d_k) = 0 \text{ for } i \in \bar{I}_k, \tag{4.35}$$

$$(u_k)_i \geq 0 \text{ for } i \in I, \tag{4.36}$$

where

$$(u_k)_i = (\lambda_k)_i - r_k \tau_k (\mu_{J_k})_i \quad \text{for } i \in J_k, \tag{4.37}$$

and

$$(u_k)_i = (\lambda_k)_i \quad \text{for } i \in \bar{I}_k. \tag{4.38}$$

Let $\mathcal{I}(x^*) = \{i : i \in I_k \text{ for infinitely many } k \in \hat{K}\}$, $I(x^*) = \{i \in I : c_i(x^*) = 0\}$. Then $I(x^*) \supset \mathcal{I}(x^*)$.

By (4.33), $||\mu_{J_k}||_0 \leq 1$, where $||\cdot||_0$ is the dual norm of $||\cdot||$ defined by (2.1). Then it follows from Lemma 4.6 that there exists a cluster point $u^* \in R^m$ of $\{u_k\}$ such that

$$g(x^*) - \nabla c(x^*)u^* = 0, \tag{4.39}$$

$$(u^*)_i c_i(x^*) = 0 \text{ for } i \in I, \tag{4.40}$$

with $(u^*)_i \geq 0$ for $i \in I$. □

The condition on λ_k is not restrictive. The boundedness of $\{r_k\}$ implies that (2.15) holds for sufficiently large k. Thus, by (4.32)–(4.33) and (4.28), we have

$$r_k \geq ((\nabla c(x_k)^T d_{k1})^T \lambda_k)/||\nabla c(x_k)^T d_{k1}|| \tag{4.41}$$

for sufficiently large k. On the other hand, if we suppose the Mangasarian–Fromovitz condition holds at x^*, it can be proved that λ_k is bounded.

It should be noted that the above convergence results do not rely on any linear independence assumption of the gradients of the constraints. Thus, the algorithm may terminate at some iteration, which is not a Kuhn–Tucker point of (1.1)–(1.3), even if the penalty parameter is bounded. A simple example will demonstrate this case.

Example 4.8. Consider the problem

$$\min\ y_1 + (1/2)y_2^2 \tag{4.42}$$

$$\text{s.t. } (1/2)y_1^2 = 0, \tag{4.43}$$

$$y_1 + y_2^3 - 3/2 = 0. \tag{4.44}$$

Let the penalty parameter $r = 1$; the algorithm will terminate at $(1, 0)$, which is not a Kuhn–Tucker point of (4.42)–(4.44).

5. Local convergence. To study local convergence properties of the algorithm, we make the following assumption.

Assumption 5.1. (1) $x_k \to x^*$, where x^* is a Kuhn–Tucker point of (1.1)–(1.3); (2) let $I^* = \{i \in I : c_i(x^*) = 0\}$; $\nabla c_i(x^*)(i \in E \cup I^*)$ are linearly independent; (3) $r_k = r$ for $k \geq \hat{k}$, where $r > 0$ is a constant, and $\hat{k}$ is a sufficiently large positive integer.

The definitions of (2.4)–(2.5) imply that for infinitely many k, there exists a small $\epsilon > 0$ such that $c_i(x_k) \geq \epsilon$ for $i \in \bar{I}_k$. Thus, by Assumption 4.3 and (2.8), we have $\tau_k \geq \tau_0$ for infinitely many k, where $\tau_0 > 0$ is a constant.

For sufficiently large k, by definitions of (2.4)–(2.5), $\bar{I}_k \supseteq \bar{I}_{k+1}$. Thus, $I_k = I^*$ for sufficiently large k. Moreover, under Assumption 5.1, it follows from (3.8) that $||d_{k1}||_2 \to 0$ for $k \to \infty$. Therefore, $\tau_k \to 1$ for sufficiently large k.

LEMMA 5.2. *Under Assumption* 5.1, *suppose that* d_{k1} *is a solution of* (2.7) *at the point* x_k; *then there exists a sufficiently large* k', *such that for* $k \geq k'$,

$$(c_{J_k}(x_k) + \nabla c_{J_k}(x_k)^T d_{k1})_- = 0. \tag{5.1}$$

Proof. The first-order necessary condition of (2.7) imply that (4.32)–(4.33) hold and $||\mu_{J_k}||_0 \leq 1$ with $||\cdot||_0$ being the dual norm of $||\cdot||$ defined by (2.1). If at the kth iteration $(c_{J_k}(x_k) + \nabla c_{J_k}(x_k)^T d_{k1})_- \neq 0$, then $||\mu_{J_k}||_0 = 1$.

Define $p_k = ||B_k d_{k1} + r_k \nabla c_{J_k}(x_k)\mu_{J_k}||_2$ and $J^* = E \cup I^*$. If for sufficiently large k, $(c_{J_k}(x_k) + \nabla c_{J_k}(x_k)^T d_{k1})_- \neq 0$, then it follows from Lemma 4.6 and the equivalence of the norm that

$$\begin{aligned} p_k &= ||r_k \nabla c_{J^*}(x^*)\mu_{J_k}||_2 + O(||x_k - x^*||_2) + O(||d_{k1}||_2) \\ &\geq r_k ||\mu_{J_k}||_2 / ||\nabla c_{J^*}(x^*)^+||_2 + o(1) \\ &\geq c_0 r_k ||\mu_{J_k}||_0 / ||\nabla c_{J^*}(x^*)^+||_2 + o(1) \\ &\geq c_0 r / (2||\nabla c_{J^*}(x^*)^+||_2), \end{aligned} \tag{5.2}$$

where $\nabla c_{J^*}(x^*) \in R^{n \times |J^*|}$ is a matrix with $\nabla c_i(x^*)$ $(i \in J^*)$ as its volume vectors, $\nabla c_{J^*}(x^*)^+$ is its generalized inverse matrix, and $c_0 > 0$ is a constant. Equation (5.2) contradicts (4.32). This completes our proof. □

The above lemma shows that there exists a sufficient large integer k_0, such that for $k \geq k_0$, the piecewise quadratic subproblem (2.7) is equivalent to the following quadratic programming problem:

$$\min\ \frac{1}{2} d^T B_k d \tag{5.3}$$

$$\text{s.t. } c_i(x_k) + \nabla c_i(x_k)^T d = 0, \quad i \in E, \tag{5.4}$$

$$c_i(x_k) + \nabla c_i(x_k)^T d \geq 0, \quad i \in I_k. \tag{5.5}$$

Assumption 5.3. Suppose that λ^* is a Lagrangian multiplier vector associated with x^*: (1) The strict complementarity condition holds at (x^*, λ^*); (2) $\nabla^2 L(x^*, \lambda^*)$ is positive definite for all nonzero d in the null space $\{d : \ \nabla c_i(x^*)^T d = 0, \ \ i \in E \cup I^*\}$, where $L(x, \lambda)$ is defined as (1.8).

It follows from Assumption 5.3 that (x^*, λ^*) is an isolated Kuhn–Tucker pair of (1.1)–(1.3). If the conditions in Assumption 5.3 hold, then for sufficiently large k, d_{k1} derived by (5.3)–(5.5) is a solution of the problem

$$\min \ \frac{1}{2} d^T B_k d \tag{5.6}$$

$$\text{s.t. } c_i(x_k) + \nabla c_i(x_k)^T d = 0 \quad \text{for } i \in E \cup I^*, \tag{5.7}$$

and d_{k2} generated by (2.9)–(2.12) solves

$$\min \ g_k^T d + \frac{1}{2} d^T B_k d \tag{5.8}$$

$$\text{s.t. } \nabla c_i(x_k)^T d = 0 \quad \text{for } i \in E \cup I^*. \tag{5.9}$$

Let $\nabla c_{J^*}(x_k)$ be an $n \times |J^*|$ matrix with $\nabla c_i(x_k)(i \in J^*)$ as its components. By direct calculations, it follows from (5.6)–(5.7) that

$$d_{k1} = -B_k^{-1} \nabla c_{J^*}(x_k)(\nabla c_{J^*}(x_k)^T B_k^{-1} \nabla c_{J^*}(x_k))^{-1} c_{J^*}(x_k), \tag{5.10}$$

and by (5.8)–(5.9),

$$\begin{aligned} d_{k2} = {} & B_k^{-1} \nabla c_{J^*}(x_k)(\nabla c_{J^*}(x_k)^T B_k^{-1} \nabla c_{J^*}(x_k))^{-1} \nabla c_{J^*}(x_k)^T B_k^{-1} g_k \\ & - B_k^{-1} g_k. \end{aligned} \tag{5.11}$$

Thus, $d_{k1} + d_{k2}$ is a solution of the problem

$$\min \ g_k^T d + \frac{1}{2} d^T B_k d \tag{5.12}$$

$$\text{s.t. } c_i(x_k) + \nabla c_i(x_k)^T d = 0 \quad \text{for } i \in J^*. \tag{5.13}$$

The above discussion can be stated as the following lemma.

LEMMA 5.4. *If the conditions in Assumptions* 5.1 *and* 5.3 *hold, then there exists a sufficiently large* k_1, $k_1 \geq k_0$, *such that for* $k \geq k_1$, *the algorithm generates identical directions with the Han–Powell method.*

By Lemma 5.4 and the related results of the *SQP* method (for example, see [1] and [22]), the superlinear convergence of the algorithm is a direct result.

LEMMA 5.5. *Suppose that the conditions in Assumption* 5.3 *hold. If*

$$\lim_{k \to \infty} \frac{||P^*(B_k - \nabla^2 L(x^*, \lambda^*))d_k||_2}{||d_k||_2} = 0, \tag{5.14}$$

where P^* *is a projection matrix on the null space* $\{d : \ \nabla c_i(x^*)^T d = 0, \ \ i \in E \cup I^*\}$; *then*

$$\lim_{k \to \infty} \frac{||x_k + d_k - x^*||}{||x_k - x^*||} = 0. \tag{5.15}$$

A superlinear convergence step may be truncated due to the nonsmoothness of the merit function, which is known as "the Marotos effect" (for example, see [22, 24]). In order to avoid this difficulty, the second-order correction technique is considered by Mayne and Polak [11], Coleman and Conn [5], Fletcher [7], and so on. For our problem, when $||c_{J_k}(x_k)|| \leq \epsilon$ (ϵ is a prescribed number), we solve the subproblem

$$\min_{d \in R^n} \frac{1}{2} d^T B_k d + r_k ||(c_{J_k}(x_k + d_k) + \nabla c_{J_k}(x_k)^T d)_-|| \tag{5.16}$$

to generate the second-order correction step $\tilde{d}_k$. The algorithm with the second-order correction technique is presented, which is a modification to Algorithm 2.1.

ALGORITHM 5.6.

Step 1 [Step 0.] Given $x_0 \in R^n$, $B_0 \in R^{n \times n}$, $r_0 > 0$, $0 < \mu < \frac{1}{2}$, $0 < \beta < 1$, $\rho > 0$, ϵ_0, ϵ_1, ϵ_2, $\epsilon_3 > 0$, $k := 0$.

Generate J_k *and* $\bar{I}_k$ *by* (2.4)–(2.6), *solve* (2.7) *giving* d_{k1};
Calculate τ_k *by* (2.8); *Solve* (2.9) *giving* d_{k2}; $d_k = \tau_k d_{k1} + d_{k2}$;
If $||d_k|| \leq \epsilon_0$, *stop;*
If $||c_{J_k}(x_k)|| \leq \epsilon_1$ *solve* (5.16) *giving* $\tilde{d}_k$; *else* $\tilde{d}_k = 0$.

Step 2. $r_{k+1} := r_k$ *if* (2.15) *holds, otherwise compute* r_{k+1} *by* (2.14).

Step 3. $s = 0, 1, 2, \ldots$; *If*

$$\phi(x_k + \beta^s d_k + \beta^{2s} \tilde{d}_k, r_{k+1}) - \phi(x_k, r_{k+1}) \leq \mu \beta^s (g_k^T d_k + r_{k+1}(||(c(x_k) + \nabla c(x_k)^T d_k)_-|| - ||(c(x_k))_-||)). \tag{5.17}$$

Let $t_k = \beta^s$ *and* $x_{k+1} = x_k + t_k d_k + t_k^2 \tilde{d}$.

Step 4. If $||x_{k+1} - x_k|| \leq \epsilon_3$, *stop.*

Step 5. Compute the values of $f(x)$, $c(x)$, $\nabla f(x)$, *and* $\nabla c(x)$ *at* x_{k+1};
Generate B_{k+1}; $k := k + 1$ *and goto Step* 1.

Similar to the above discussion on Lemmas 5.2 and 5.4, and to the analyses in [11, 22, 24], we have the following result.

THEOREM 5.7. *Under Assumptions* 5.1 *and* 5.3, *suppose that* (5.14) *holds,* $\epsilon_i = 0 (i = 0, 1, 2, 3)$, *and* $\{x_k\}$ *is an infinite sequence generated by Algorithm* 5.6. *Then*

$$\lim_{k \to \infty} \frac{||x_k + d_k + \tilde{d}_k - x^*||}{||x_k - x^*||} = 0, \tag{5.18}$$

and there exists a sufficiently large k_2 *such that for* $k \geq k_2$, $t_k = 1$. *Thus,* $\{x_k\}$ *converges Q-superlinearly.*

6. Numerical results. A FORTRAN subroutine was programmed to test our algorithm. Our experiments were done on an SGI indigo workstation at the State Key Laboratory of Scientific and Engineering Computing, Chinese Academy of Sciences, Beijing.

The norm in (2.1) is selected to be the l_∞ norm. We solve the piecewise quadratic subproblem (2.7) by reformulating it as follows:

$$\min \frac{1}{2} w^T Q_k w + p_k^T w \tag{6.1}$$

$$\text{s.t. } y - \nabla c_i(x_k)^T d \geq c_i(x_k) \quad \text{for } i \in E, \tag{6.2}$$

$$y + \nabla c_i(x_k)^T d \geq -c_i(x_k) \quad \text{for } i \in J_k, \tag{6.3}$$

where $Q_k = \begin{pmatrix} B_k & \\ & 0 \end{pmatrix} \in R^{(n+1)\times(n+1)}$, $w = \begin{pmatrix} d \\ y \end{pmatrix} \in R^{n+1}$, $p_k = \begin{pmatrix} 0 \\ r_k \end{pmatrix} \in R^{n+1}$. Problem (6.1)–(6.3) is a convex quadratic programming problem, which is, in our algorithm, solved by Powell's subroutine ZQPCVD [15]. The second-order correction subproblem (5.16) is solved similarly.

The first test problem that we solved is taken from [16]:

$$\min x_1 x_2 \tag{6.4}$$

$$\text{s.t.} -\sin(x_1) \geq 0, \tag{6.5}$$

$$\cos(x_1) \geq 0, \tag{6.6}$$

$$-x_1^2 - x_2^2 + \pi/2 \geq 0, \tag{6.7}$$

$$x_1 + \pi \geq 0, \tag{6.8}$$

$$x_2 + \pi/2 \geq 0, \tag{6.9}$$

and the standard starting point is $x_0 = (0, 5)^T$. Sahba's algorithm terminates at the point $x^* = (0, -1.25331)^T$, which is an approximate Kuhn–Tucker point and not the approximate minimum point of (6.4)–(6.9). The other test problems are from Hock and Schittkowski [9].

In our implementation of the algorithm, a small positive tolerance number ϵ is introduced and the index sets

$$I_k(\epsilon) = \{i \in I : c_i(x_k) \leq \epsilon\}, \tag{6.10}$$

$$\bar{I}_k(\epsilon) = \{i \in I : c_i(x_k) > \epsilon\}, \tag{6.11}$$

$$J_k(\epsilon) = I_k(\epsilon) \cup E \tag{6.12}$$

are employed instead of (2.4)–(2.6).

For each problem, the standard initial point is used. We choose initial parameters $\mu = 0.1$, $\beta = 0.5$, $\rho = 1$, and $\epsilon_i = 10^{-6}$ for $i = 0, 1, 2, 3$. The choice of the initial penalty parameter is scale dependent and $r_0 = 1$ is chosen for our test problems. The initial Lagrangian Hessian estimate $B_0 = I$ and B_k is updated by the damped BFGS formula (see [14]):

$$B_{k+1} = B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{y_k y_k^T}{s_k^T y_k}, \tag{6.13}$$

where

$$y_k = \begin{cases} \hat{y}_k, & \hat{y}_k^T s_k \geq 0.2 s_k^T B_k s_k, \\ \theta_k \hat{y}_k + (1-\theta_k) B_k s_k, & \text{otherwise}, \end{cases} \tag{6.14}$$

and $\hat{y}_k = g_{k+1} - g_k + (\nabla c(x_{k+1}) - \nabla c(x_k))\lambda_k$, $s_k = x_{k+1} - x_k$, $\theta_k = 0.8 s_k^T B_k s_k / (s_k^T B_k s_k - s_k^T \hat{y}_k)$, λ_k is a multiplier associated with (2.9)–(2.12).

The test problems are also solved by Powell's subroutine VMCWD, which is a very successful algorithm for many nonlinear programming problems. The error tolerance for VMCWD is 10^{-8}.

The subroutine VMCWD failed to solve Sahba's problem (6.4)–(6.9) since the constraints seem to be inconsistent after the first iteration. The numerical results given by our algorithm are presented in Table 1, where $R - KT$ and $R - CV$ represent the l_2 norms of the gradient of the Lagrangian and the violation of the constraints,

TABLE 1

k	$x_k^{(1)}$	$x_k^{(2)}$	r_k	τ_k	t_k	$R-KT$	$R-CV$
0	0.0	5.0	1.0	1.0	1.0	5.0	2.3429E+1
1	-3.1416	2.6571	1.0	1.0	1.0	3.2415	1.5391E+1
2	-1.1116	2.3552	1.0	1.0	1.0	1.7142	5.2119
3	-0.8262	1.3834	1.0	1.0	1.0	8.8489E-1	1.0257
4	-0.9236	0.9546	1.0	1.0	1.0	4.4570E-2	1.9339E-1
5	-0.8899	0.8858	1.0	1.0	1.0	6.1008E-3	5.8572E-3
6	-0.8870	0.8854	1.0	1.0	1.0	2.4008E-3	7.7911E-6
7	-0.8863	0.8862	1.0	1.0	1.0	1.3274E-4	0.0

TABLE 2

				VMCWD		Our algorithm	
Problem	n	m_e	m	NI-NF-NG	Residual	NI-NF-NG	Residual
HS7	2	1	1	12-14-14	4.94E-08	9-18-10	3.85E-08
HS14	2	1	2	5-6-6	7.90E-11	4-5-5	2.98E-07
HS22	2	0	2	5-7-7	3.18E-10	23-46-24	4.58E-08
HS38	4	0	8	81-104-104	8.96E-04	38-64-39	6.68E-05
HS43	4	0	3	12-15-15	5.14E-06	12-23-13	4.48E-06
HS52	5	3	3	5-9-9	2.21E-05	16-21-17	2.43E-12
HS63	3	2	5	8-9-9	6.72E-07	7-8-8	2.41E-07
HS76	4	0	7	5-6-6	1.45E-04	6-7-7	2.12E-07
HS86	5	0	15	4-6-6	1.78E-04	4-7-5	6.22E-05
HS113	10	0	8	12-17-17	6.46E-06	14-20-15	4.07E-05

respectively. The algorithm terminates at the approximate minimum point of (6.4)–(6.9).

Some numerical results for equality constrained optimization problems have been reported in Liu and Yuan [10]. It has been noticed that our algorithm can overcome the difficulties associated with the linear dependence of the gradients of the constraints, since an unconstrained subproblem is solved at each iterate.

The numerical results for other test problems are listed in Table 2. The problems are numbered in the same way as in Hock and Schittkowski [9]. For example, "HS43" is problem 43 in Hock and Schittkowski [9]. NI, NF, and NG represent the numbers of iterations, function, and gradient calculations, respectively.

The numerical results show that our algorithm is comparable to VMCWD. However, our algorithm requires slightly more function evaluations.

Acknowledgement. We would like to thank an anonymous referee for valuable comments.

REFERENCES

[1] P.T. Boggs, J.W. Tolle, and P. Wang, *On the local convergence of quasi-Newton methods for constrained optimization*, SIAM J. Control Optim., 20 (1982), pp. 161–171.

[2] J.V. Burke, *A sequential quadratic programming method for potentially infeasible mathematical programs*, J. Math. Anal. Appl., 139 (1989), pp. 319–351.

[3] J.V. Burke and S.P. Han, *A robust sequential quadratic programming method*, Math. Programming, 43 (1989), pp. 277–303.

[4] J.V. Burke and S.P. Han, *A Gauss-Newton approach to solving generalized inequalities*, Math. Oper. Res., 11 (1986), pp. 632–643.

[5] T.F. Coleman and A.R. Conn, *Nonlinear programming via an exact penalty function: Asymptotic analysis*, Math. Programming, 24 (1982), pp. 123–136.
[6] R. Fletcher, *Practical Methods for Optimization*, *Vol.* 2, *Constrained Optimization*, John Wiley and Sons, Chichester, UK, 1981.
[7] R. Fletcher, *A model algorithm for composite nondifferentiable optimization problems*, Math. Programming Stud., 17 (1982), pp. 67–76.
[8] S.P. Han, *A globally convergent method for nonlinear programming*, J. Optim. Theory Appl., 22 (1977), pp. 297–309.
[9] W. Hock and K. Schittkowski, *Test Examples for Nonlinear Programming Codes*, Lecture Notes in Econ. and Math. Systems 187, Springer-Verlag, Berlin, New York, 1981.
[10] X. Liu and Y. Yuan, *A Globally Convergent, Locally Superlinearly Convergent Algorithm for Equality Constrained Optimization*, Research Report, ICM-97-84, Institute of Computational Mathematical and Scientific/Engineering Computing, Chinese Academy of Sciences, Beijing, China.
[11] D.Q. Mayne and E. Polak, *A superlinearly convergent algorithm for constrained optimization problems*, Math. Programming Stud., 16 (1982), pp. 45–61.
[12] J.S. Pang and S.A. Gabriel, *NE/SQP: A robust algorithm for the nonlinear complementarity problem*, Math. Programming, 60 (1993), pp. 295–337.
[13] M.J.D. Powell, *A fast algorithm for nonlinearly constrained optimization calculations*, in Proceedings 1977 Dundee Biennial Conference on Numerical Analysis, G.A.Watson, ed., Springer-Verlag, Berlin, 1978, pp. 144–157.
[14] M.J.D. Powell, *The convergence of variable metric methods for nonlinear constrained optimization calculations*, in Nonlinear Programming 3, O.L. Mangasarian, R.R. Meyer, and S.M. Robinson, eds., Academic Press, New York, 1978, pp. 27–63.
[15] M.J.D. Powell, *ZQPCVX: A Fortran Subroutine for Convex Quadratic Programming*, Report DAMTP 1983/NA17, University of Cambridge, UK, 1983.
[16] M. Sahba, *Globally convergent algorithm for nonlinearly constrained optimization problems*, J. Optim. Theory Appl., 52 (1987), pp. 291–309.
[17] K. Schittkowski, *The nonlinear programming method of Wilson, Han and Powell with an augmented Lagrangian type line search function, Part* 1: *Convergence analysis*, Numer. Math., 38 (1981), pp. 83–114.
[18] K. Schittkowski, *On the convergence of a sequential quadratic programming method with an augmented Lagrangian line search function*, Math. Operationsforsch. Statist. Ser. Optim., 14 (1983), pp. 197–216.
[19] J. Stoer, *Principles of sequential quadratic programming methods for solving nonlinear programs*, in Computational Mathematical Programming, NATO Advanced Science Institute Series F: Computer and Systems Sciences 15, K. Schittkowski, ed., Springer-Verlag, Berlin, 1985, pp. 166–207.
[20] K. Taji and M. Fukushima, *A new merit function and a successive quadratic programming algorithm for variational inequality problems*, SIAM J. Optim., 6 (1996), pp. 704–713.
[21] R.B. Wilson, *A Simplicial Algorithm for Concave Programming*, Ph.D. thesis, Harvard University, Cambridge, MA, 1963.
[22] Y. Yuan, *Numerical Methods for Nonlinear Programming*, Modern Mathematics Series, Shanghai Scientific and Technical Publishers, Shanghai, China, 1993 (in Chinese).
[23] Y. Yuan, *On the convergence of a new trust region algorithm*, Numer. Math., 70 (1995), pp. 515–539.
[24] Y. Yuan and W. Sun, *Theories and methods for optimization*, Scientific Press, Beijing, 1997 (in Chinese).

SIAM J. SCI. COMPUT.
Vol. 22, No. 2, pp. 535–560
© 2000 Society for Industrial and Applied Mathematics

MODELING MICROSTRUCTURE EVOLUTION USING GRADIENT-WEIGHTED MOVING FINITE ELEMENTS*

ANDREW KUPRAT†

Abstract. Microstructure evolution, where grain boundaries evolve by mean curvature motion, is modeled in three dimensions (3-D) using gradient-weighted moving finite elements (GWMFE). To do this, we modify and extend an existing 2-D GWMFE code to create a new code GRAIN3D which makes the 3-D microstructure modeling possible. The right-hand side term which drives the GWMFE motion can be viewed as surface tension forces, that is, as the negative gradient of the surface integral of a constant energy density μ on the triangular interfacial grid. Extensions to the method include equations for the motion of tetrahedra that are conformally attached to the moving piecewise linear triangular facets which represent the GWMFE discretization of the evolving grain boundaries. We present some new regularization terms which control element quality, as well as preventing element collapse in the simulation. New capabilities for changing the mesh topology are used to keep the grid edge lengths below a maximum allowable length $h_{\max}$ and to mimic actual changes in the physical topology, such as collapse and disappearance of individual grains. Validating runs are performed on some test cases that can be analytically solved, including collapse of a spherical grain and the case of columnar microstructure. In the spherical collapse case, the GWMFE method appears to have an error in the surface area collapse rate $-\frac{dA}{dt}$ which is $\mathcal{O}((\Delta\theta)^2)$, where $\Delta\theta$ is a measure of the angular resolution of the mesh. Finally, a run is presented where a true 3-D microstructure (possessing triple lines and quadruple points in the interior and triple points on the exterior boundaries) is evolved to a "2-D" columnar microstructure and finally evolved down to a single grain.

Key words. microstructure evolution, motion by mean curvature, gradient-weighted moving finite elements, unstructured tetrahedral meshes, deforming grids, changing grid topology, front-tracking

AMS subject classifications. 65M60, 51P05, 73S10, 65M50

PII. S1064827598348374

1. Introduction.

Under close examination, a sample of metal (such as copper or aluminum used in semiconductor manufacture) possesses a microstructure wherein the sample is decomposed into separate grains. The atoms in individual grains exist in a crystal lattice and the lattice orientations of adjacent grains differ. The boundary surfaces between grains are thus areas of lattice misalignment and effectively possess an excess energy per unit grain boundary area. In the simplest approximation, this excess energy density is a constant μ per unit area for all grain boundaries. When the sample is heated, the grain boundaries become mobile and grain growth takes place. The motion resulting from the desire to minimize surface energy is *mean curvature motion* where the normal velocity of a point on a grain boundary is proportional to the mean curvature at that point [19]. Thus the grain boundaries move as if they are under the influence of a driving force proportional to the mean curvature, with motion opposed by a frictional force proportional to normal velocity (as if the grain boundaries are immersed in a uniform, isotropic viscous medium).

*Received by the editors December 23, 1998; accepted for publication (in revised form) December 31, 1999; published electronically July 25, 2000. The U.S. Government retains a nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or allow others to do so, for U.S. Government purposes. Copyright is owned by SIAM to the extent not limited by these rights. This research was supported by the Department of Energy under contract W-7405-ENG-36.
http://www.siam.org/journals/sisc/22-2/34837.html

†Theoretical Division, Group T-1, Mailstop B-221, Los Alamos National Laboratory, Los Alamos, NM 87545 (kuprat@lanl.gov).

Approaches to mean curvature motion and/or grain growth modeling have included 2-D front-tracking models [5]; Surface Evolver, a popular energy gradient descent method [2]; level sets [23]; vertex methods [6, 9]; and gradient-weighted moving finite elements (GWMFE) [1, 3, 4, 15]. 2-*D front tracking models* have been the workhorse of grain growth modeling—they have been used in many research articles and are particularly successful in the case of thin films, where the microstructure is for the most part columnar. *Surface Evolver* is a widely used code available over the web which allows the user to interactively reduce the surface energies of arbitrary surface configurations. Constant surface energy leads to mean curvature motion, but much more general energies and motions are possible. The method uses an explicit step where mesh points are advanced along either the energy gradient descent direction or the conjugate gradient direction. *Level set methods* have the advantage of automatically resolving certain types of topological changes that occur during grain evolution such as *self-pinchoff* where a single grain divides into two or more grains. However, when three or more grains intersect at a point (*triple* or *quadruple* point) or on a curve (*triple line*), the standard level set formulation fails and a nontrivial increase in algorithmic complexity is necessary to correctly model the evolution of these points of intersection [21, 25]. *Vertex methods* consider points of maximum grain intersection (triple or quadruple points) and assume the curves between these points are straight lines. These methods may produce valid statistical results on large collections of grains but cannot hope to compute detailed grain shapes.

Moving finite elements (MFE) [1, 14, 16] is a standard Galerkin finite element method of computing the solution $u(\mathbf{x}, t)$ of a time-dependent PDE

$$u_t = F(\mathbf{x}, t, u, \nabla u, \ldots),$$

with the additional novel feature that the Galerkin formalism is used to compute domain velocities for the computational grid points in addition to determining the time derivative of u at the grid points. Because this method would frequently overconcentrate computational nodes in the steep portions of the graph of u, it was necessary to devise GWMFE which deemphasized the importance of node placement in high-gradient regions by multiplying the PDE residual by the gradient-weighting factor $\frac{1}{\sqrt{1+|\nabla u|^2}} \leq 1$. It turns out that GWMFE can be viewed as a nonweighted method when the independent variables are taken to be the nodal positions on the graph of u, the residual is taken to be the norm of errors in u_t in the direction normal to the graph of u, and the inner product integrals are taken over the surface area of the graph. In fact, GWMFE is naturally seen to be a method for evolving a surface *without* the restriction that the surface be the graph of a function u and is thus a very natural way to compute mean curvature motion. Examples of GWMFE applied to normal motion by mean curvature are given in [15, 4]; other examples of GWMFE computations involving mean curvature are given in [12, 13, 18]. Other GWMFE computations and an exposition of the method from a somewhat different viewpoint are given in the now standard book by Baines [1].

Recently, we took the 2-D version of GWMFE used in [4], which represents evolving surfaces by piecewise linear triangular elements, and incorporated it into a new code GRAIN3D which extended its capabilities so that general microstructure evolution could be simulated. Specifically, a program called LaGriT [8] was used to produce 3-D models of metallic microstructure which consisted of a domain $\Omega \subseteq \mathbb{R}^3$ partitioned into grains, each of which were composed of hundreds of tetrahedra. The triangular interfaces between the grains were extracted from the 3-D model and evolved by

GRAIN3D using the GWMFE algorithm. Additional equations of motions were devised for the "extra" vertices in the 3-D model that were interior to the grains and not on the grain boundaries. Motions were determined solely by the requirement that the tetrahedra remain well-shaped (i.e., have a low aspect ratio) as the surfaces to which they are conformally attached deform under mean curvature motion. Finally, grid topology-change software was written to maintain to some degree a roughly constant spacing between grid points as the mesh deforms and to make possible physically significant topology changes, such as the collapse and disappearance of certain grains during the course of evolution.

The new capability of supporting a system of tetrahedra that is conformally attached to the moving triangular surfaces will be crucial in the future when we use these volume elements to compute ambient quantities throughout Ω (such as temperature). This will allow more detailed simulations where the medium's resistance to grain boundary movement is not assumed uniform. (Note, for example, that grain boundary mobility in fact strongly depends on temperature [19].)

In section 2, we review GWMFE from a surface evolution point of view. We present a detailed derivation of how the GWMFE right-hand side integral involving curvature is evaluated and show that this integral can be seen as the negative gradient, with respect to the nodal position vector $\mathbf{x} = (x_i, y_i, z_i)$ of the surface integral of a constant energy per unit area μ on the discretized interfaces. In section 3, we review regularization terms necessary for the GWMFE method to work efficiently, and we introduce a right-hand regularization term involving a quality force that controls element quality as well as preventing element collapse in the simulation. We also present a regularization force that prevents excessive stretching of elements, derived from a small artificial surface energy associated with each triangular facet. In section 4, we introduce additional equations for moving tetrahedra conformally attached to the GWMFE triangular interface mesh. In section 5, we touch on grid maintenance operations necessary to keep grid edge lengths below a maximum allowable $h_{\max}$. In section 6, we describe *recoloring* operations necessary to perform important topological changes in the microstructure simulation, such as the collapse and disappearance of individual grains. In section 7, we compare numerical solutions computed by GRAIN3D to analytic solutions for the cases of spherically symmetric grain collapse and columnar microstructures. In section 8, we show a numerical example where GRAIN3D evolves a truly 3-D microstructure.

2. Mean curvature motion and GWMFE.

2.1. Review of method. We use GWMFE [3, 4, 15] to move a multiply connected network of piecewise linear triangles for the modeling of deformation of 3-D grains. In one model of metallic grain growth [19], interface surfaces obey the simple equation

$$v_n = \mu K,$$

where v_n is the normal velocity of the interface, K is the curvature, and μ is the product of the excess energy per unit area and the mobility of the grain boundary. K is the sum of principle curvatures, i.e., twice the mean curvature. In this paper, we assume that μ is constant, i.e., it does not depend on the choice of materials on either side of the interface, nor does it depend on the orientation of the interface. We represent interfaces as parametrized surfaces:

$$\mathbf{x}(s_1, s_2) = \sum_{\text{nodes } j} \alpha_j(s_1, s_2)\mathbf{x}_j.$$

Here, (s_1, s_2) is the surface parametrization, the sum is over the N interface nodes, $\alpha_j(s_1, s_2)$ is the piecewise linear basis function (hat function) which is unity at node j and zero at all other interface nodes, and $\mathbf{x}_j = (x_j^1, x_j^2, x_j^3) \in \mathbb{R}^3$ is the vector position of node j.

We have that

$$\dot{\mathbf{x}}(s_1, s_2) = \sum_j \alpha_j(s_1, s_2)\dot{\mathbf{x}}_j,$$

is the velocity of the surface at the point $\mathbf{x}(s_1, s_2)$ (based upon linear interpolation of node velocities) and

$$v_n = \dot{\mathbf{x}}(s_1, s_2) \cdot \hat{\mathbf{n}} \quad (\hat{\mathbf{n}} \text{ is local surface normal}).$$

So

$$v_n = \sum_j (\hat{\mathbf{n}}\alpha_j) \cdot \dot{\mathbf{x}}_j. \tag{1}$$

In effect, we have that the $3N$ basis functions for v_n are $n_k\alpha_j$, where $\hat{\mathbf{n}} = (n_1, n_2, n_3)$. These basis functions are discontinuous piecewise linear, since the n_k are piecewise constant.

The GWMFE method is to minimize

$$\int (v_n - \mu K)^2 \, dS \tag{2}$$

over all possible values for the derivatives $\dot{\mathbf{x}}_i$. (The integral is over the surface area of the interfaces.) We thus obtain

$$\begin{aligned} 0 &= \frac{1}{2}\frac{\partial}{\partial \dot{x}_i^k} \int (v_n - \mu K)^2 \, dS, \quad 1 \le k \le 3, \quad 1 \le i \le N \\ &= \int (v_n - \mu K) n_k \alpha_i \, dS. \end{aligned}$$

Using (1), we obtain a system of $3N$ ODEs:

$$\left[\int \hat{\mathbf{n}}\hat{\mathbf{n}}^T \alpha_i \alpha_j \, dS \right] \dot{\mathbf{x}}_j = \int \mu K \hat{\mathbf{n}} \alpha_i \, dS,$$

or

$$\mathbf{C}(\mathbf{x})\dot{\mathbf{x}} = \mathbf{g}(\mathbf{x}), \tag{3}$$

where $\mathbf{x} = (x_1^1, x_1^2, x_1^3, x_2^1, \ldots, x_N^3)^T = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N)^T$ is the $3N$-vector containing the x, y, and z coordinates of all N interface nodes, $\mathbf{C}(\mathbf{x})$ is the matrix of inner products of basis functions, and $\mathbf{g}(\mathbf{x})$ is the right-hand side of inner products involving surface curvature. Since $\hat{\mathbf{n}}\hat{\mathbf{n}}^T$ is a 3×3 matrix, it is clear that $\mathbf{C}(\mathbf{x})$ has a 3×3 block structure.

The ODEs are solved with an implicit second order backwards difference variable time-step ODE solver [3]. We use generalized minimal residual (GMRES) iteration [22] with block-diagonal preconditioner to solve the linear equations arising from the Newton's method. (A more complex and robust choice of iterative method and preconditioner is used in [24] to solve systems involving indefinite Jacobians that arise when GWMFE is applied to problems involving convection.) The ODEs are scaled so that the variation of the x_j^k values is $\mathcal{O}(1)$ and the truncation error η (i.e., the maximum acceptable estimated error made in the x_j^k every time step) is usually set to 10^{-3}.

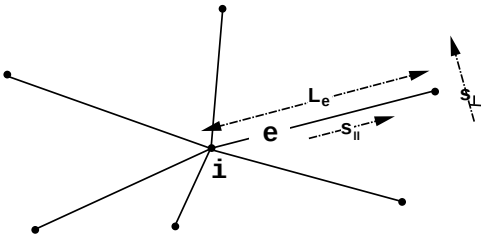


FIG. 1. *Definition of some quantities around node i, edge e.*

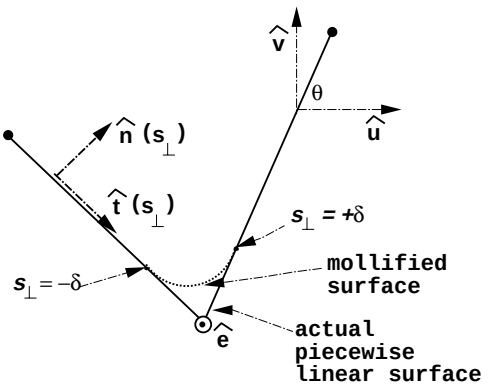


FIG. 2. *Cross-sectional view of grid. Edge e is orthogonal to page.*

2.2. Evaluation of right-hand side curvature term. The right-hand side term due to curvature,

$$\int \mu K \hat{\mathbf{n}} \alpha_i \, dS,$$

requires special consideration because on a piecewise linear manifold K is actually a distribution which is zero in the interiors of the triangles and infinite on the edges. Evaluation of this term is undertaken by mollifying (smoothing) the manifold in a small neighborhood (within a small distance δ) of the edges and then showing that $\int K \hat{\mathbf{n}} \alpha_i \, dS$ on the δ-mollified manifold tends to a limit as $\delta \to 0$ which is independent of the mollification process. This mollification interpretation is extensively explained in section 2.6 of [15], sections 2.3 and 4.10 in [3] (for 1-D), and sections 3.3 and 6.4 in [4]. For completeness we include a derivation here. Indeed, referring to Figure 1, we consider one of the edges e emanating from node i and we let $s_{||}$ be the arclength parameter running parallel to the edge and $s_\perp$ be the arclength parameter corresponding to movement on the manifold perpendicular to the edge. The length of the edge is L_e. In Figure 2, we show the intersection of the surface with a plane orthogonal to the edge e. The intersection yields a smoothed curve (due to the mollification of the surface). We assume the mollified region runs from $s_\perp = -\delta$ to $s_\perp = +\delta$. The intersection curve has tangent $\hat{\mathbf{t}}(s_\perp)$ which varies smoothly between $\hat{\mathbf{t}}(-\delta)$ and $\hat{\mathbf{t}}(+\delta)$. $\hat{\mathbf{n}}(s_\perp)$ is the normal to the surface, which smoothly varies between $\hat{\mathbf{n}}(-\delta)$ and $\hat{\mathbf{n}}(+\delta)$. We define $\hat{\mathbf{u}}$, $\hat{\mathbf{v}}$ to be unit vectors in the plane with $\hat{\mathbf{u}} \times \hat{\mathbf{v}} = \hat{\mathbf{e}}$ (the unit vector parallel

to e). θ is defined as the angle that $\hat{\mathbf{t}}(s_\perp)$ makes with $\hat{\mathbf{u}}$. Now we write

$$\begin{aligned}\int \mu K\hat{\mathbf{n}}\alpha_i \, dS &= \mu \sum_{\substack{\text{edges } e \\ \text{with } i\in e}} \int_0^{L_e} \left(\int_{-\delta}^{\delta} K\hat{\mathbf{n}}\alpha_i \, ds_\perp \right) ds_{||} \\ &\approx \mu \sum_{\substack{\text{edges } e \\ \text{with } i\in e}} \left(\int_0^{L_e} \alpha_i(s_{||}) \, ds_{||} \right) \left(\int_{-\delta}^{\delta} \frac{d\theta}{ds_\perp} \hat{\mathbf{n}} \, ds_\perp \right).\end{aligned}$$

That is, the right-hand side curvature inner product at node i is the sum of contributions from edges e incident on i. In the 2δ-wide strip near edge e, $\alpha_i(s_{||}, s_\perp)$ is nearly a piecewise linear function of only $s_{||}$ which is 1 at $s_{||} = 0$ and 0 at $s_{||} = L_e$. This means that the first integral evaluates to $\frac{1}{2}L_e$. The curvature K of the mollified surface is $\frac{d\theta}{ds_\perp}$ since there is no curvature parallel to edge e in the unmollified surface (and hence no curvature in this direction for the mollified surface as well). Now

$$\begin{aligned}\int_{-\delta}^{\delta} \frac{d\theta}{ds_\perp} \hat{\mathbf{n}} \, ds_\perp &= \int_{\theta(-\delta)}^{\theta(\delta)} \hat{\mathbf{n}} \, d\theta \\ &= \int_{\theta(-\delta)}^{\theta(\delta)} (-\sin\theta\hat{\mathbf{u}} + \cos\theta\hat{\mathbf{v}}) \, d\theta \\ &= \cos\theta\hat{\mathbf{u}} + \sin\theta\hat{\mathbf{v}} \Big|_{\theta(-\delta)}^{\theta(\delta)} \\ &= \hat{\mathbf{t}}(\delta) - \hat{\mathbf{t}}(-\delta).\end{aligned}$$

So we obtain as $\delta \to 0$ that

$$\int \mu K\hat{\mathbf{n}}\alpha_i \, dS = \mu \sum_{\substack{\text{edges } e \\ \text{with } i\in e}} \frac{1}{2} L_e(\hat{\mathbf{t}}_e^{(1)} + \hat{\mathbf{t}}_e^{(2)}),$$

where $\hat{\mathbf{t}}_e^{(1)}$, $\hat{\mathbf{t}}_e^{(2)}$ are unit normals in the piecewise linear surface which are both orthogonal to e. $\hat{\mathbf{t}}_e^{(1)}$ points into one triangle sharing e, and $\hat{\mathbf{t}}_e^{(2)}$ points into the other triangle sharing e. Now we can rewrite this sum as a sum over the triangles incident on i:

$$\int \mu K\hat{\mathbf{n}}\alpha_i \, dS = \mu \sum_{\substack{\text{triangles } k \\ \text{with } i\in k}} \frac{1}{2} \big(L_k^{(1)}\hat{\mathbf{t}}_k^{(1)} + L_k^{(2)}\hat{\mathbf{t}}_k^{(2)}\big),$$

where for each triangle k, $L_k^{(1)}$ is the length of one of the edges bordering k that contains i, $\hat{\mathbf{t}}_k^{(1)}$ is the inward normal to that edge, and $L_k^{(2)}$, $\hat{\mathbf{t}}_k^{(2)}$ are the corresponding quantities for the other edge. (See Figure 3.)

Interestingly, since $\sum_{j=1}^{3} L_k^{(j)}\hat{\mathbf{t}}_k^{(j)} = 0$ (again, see Figure 3 for the definition of $L_k^{(3)}$, $\hat{\mathbf{t}}_k^{(3)}$), we have

$$\begin{aligned}\int \mu K\hat{\mathbf{n}}\alpha_i \, dS &= -\mu \sum_{\substack{\text{triangles } k \\ \text{with } i\in k}} \frac{1}{2} L_k^{(3)}\hat{\mathbf{t}}_k^{(3)} \\ &= -\nabla_{\mathbf{x}_i} \left(\sum_{\substack{\text{triangles } k \\ \text{with } i\in k}} \mu A_k \right),\end{aligned}$$

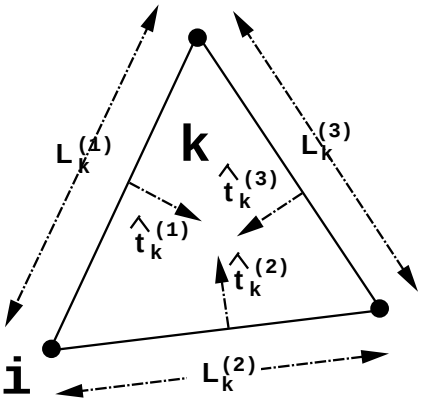


FIG. 3. *Definition of quantities around node i, triangle k.*

where A_k is the area of triangle k. If we attribute an energy of μ per unit area to the surface, then this becomes

$$\int \mu K \hat{\mathbf{n}} \alpha_i \, dS = -\nabla_{\mathbf{x}_i} E, \tag{4}$$

where

$$E = \sum_{\text{triangles } k} \mu A_k \tag{5}$$

is the total energy of the surface. This shows that the node-concentrated right-hand side force on the ith node is derivable from the negative gradient, with respect to the node position variables $\mathbf{x}_i = (x_i, y_i, z_i)$ of the surface integral of an assumed constant surface energy density of μ in our GWMFE manifold. This mirrors the physical model where mean curvature motion was derived under the assumption that the mechanism of grain growth is minimization of excess surface energy proportional to the surface integral of μ. This negative gradient force on the ith node is exactly that which would be given by a "surface tension" of magnitude μ in the planar triangular cells of our GWMFE manifold, as pointed out in [15, 4].

We note here that the form (4)–(5) for the right-hand side PDE driving terms makes conceivable computations where $\mu = \mu(\hat{\mathbf{n}})$, that is, where μ depends on the direction of the normal at the surface. In this case, one need only loop through triangles in the mesh and evaluate the quantities (4)–(5) using the orientation-dependent μ. If μ depends strongly on $\hat{\mathbf{n}}$, faceting of the surface can occur [7].

3. Regularization. In [15] it is shown how (3) represents a balance of forces on the moving GWMFE surface. The right-hand side represents driving forces from the PDE, and the left-hand side represents viscosity forces that resist these driving forces; all these forces are taken to be concentrated at the nodes. In practice, certain grid configurations can cause $\mathbf{C}(\mathbf{x})$ to be nearly singular, so that there exist essentially undamped node motions that lead to chaotic grid behavior and collapse of the time step. To alleviate this, Carlson and Miller [4] added a term (a left-hand side regularizing force) that was of the form $\mathbf{C}_1(\mathbf{x})\dot{\mathbf{x}}$ so that the total matrix $(\mathbf{C} + \mathbf{C}_1)(\mathbf{x})$ was positive definite.

In addition certain grid behaviors are possible, such as the collapse to zero of the inscribed radius of a very thin or large aspect ratio triangle, which has no physical

significance (i.e., which does not affect the shape of the surface) but which causes a catastrophic loss of time step and run termination. To alleviate these problems, small regularizing forces are added to the right-hand side of (3). The form of these forces differs from those used in [4].

3.1. Left-hand side grid viscosity forces. Suppose a node i and its neighboring nodes are all nearly coplanar. More precisely, suppose that there exists a plane P (spanned by $\hat{\mathbf{u}}$ and $\hat{\mathbf{v}}$) such that $\mathbf{x}_i$ and each member of $\{\mathbf{x}_j \mid \text{nodes } i \text{ and } j \text{ share an edge}\}$ have normal distance $\leq \eta$ to this plane. Then, since the local truncation error tolerance is η, computationally this situation is indistinguishable from the case where $\mathbf{x}_i$ and all the $\mathbf{x}_j$ exactly lie in the same plane. In this case, it is apparent that one could alter $\dot{\mathbf{x}}_i$ (or $\mathbf{x}_i$) by a plane vector $a\hat{\mathbf{u}} + b\hat{\mathbf{v}}$ without altering the normal velocity (or the shape) of the surface in the neighborhood of node i. Since (3) is derived from (2), which is a minimization that is unaffected by these additional node motions for node i, we conclude that $\mathbf{C}(\mathbf{x})$ nearly annihilates the 2-D subspace corresponding to the in-plane motions of node i. If this degeneracy or near-degeneracy is not remedied, the motion of node i will not be smooth, if it is defined at all, and the time step will decrease greatly. (This describes a *planar*-type degeneracy at node i; there are also possible *crease*-type geometrical degeneracies in which the manifold has a straight crease through node i, as in the straight ridge of a roof. See section 3.1 of [15] or section 2.4 of [4] for a further discussion and for a derivation of the following grid viscosity forces for GWMFE. Also see Chapters 6 and 7 of [1].)

To remedy such degeneracies, we add the small grid viscosity force

$$\epsilon_1 \left[\int \mathbf{I}_3 (\nabla_{\mathbf{s}} \alpha_i \cdot \nabla_{\mathbf{s}} \alpha_j) \, dS \right] \dot{\mathbf{x}}_j \equiv \mathbf{C}_1(\mathbf{x}) \dot{\mathbf{x}} \tag{6}$$

to the left-hand side of (3). Here $\mathbf{I}_3$ represents the 3×3 identity matrix, which indicates that grid viscosity is isotropic in that it treats independently and identically the x, y, and z components of the velocities $\dot{\mathbf{x}}_j$. ϵ_1 is a small number, and the gradient $\nabla_{\mathbf{s}}$ is with respect to the tangential coordinate system of the surface. We now show (6) is equal to

$$\epsilon_1 \nabla_{\dot{\mathbf{x}}_i} \frac{1}{2} \int ||\nabla_{\mathbf{s}} \dot{\mathbf{x}}||^2 \, dS. \tag{7}$$

Here $\nabla_{\dot{\mathbf{x}}_i}$ is the gradient with respect to the node velocity $\dot{\mathbf{x}}_i$ and $\dot{\mathbf{x}} = \sum \dot{\mathbf{x}}_j \alpha_j(\mathbf{s})$ is interpolated grid velocity. $\nabla_{\mathbf{s}} \dot{\mathbf{x}}$ is the gradient with respect to surface measure of interpolated grid velocity and, hence, it is a 3×2 matrix. $||\nabla_{\mathbf{s}} \dot{\mathbf{x}}||^2$ is the sum of squares of the six components (i.e., the Froebenius norm). To show (7), we have that

$$\begin{aligned}
\nabla_{\dot{\mathbf{x}}_i} \frac{1}{2} \int ||\nabla_{\mathbf{s}} \dot{\mathbf{x}}||^2 \, dS &= \nabla_{\dot{\mathbf{x}}_i} \frac{1}{2} \int \Big|\Big|\nabla_{\mathbf{s}} \sum_k \dot{\mathbf{x}}_k \alpha_k\Big|\Big|^2 \, dS \\
&= \nabla_{\dot{\mathbf{x}}_i} \frac{1}{2} \int \sum_k \sum_j (\dot{\mathbf{x}}_k \cdot \dot{\mathbf{x}}_j)(\nabla_{\mathbf{s}} \alpha_k \cdot \nabla_{\mathbf{s}} \alpha_j) \, dS \\
&= \int \sum_j \dot{\mathbf{x}}_j (\nabla_{\mathbf{s}} \alpha_i \cdot \nabla_{\mathbf{s}} \alpha_j) \, dS \\
&= \sum_j \left[\int \mathbf{I}_3 (\nabla_{\mathbf{s}} \alpha_i \cdot \nabla_{\mathbf{s}} \alpha_j) \, dS \right] \dot{\mathbf{x}}_j.
\end{aligned}$$

Thus, our regularization force is, for each node, the velocity gradient of a velocity potential and the effect of this force is to reduce $\int ||\nabla_{\mathbf{s}}\dot{\mathbf{x}}||^2\ dS$, a measure of the nonuniformity of the interpolated velocity field.

Using arguments in [3], ϵ_1 is chosen in the range

$$.25\eta^2 \leq \epsilon_1 \leq 25\eta^2, \tag{8}$$

so that the effect of the regularization term dominates only when $\mathbf{x}_i$ and the neighboring $\mathbf{x}_j$ are within η of an exact plane. If the graph is not nearly planar in this sense, the left-hand side will be dominated by $\mathbf{C}(\mathbf{x})\dot{\mathbf{x}}$, which represents viscous node resistance arising from the minimization principle (2).

3.2. Right-hand side triangle quality force. Minimization of (2) does not prevent the collapse of triangles in the moving grid; if triangle collapse (i.e., shrinking to zero of inscribed radius) is allowed to occur, the numerical run will have to be terminated. To prevent triangle collapse, a right-hand side regularizing quality force is added of the form

$$-\epsilon_2 \nabla_{\mathbf{x}_i} Q^{\text{tri}}, \tag{9}$$

where Q^{tri} is a dimensionless quality energy:

$$Q^{\text{tri}} = \sum_{\text{triangles k}} Q_k^{\text{tri}} = \sum_{\text{triangles k}} \left[\frac{(L_k^{(1)})^2 + (L_k^{(2)})^2 + (L_k^{(3)})^2}{A_k} \right]^2, \tag{10}$$

where the $L_k^{(i)}$ are edge lengths for triangle k and A_k is the area. Q_k^{tri} has a minimal value of $Q_{\text{eql}}^{\text{tri}} = 48$ for an equilateral triangle and gets larger as the aspect ratio of the triangle gets larger. That is, poorly shaped triangles have poor quality and a large quality energy. Q_k^{tri} is dimensionless and is thus independent of the scale of the triangle k. However, the force $-\nabla_{\mathbf{x}_i} Q_k^{\text{tri}}$ is not scale invariant and its magnitude increases as the scale is decreased. We can thus choose ϵ_2 to be equal to a value such that $-\nabla_{\mathbf{x}_i} Q_k^{\text{tri}}$ dominates for small triangles with shortest altitude $\leq \eta$. This will prevent triangle collapse. Indeed, for triangle k (assuming the nomenclature of Figure 3), we have that

$$\begin{aligned}
-\nabla_{\mathbf{x}_i}\epsilon_2 Q_k^{\text{tri}} &= -\epsilon_2 \nabla_{\mathbf{x}_i} \left(\frac{\sum_{j=1}^3 (L_k^{(j)})^2}{A_k} \right)^2 \\
&= -2\epsilon_2 \frac{\sum_{j=1}^3 (L_k^{(j)})^2}{A_k} \left(\frac{2\sum_{j=1}^3 L_k^{(j)} \nabla_{\mathbf{x}_i} L_k^{(j)}}{A_k} - \frac{\sum_{j=1}^3 (L_k^{(j)})^2}{A_k} \frac{\nabla_{\mathbf{x}_i} A_k}{A_k} \right) \\
&= -2\epsilon_2 Q_k^{\text{tri}} \left(\frac{2(L_k^{(1)} \nabla_{\mathbf{x}_i} L_k^{(1)} + L_k^{(2)} \nabla_{\mathbf{x}_i} L_k^{(2)})}{\sum_{j=1}^3 (L_k^{(j)})^2} - \frac{\nabla_{\mathbf{x}_i} A_k}{A_k} \right) \\
&= 2\epsilon_2 Q_k^{\text{tri}} \left(-\frac{2L_k^{(1)}}{||L_k||^2} \nabla_{\mathbf{x}_i} L_k^{(1)} - \frac{2L_k^{(2)}}{||L_k||^2} \nabla_{\mathbf{x}_i} L_k^{(2)} + \frac{\hat{\mathbf{t}}_k^{(3)}}{h_k^{(3)}} \right),
\end{aligned}$$

where $h_k^{(3)}$ is the altitude of the vertex $\mathbf{x}_i$ above the opposite edge in triangle k and $||L_k||^2$ is the square of the l_2 norm of the edge lengths. Since $-\nabla_{\mathbf{x}_i} L_k^{(1)}$ and $-\nabla_{\mathbf{x}_i} L_k^{(2)}$ are unit vectors directed along the edges in triangle k that originate at $\mathbf{x}_i$,

we have that the first two terms represent tension forces of magnitudes $4\epsilon_2 Q_k^{\text{tri}} \frac{L_k^{(1)}}{||L_k||^2}$ and $4\epsilon_2 Q_k^{\text{tri}} \frac{L_k^{(2)}}{||L_k||^2}$, respectively, which act along edges that attract node $\mathbf{x}_i$ to its neighboring nodes in triangle k. Since $\hat{\mathbf{t}}_k^{(3)}$ is a unit vector pointing normally towards node $\mathbf{x}_i$ from the edge opposite $\mathbf{x}_i$, the third term represents a repulsive force of magnitude $\frac{2\epsilon_2 Q_k^{\text{tri}}}{h_k^{(3)}}$ that prevents triangle collapse. The quality force thus causes a node to be attracted along edges and repelled along altitudes. It is zero only for the case of an equilateral triangle. If the triangle is nonequilateral, there will be a node with a shortest altitude that will feel a net repulsion from the opposite edge. If the triangle shape becomes very distorted, the node will feel a repulsive force that is arbitrarily larger than the attractive forces directed along the edges.

Assuming the case of distorted, nonequilateral triangles with shortest altitude at $\mathbf{x}_i$, we wish the dominant repulsive force be comparable to the "physically justified" driving force arising from the PDE, $-\nabla_{\mathbf{x}_i} \int \mu \, dS$, when the shortest altitude of the triangle is of order η. Thus, referring to (5), we set

$$\begin{aligned} \frac{2\epsilon_2 Q_k^{\text{tri}}}{h_k^{(3)}} &= || - \nabla_{\mathbf{x}_i} \mu A_k || \\ &= \frac{\mu L_k^{(3)}}{2}, \end{aligned}$$

so that

$$\epsilon_2 = \frac{\mu L_k^{(3)} h_k^{(3)}}{4 Q_k^{\text{tri}}}.$$

Since we are assuming $Q_k^{\text{tri}} > Q_{\text{eql}}^{\text{tri}}$ and $L_k^{(3)} > h_k^{(3)} = \mathcal{O}(\eta)$, to obtain an order of magnitude estimate for ϵ_2 we insert the trial values $Q_k^{\text{tri}} = 10 \; Q_{\text{eql}}^{\text{tri}}$ and $h_k^{(3)} = \eta$ (which implies that roughly $L_k^{(3)} = 6 \, \eta$). This leads to

$$\epsilon_2 = 3 \times 10^{-3} \, \mu \eta^2. \tag{11}$$

Certainly with $Q_k^{\text{tri}} = 10 \; Q_{\text{eql}}^{\text{tri}}$ we have that the repulsive force along the triangle altitude dominates, as assumed. If $Q_k^{\text{tri}} > 10 \; Q_{\text{eql}}^{\text{tri}}$, for triangles with shortest altitude of order η (or larger than η for sufficiently distorted triangles) the "quality force" is stronger than the physical tension force. This means that large aspect ratio triangles are prevented from forming or are "locked up," prevented from attaining an even worse shape. In practice, these needle-shaped triangles make up a small amount of the surface area of the computational interface surfaces and do not significantly affect the accuracy of the computed solutions. If $Q_k^{\text{tri}} < 10 \; Q_{\text{eql}}^{\text{tri}}$, the quality force is weaker and for Q_k^{tri} approaching $Q_{\text{eql}}^{\text{tri}}$ we have that the attractive forces along edges begin to cancel the repulsive force until at $Q_k^{\text{tri}} = Q_{\text{eql}}^{\text{tri}}$ there is no force whatsoever. Thus the force (9)–(10) *does* allow for the collapse of triangles *if they remain perfectly equilateral* during collapse. However, in practice, there is always some asymmetry in the computational runs, and so (9) with choice (11) is indeed effective at preventing triangle collapse. In fact, ϵ_2 is chosen to be somewhat less than the estimate (11) since we have found that the choice of ϵ_2 (and of the other regularization coefficients ϵ_i in this paper) is not critical and we can err on the side of introducing *less* artificial regularization forces.

3.3. Right-hand side perimeter-dependent surface energy. The opposite of grid collapse is excessive stretching of triangular elements. If edges exceed a certain maximum acceptable edge length $h_{\max}$, then the GWMFE method is running on too coarse a discretization, degrading accuracy of the results. Even if the initial grid has all edges $\leq h_{\max}$ in length without explicit intervention, it is quite possible that some edges exceed $h_{\max}$ in length after some grid evolution. To discourage excessive grid stretching, we add a right-hand side surface energy that is proportional to the perimeter of the triangles:

$$\begin{aligned} -\nabla_{\mathbf{x}_i}\epsilon_3 E^{\text{perim}} &= -\nabla_{\mathbf{x}_i}\left(\epsilon_3 \sum_{\text{triangles k}} E_k^{\text{perim}}\right) \\ &= -\nabla_{\mathbf{x}_i}\left(\epsilon_3 \sum_{\text{triangles k}} (L_k^{(1)} + L_k^{(2)} + L_k^{(3)})A_k\right). \end{aligned} \tag{12}$$

This corresponds to an artificial energy density $\mu_3 \equiv \epsilon_3 p_k$, where $p_k = L_k^{(1)} + L_k^{(2)} + L_k^{(3)}$ is the triangle perimeter. This force grows in strength as the perimeter gets large, and thus excessive stretching of individual elements will be avoided. Since it is highly likely that most edges will be near $h_{\max}$ in length, the artificial energy density will constantly be affecting the solution, introducing errors throughout the grid. (This is as opposed to the triangle quality force which will only affect the solution near "hot spots" where the grid has "thin" elements nearing collapse.) Errors in surface velocities due to (12) will be roughly $\mathcal{O}(\mu_3/\mu)$ throughout the grid, so we want this quantity to be less than some tolerance. On our $\mathcal{O}(1)$ grid, η represents a tolerance on fractional error in node position. It is reasonable to also take this as fractional error in surface velocity, and so using $\mu_3 \approx \epsilon_3(3h_{\max})$, we obtain

$$\epsilon_3 \lesssim \frac{\eta\mu}{3h_{\max}}. \tag{13}$$

4. Enlargement of system to move noninterface nodes. System (3) is integrated using a second-order implicit variable time step ODE solver [3]. However, it gives velocities of the interface nodes only ($\dot{\mathbf{x}} = (\,\dot{x}_j^k\,)_{,1\leq l\leq N}^{,1\leq k\leq 3}$), and so the system must be enlarged to include velocities for M interior nodes that are not part of the interface. That is, we extend $\mathbf{x}$ to

$$\mathbf{x} = \begin{pmatrix} \mathbf{x}_{interface} \\ \mathbf{x}_{interior} \end{pmatrix},$$

where $\mathbf{x}_{interface} = (\,x_j^k\,)_{,1\leq l\leq N}^{,1\leq k\leq 3}$, and $\mathbf{x}_{interior} = (\,x_j^k\,)_{,N+1\leq l\leq N+M}^{,1\leq k\leq 3}$. With this extension, we enlarge system (3) to be order $N+M$.

Since interface physics only tells us how to evolve the N interface nodes, we must "artificially" construct the extra elements in the enlarged $\mathbf{C}(\mathbf{x})$, $\mathbf{g}(\mathbf{x})$ to allow for orderly (tetrahedra orientation preserving) evolution of the mesh. That is, given a physically meaningful method of evolving the triangular interfaces, we are free to develop auxiliary equations for moving the tetrahedra (some of which are conformally attached to the triangular interfaces) with the only requirements being that these equations lead to efficient solution of the system (3) and that they maintain positive orientation of tetrahedra.

A natural choice for the additional equations is to generalize the left-hand side regularization force (6) to viscously damp volume grid motions, and to generalize the right-hand side quality force (9)–(10) to seek well-shaped tetrahedra.

4.1. Volume grid viscosity force. To the left-hand side of (3), we add the following contribution:

$$\epsilon_4 \left[\int \mathbf{I}_3 (\nabla \tilde{\alpha}_i \cdot \nabla \tilde{\alpha}_j)\, dV \right] \dot{\mathbf{x}}_j \equiv \mathbf{C}_4(\mathbf{x}) \dot{\mathbf{x}}. \tag{14}$$

Here, $\nabla \tilde{\alpha}_j = \nabla \tilde{\alpha}_j(\mathbf{x})$ is a piecewise linear hat function defined for $\mathbf{x} \in \mathbb{R}^3$. $\tilde{\alpha}_j$ is 1 on the jth node in the tetrahedral mesh and zero on all other nodes. $\nabla \tilde{\alpha}_j$ is the gradient of $\tilde{\alpha}_j$ over the tetrahedra (i.e., $\nabla = \nabla_{\mathbf{x}}$, where $\mathbf{x} \in \mathbb{R}^3$.) The integral is taken over the whole volume domain $\Omega \subseteq \mathbb{R}^3$ which has been partitioned into tetrahedra. This is a generalization of the surface grid viscosity term (6). Similar to (6)–(7), (14) is easily seen to be equal to

$$\epsilon_4 \nabla_{\dot{\mathbf{x}}_i} \frac{1}{2} \int ||\nabla \dot{\mathbf{x}}||^2 \, dV, \tag{15}$$

where $\dot{\mathbf{x}} = \sum_j \dot{\mathbf{x}}_j \tilde{\alpha}_j(\mathbf{x})$ is interpolated volume mesh velocity. As in the case of surface grid viscosity, the effect of (14) is to reduce nonuniformities in volume grid velocity. So, for example, if a collection of noninterface nodes were surrounded by a set of interface nodes all moving at constant velocity $\mathbf{a}$, the solution of the system

$$\begin{aligned} \mathbf{C}_4 \, \dot{\mathbf{x}} &= 0, \\ \dot{\mathbf{x}}_i &= \mathbf{a} \quad \text{at interface nodes} \end{aligned}$$

would be $\dot{\mathbf{x}} = \mathbf{a}$ at noninterface nodes as well. This is clear, since in this case $\int ||\nabla \dot{\mathbf{x}}||^2 \, dV$ is zero and is thus minimized.

To choose a suitable value for ϵ_4, we compare (6) and (14) and arrange for these terms to be roughly of the same order. Suppose node i is on the interfacial triangular network. We choose ϵ_4 so that the diagonal elements of the matrices $\mathbf{C}_1$ and $\mathbf{C}_4$ are roughly the same. That is,

$$\epsilon_4 \int \nabla \tilde{\alpha}_i \cdot \nabla \tilde{\alpha}_i \, dV \approx \epsilon_1 \int \nabla_{\mathbf{s}} \alpha_i \cdot \nabla_{\mathbf{s}} \alpha_i \, dS.$$

If we assume that the altitudes of the volume elements sharing node i are about the same magnitude as the altitudes of the surface elements sharing node i, then the integrands of the two integrals are roughly the same. Thus we should have

$$\begin{aligned} \epsilon_4 &\approx \epsilon_1 \int \nabla_{\mathbf{s}} \alpha_i \cdot \nabla_{\mathbf{s}} \alpha_i \, dS \Big/ \int \nabla \tilde{\alpha}_i \cdot \nabla \tilde{\alpha}_i \, dV \\ &\approx \epsilon_1 \int_{|\mathbf{x}-\mathbf{x}_i| \le h} dS \Big/ \int_{|\mathbf{x}-\mathbf{x}_i| \le h} dV \\ &= \frac{3}{4} \frac{1}{h} \epsilon_1, \end{aligned}$$

where h is the length of a typical edge in the mesh near node i. We approximate h by $h_{\max}$, since in practice many edges are at (or near) the longest allowed edge length in the computation. Thus, using (8), we arrive at the guideline,

$$.2\eta^2 / h_{\max} \le \epsilon_4 \le 20\eta^2 / h_{\max}. \tag{16}$$

4.2. Volume quality force. Similar to section 3.2, to prevent tetrahedral collapse and inversion as the mesh evolves, to the right-hand side of (3) we add the tetrahedral quality force

$$-\epsilon_5 \nabla_{\mathbf{x}_i} Q^{\text{tet}}, \tag{17}$$

where Q^{tet} is the dimensionless quality energy

$$Q^{\text{tet}} = \sum_{\text{tets } p} Q_p^{\text{tet}} = \sum_{\text{tets } p} \frac{\sum_{n=1}^{6} (L_p^{(n)})^2 \sum_{n=1}^{4} (A_p^{(n)})^2}{V_p^2}, \tag{18}$$

where the $L_p^{(n)}$ are the edge lengths, the $A_p^{(n)}$ are the face areas, and V_p is the volume of tetrahedron p. Q_p is a dimensionless quality measure which has the minimal value $Q_{\text{eql}}^{\text{tet}} = 324$ if p is a regular (equilateral) tetrahedron but approaches infinity if p has a worsening aspect ratio. We note that Q_k^{tri} (10) and Q_p^{tet} are both equivalent to $||L||^2||\frac{1}{H}||^2$, the square of the l_2 norm of the edge lengths of the element multiplied by the square of the l_2 norm of the inverse altitudes of the element. They are thus dimensionless element quality measures which are extremely smooth (i.e., require no square root evaluations).

The argument for setting ϵ_5 is similar to that for setting ϵ_2. Suppose one face k of a nearly collapsed tetrahedron p is on an interface. Suppose node i is on this face. We choose ϵ_5 so the force on this node due to the quality force begins to dominate for small nonregular tetrahedra with shortest altitude $\leq \eta$. Analogous to the triangle quality force of section 3.2, node i is attracted along the edges of tetrahedron p emanating from the node (terms involving $\nabla_{\mathbf{x}_i} L_p^{(n)}$), node i is attracted along the altitudes of the faces emanating from the node (terms involving $\nabla_{\mathbf{x}_i} A_p^{(n)}$), and node i is repelled along the altitude from the face opposite the node (the term involving $\nabla_{\mathbf{x}_i} V_p$). The quality force is zero only for the case of a regular tetrahedron. If the tetrahedron is nonregular, there will be a node with a shortest altitude that will feel a net repulsion from the opposite face. If the tetrahedron shape becomes very distorted, the node will feel a repulsive force that is arbitrarily larger than the attractive forces directed along the edges and faces. So assuming that the shortest altitude (and hence greatest quality force) occurs at $\mathbf{x}_i$, we have

$$\begin{aligned} -\epsilon_5 \nabla_{\mathbf{x}_i} Q_p^{\text{tet}} &= -\epsilon_5 \nabla_{\mathbf{x}_i} \frac{\sum_{n=1}^{6} (L_p^{(n)})^2 \sum_{n=1}^{4} (A_p^{(n)})^2}{V_p^2} \\ &\approx 2\epsilon_5 \frac{\sum_{n=1}^{6} (L_p^{(n)})^2 \sum_{n=1}^{4} (A_p^{(n)})^2}{V_p^2} \frac{\nabla_{\mathbf{x}_i} V_p}{V_p} \\ &= \frac{2\epsilon_5 Q_p^{\text{tet}}}{h_p^i} \hat{\mathbf{t}}_p^i, \end{aligned}$$

where h_p^i is the altitude of point $\mathbf{x}_i$ above the opposite face in tetrahedron p, and $\hat{\mathbf{t}}_p^i$ is a unit vector pointing along the altitude (towards $\mathbf{x}_i$). As in section 3.2, we wish the magnitude of this force to be equal to the magnitude of the physical force $-\nabla_{\mathbf{x}_i} \mu A_k$ due to surface tension over the interface triangle k of tetrahedron p. Again using the nomenclature of Figure 3, we have

$$\frac{2\epsilon_5 Q_p^{\text{tet}}}{h_p^i} = || - \nabla_{\mathbf{x}_i} \mu A_k || = \frac{\mu L_k^{(3)}}{2}.$$

So

$$\epsilon_5 = \frac{\mu L_k^{(3)} h_p^i}{4 Q_p^{\text{tet}}}.$$

To obtain an order of magnitude estimate for ϵ_5, we insert the trial values $Q_p^{\text{tet}} = 10\, Q_{\text{eql}}^{\text{tet}}$ and $h_p^i = \eta$, which implies that roughly $L_k^{(3)} = 5\,\eta$ (see below). This leads to

$$\epsilon_5 \approx 4 \times 10^{-4} \mu \eta^2. \tag{19}$$

We justify the assignment $L_k^{(3)}/h_p^i = 5$ when $Q_p^{\text{tet}} = 10\, Q_{\text{eql}}^{\text{tet}}$ as follows. We assume face k is roughly equilateral (the force in section 3.2 presuming to have prevented poor aspect ratio triangles on the interface). So say $L_k^{(1)} = L_k^{(2)} = L_k^{(3)} = L$. But we are assuming tetrahedron p has a short altitude h_p^i at node $\mathbf{x}_i$ pointing away from interface triangle k into the volume. Given the classification of nearly degenerate tetrahedra in [11, p. 286], tetrahedron p either has a single short edge at $\mathbf{x}_i$ or no short edges at all. In both cases, a rough calculation shows $L/h_p^i \approx 5$ when $Q_p^{\text{tet}} = 10\, Q_{\text{eql}}^{\text{tet}}$. The remarks at the end of section 3.2 (which discuss the cases $Q < 10$ and $Q > 10$ for triangles) are analogously true here for tetrahedra. Again, these are only order of magnitude estimates which are adequate for setting regularization force coefficients.

4.3. Tetrahedron lock-up. The grid forces acting on tetrahedra move the grid by (1) acting to minimize nonuniformities in grid velocity, and (2) acting to continually improve grid element aspect ratios. The artificial grid forces have the effect of necessarily overriding physically justified node movement when it is necessary to prevent inversion of tetrahedra. For instance, if a tetrahedron p has all 4 nodes on an interface, the motion given by (3) might cause the tetrahedron to invert, especially if the interface changes its sense of curvature. By adding the quality force (17) and (18), the tetrahedron will lock up when the shortest altitude is close to η and will be prevented from inverting. The effect of this on the simulation is acceptable with regard to accuracy since the lock up of a small number of tetrahedra simply removes a small fraction of numerical degrees of freedom from the simulation. The effect is further reduced if the locked up tetrahedra are effectively removed by merge and swap operations mentioned in the next section.

4.4. Equation summary. After adding regularization and tetrahedral dynamics terms to (3), the complete system of ODEs that we solve in our implementation of GWMFE is

$$\left[\int \left(\hat{\mathbf{n}}\hat{\mathbf{n}}^T \alpha_i \alpha_j + \mathbf{I}_3 \epsilon_1 (\nabla_{\mathbf{s}} \alpha_i \cdot \nabla_{\mathbf{s}} \alpha_j) \right) dS \; + \; \int \mathbf{I}_3 \epsilon_4 (\nabla \tilde{\alpha}_i \cdot \nabla \tilde{\alpha}_j)\, dV \right] \dot{\mathbf{x}}_j$$

$$= \int \mu K \hat{\mathbf{n}} \alpha_i \, dS \; - \; \nabla_{\mathbf{x}_i} (\epsilon_2 Q^{\text{tri}} + \epsilon_3 E^{\text{perim}} + \epsilon_5 Q^{\text{tet}}) \, , \quad 1 \le i \le N + M. \tag{20}$$

Here the system has been written as $N + M$ 3-vector equations, one associated with each of the N interface nodes and M interior nodes. The left-hand side is an implied

sum over all the nodes j in the mesh. The "artificial" terms involving $\epsilon_1, \ldots, \epsilon_5$ and guidelines for setting $\epsilon_1, \ldots, \epsilon_5$ are explained in (6), (8), (9)–(11), (12), (13), (14), (16), (17)–(19). For the left-hand side, if either i or j is not an interface node, then the surface integral is not defined and does not contribute to the left-hand side. On the right-hand side of (20), if i is not an interface node, then only the term involving Q^{tet} is defined and contributes to the right-hand side.

5. Grid maintenance operations. As the surfaces move and grains deform, mesh maintenance and mesh optimization tools are used to ensure good element quality and to ensure grid edges do not stretch beyond the allowable maximum length $h_{\max}$. Primitive grid operations provided by periodic calls to LaGriT, the Los Alamos Grid Toolbox [8], provide a basis for mesh maintenance and optimization. The *merge* primitive accepts as input lists of pairs of neighboring nodes: merge candidate nodes and survivor nodes. If the merge is completed, only one of the pair survives and the mesh connectivity is repaired to reflect that one node has been deleted. Before the merge takes place, LaGriT verifies that the merge will not cause tetrahedra to become inverted and that the node types and surface constraints of the survivor and merged nodes will lead to a legal merge. The *refine* primitive used in these simulations adds nodes at the midpoints of selected edges. LaGriT sets the node type and surface constraints of the added nodes by determining if the added node is in the interior, on a material interface, or on an exterior boundary. The grid connectivity is repaired to include the new elements created by connecting the new node to the other vertices of the elements which contained the refined edge. Depending on what is desired by the user, the *recon* primitive swaps connections to either improve a measure of the geometric quality of the elements (closely related to Q_p^{tet} in (18)) or to maximize the number of elements satisfying the familiar "Delaunay" criterion.

The three primitives, refine, merge, and recon, are combined into the LaGriT grid optimization operation called *massage*. (Massage is similar to the algorithm presented in [10].) Massage accepts three arguments: a creation edge length, an annihilation edge length, and a damage tolerance. (In our numerical runs, we set these three tolerances to .3, .3, and .01, respectively.) Refinement is carried out such that no edge in the grid has length greater than the creation edge length. The refine primitive is invoked using a version of an algorithm due to Rivara [20]. In the algorithm, an edge marked for refinement is placed on a stack. The algorithm then checks that the elements containing the marked edge have no other edges longer than the marked edge. If longer edges are encountered, they are placed on the stack ahead of the marked edge. The process continues recursively. The refinement candidates are then popped off the stack and refined, resulting in a refinement pattern proceeding from the longest edges to the shortest; this pattern is desirable since it usually does not degrade element quality. Massage may attempt to merge pairs of points only if the resulting grid has no edge length greater than the annihilation length. The damage tolerance specifies the amount of change to the shape of a material interface that is permissible. If a node that sits on a material interface is merged out, the interface will become flatter at that point. If the distance from the merged node's original position to its projected position on the flattened interface is greater than the damage tolerance, the merge will not be allowed. Since *merge* and (less commonly) *refine* can produce poorly shaped tetrahedra, then *recon* is used to restore well-shaped elements.

6. Mesh response to grain topology changes: Recoloring. As grain boundaries move, topology changes must be detected and the mesh must be modified to

reflect these topological changes. The topology changes are detected by assembling and monitoring the rate of change of sets of topological components. To detect grain collapse, we assemble sets of connected elements of the same material. For interface surface collapse we assemble sets of connected interface triangles between two materials; for boundary surface detachment, we assemble sets of connected boundary triangles that lie on a given boundary surface; for triple line collapse where a line is surrounded by three or more materials, we assemble sets of connected edges. We monitor the rate of collapse of these sets, and when a collapse or detachment is imminent, the mesh is adjusted. We identify a neighborhood that completely surrounds the collapsing feature and assign a new material to the elements in this neighborhood. We refer to this as *recoloring*. Ideally, the encroaching material that is accumulating most rapidly is chosen to be the new material, but in this first version of our algorithm, the new material is chosen randomly from a list of viable adjacent materials. Soon after the material reassignment, the curvature driven interface motion will effectively straighten the interfaces. Figure 4 is a schematic of three types of topological change. The first frame in each sequence shows the event as it is detected by GRAIN3D; the dotted line demarcates the neighborhood to receive a new material assignment. The second frame shows the mesh just after the material reassignment, and the third frame shows the mesh after the interfaces have straightened. A further topological change, self-pinchoff, where a grain intersects itself and splits into two pieces, is yet to be incorporated in GRAIN3D.

7. Comparison against known solutions. To gauge the accuracy of the GWMFE method, we set up two types of test cases with analytically known collapse rates: spherical collapse and columnar grains.

7.1. Sphericallly symmetric collapse. The collapse of a spherical grain is easily solvable analytically and provides our first test of the accuracy of GWMFE. Assuming that a sphere is collapsing with normal velocity equal to the curvature (i.e., the *sum* of principle curvatures), we have

$$\frac{dr}{dt} = K = -\frac{2}{r}.$$

So

$$\begin{aligned}\frac{dA}{dt} &= \frac{d(4\pi r^2)}{dt} \\ &= 8\pi r \frac{dr}{dt} \\ &= -16\pi.\end{aligned}$$

The rate of change of surface areas is thus constant and is thus a convenient quantity to use for comparison with a numerical GWMFE solution. In Figure 5 we show the initial grid for a 42-node polyhedral representation of the sphere and then at $t = 0.04$. In Figure 6, we do the same for a 162-node sphere. The highest resolution case of a 642-node sphere is not shown. For each case, the triangles on the surface of the sphere are visible. Edges from tetrahedra conformally attached to the outside of the sphere are visible as well. Of course, tetrahedra within the sphere are not visible. Each surface point on the sphere is placed at exactly radius 0.5. We ran the code with $\mu = 1$, $\eta = 10^{-3}$, $h_{\max} = 0.3$, $\epsilon_1 = 5\eta^2$, $\epsilon_2 = 10^{-4}\eta^2\mu$, $\epsilon_3 = \eta\mu/h_{\max}$, $\epsilon_4 = 5\eta^2/h_{\max}$, and $\epsilon_5 = 10^{-4}\eta^2\mu$. In Figures 7–9, we display results for these cases

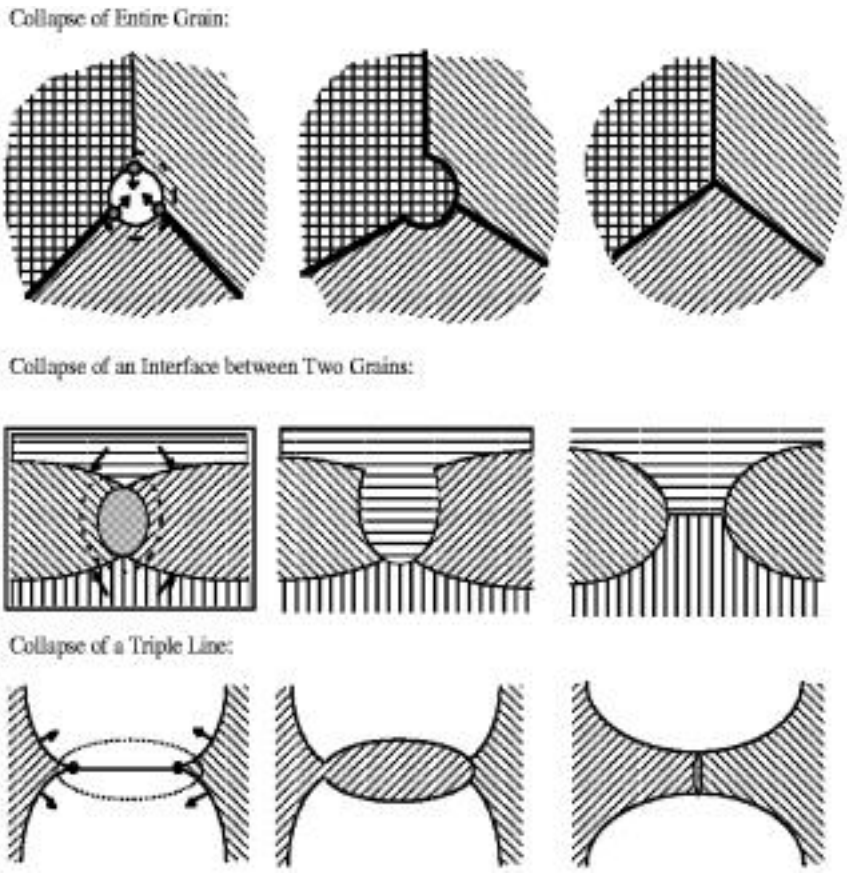


FIG. 4. *Topological collapse detection and resolution by GRAIN3D.*

showing computed graphs of $\frac{dA}{dt}$ vs. t and A vs. t (each dot signifies a time step) and compare them against the analytical results. Note that we apparently converge to the correct $-\frac{dA}{dt}$ as the number of nodes is increased. Also note that at $t = 0$, each numerical solution starts with a surface area less than the exact area π due to the flatness of the triangular facets. As can be seen, this initial surface area deficit is essentially maintained unchanged as the sphere collapses. Hence, errors introduced by the numerical solution seem to be less than errors introduced by simply discretizing the initial condition!

Of course, near the terminal time $t = \frac{\pi}{16\pi} = 0.0625$, we have that the numerical solution for these three cases deviates from the analytic solution. Near the terminal time, the tetrahedra inside the sphere have been collapsed to the point that the shortest altitudes approach η and the tetrahedral quality force (17)–(18) takes over and stops the collapse. If this were a grain growth simulation run, the topology change software would have recolored the collapsing tetrahedra to the color of the surrounding medium close to the terminal time. However, by turning off the recoloring algorithm, we are able to see how the tetrahedral quality force eventually will dominate if no corrective topological change is taken.

To get a better idea of the accuracy of the *pure* GWMFE method without regularization, we reran these three cases with $\mu = 1$, $\eta = 10^{-3}$, $h_{\max} = 0.3$, $\epsilon_1 = .001\eta^2$,

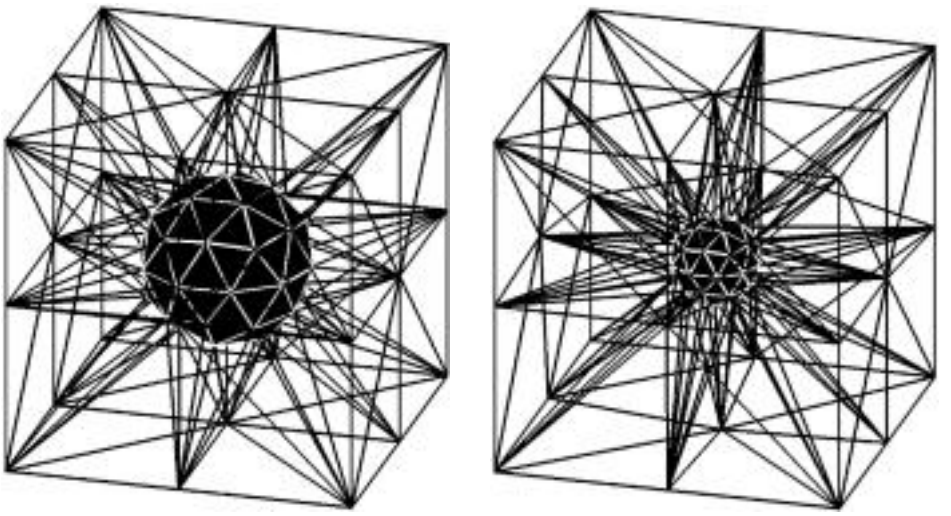


FIG. 5. *Initial* 162*-node sphere (with attached tetrahedra) and at* $t = 0.04$.

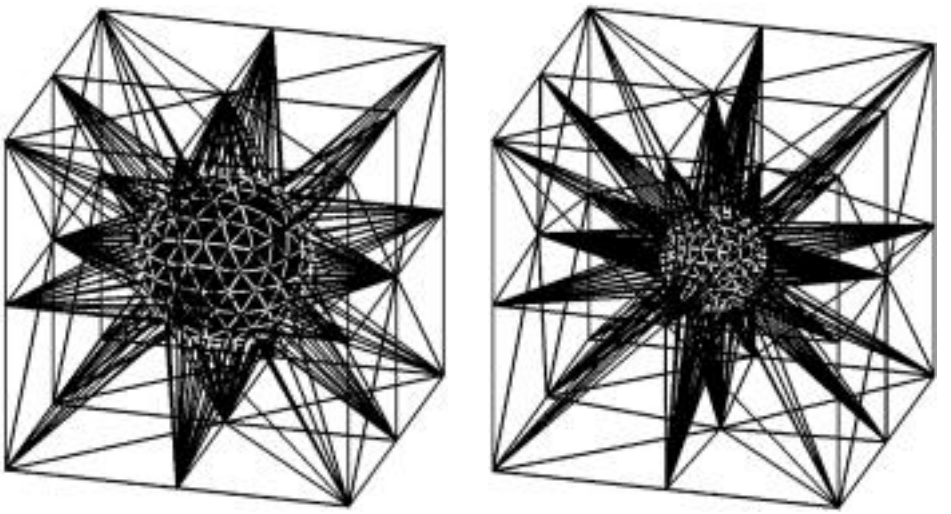


FIG. 6. *Initial* 162*-node sphere (with attached tetrahedra) and at* $t = 0.04$.

$\epsilon_2 = 0$, $\epsilon_3 = 0$, $\epsilon_4 = .001\eta^2/h_{\max}$, and $\epsilon_5 = 0$, which represent zero right-hand side regularization forces and left-hand side internodal viscosities far smaller than recommended. We plotted $-\frac{dA}{dt}$ for these three cases against the theoretical collapse rate in Figure 10. These simulations abruptly end when the terminal time is reached in the absence of counterbalancing tetrahedral quality forces. It is apparent from this figure that the error in $-\frac{dA}{dt}$ is constant over each run, and that the error is cut by roughly a factor of 4 when one increases the number of nodes from 42 (Case 1) to 162 (Case 2), and then the error drops by a factor of four when the number of nodes is increased again from 162 to 642 (Case 3). Since Case 2 represents a refinement which reduces by a factor of 2 the spacing h between nodes on the initial polyhedron (as compared to the 42-noded polyhedron of Case 1), and since Case 3 represents a further refinement

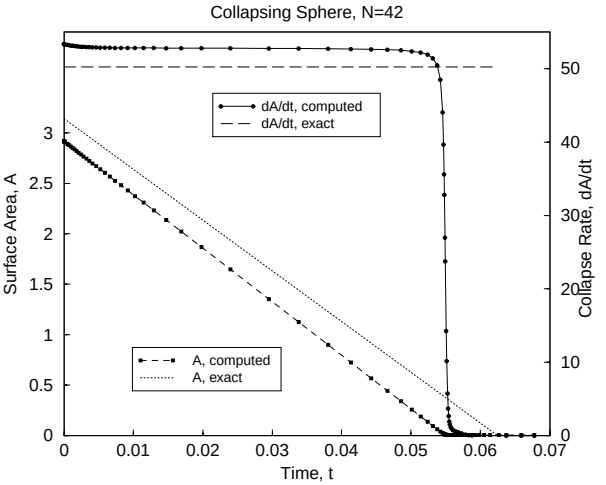


FIG. 7. *Results for* 42*-node sphere run.*

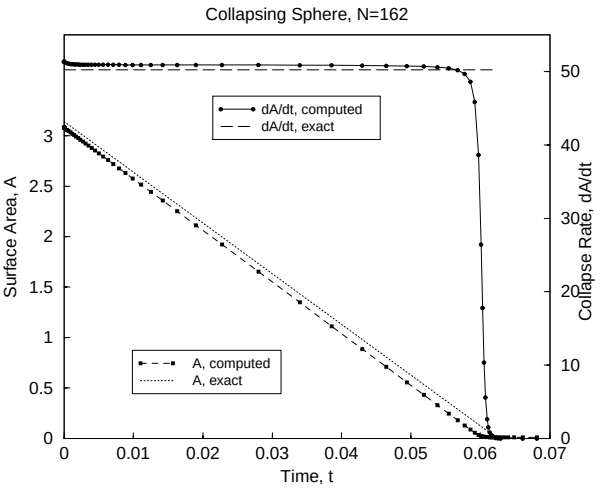


FIG. 8. *Results for* 162*-node sphere run.*

and halving of h, it is tempting to say the error in $-\frac{dA}{dt}$ as computed by the pure GWMFE method is $\mathcal{O}(h^2)$. However, as $t \to 0.0625$, we have that $h \to 0$, but the error remains constant. Instead, consider the Gauss map $\hat{\mathbf{n}} : M \mapsto S^2$ which maps each point $\mathbf{x}$ on a surface M to the unit outward normal $\hat{\mathbf{n}}(\mathbf{x})$ of the surface at $\mathbf{x}$. We define $\Delta\theta$ to be the distance between mapped grid points on the unit sphere. Then at all times for these three numerical test runs, the error in $-\frac{dA}{dt}$ using GWMFE appears to be $\mathcal{O}\big((\Delta\theta)^2\big)$. Thus, it would appear that high accuracy of the GWMFE solution depends on good *angular* resolution of the discretized surface.

The grid maintenance operations in section 5 (which are used in more complex microstructures such as that in section 8) strive to maintain a maximum edge length $h_{\max}$ rather than a coarsest angular resolution $\Delta\theta$. Given the statements of the last paragraph, this implies that small grains in the computations involving grid maintenance operations will have a relatively coarser angular discretization and will suffer

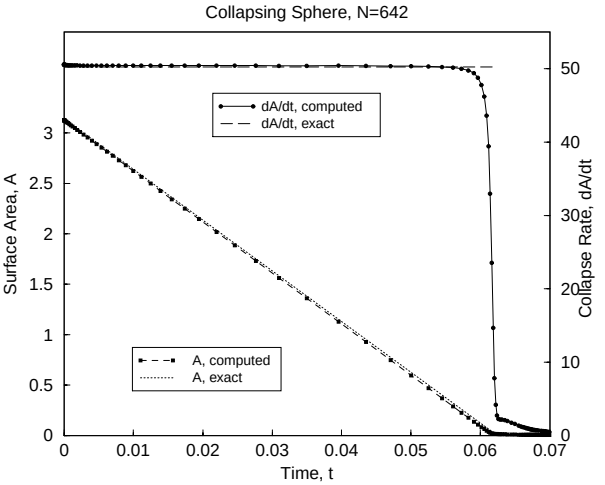


FIG. 9. *Results for* 642*-node sphere run.*

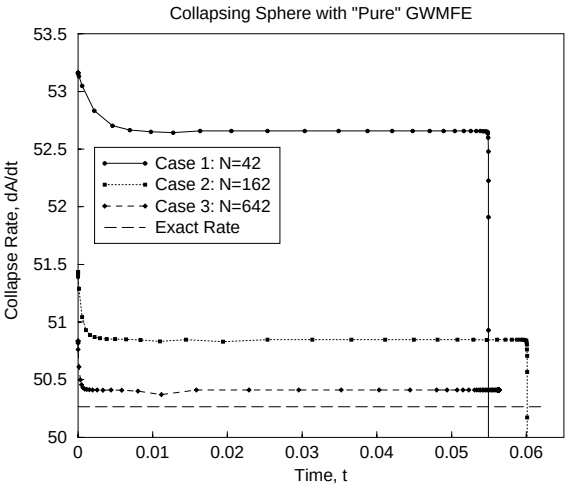


FIG. 10. *Results for nonregularized GWMFE.*

larger computational errors than larger grains. This is somewhat unavoidable, however, since in practical computations grains must be coarsened just prior to collapse.

7.2. Columnar microstructures. Our second set of test cases involve columnar microstructures which are arbitrary 2-D collections of grains that are extended into 3-D to be right cylinders. These microstructures do not possess curvature in 3-D, and thus are amenable to a 2-D analysis.

In 2-D, microstructures moving under curvature driven motion also obey Von Neumann's law [17], which states that for an interior grain G surrounded by n neighbors $G_1, G_2, \ldots, G_n$, the rate of area growth of G is

$$\frac{dA}{dt} = \mu \frac{\pi}{3}(n-6). \tag{21}$$

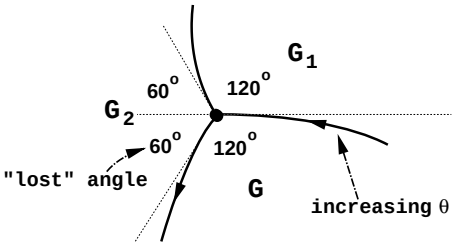


FIG. 11. *Loss of angle at triple point.*

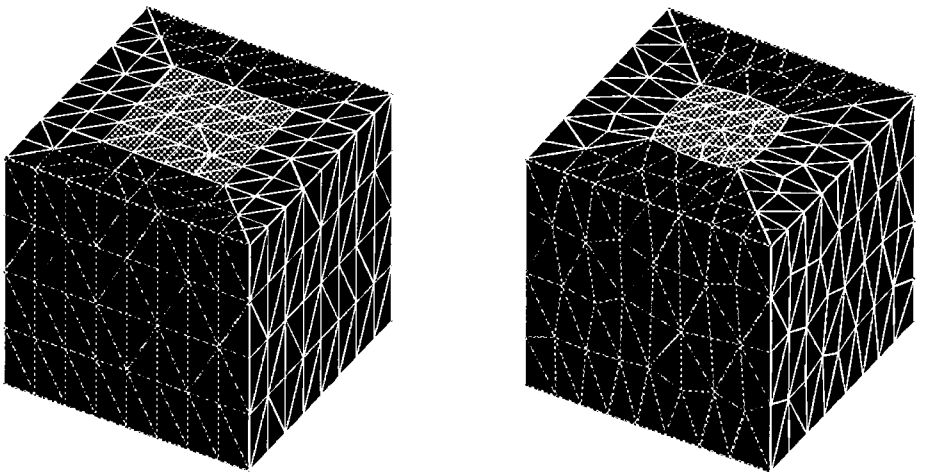


FIG. 12. *Initial* 16*-node columnar microstructure and at* $t = 0.2$. *The term* 16*-node refers to the number of nodes bounding top surface of internal grain.*

This is easily derived as follows. We have that

$$\frac{dA}{dt} = -\int_{\partial G} v_n \, ds,$$

where v_n is normal inward velocity, and the integral is taken over the arclength of the boundary of G. Since $v_n = \mu K$,

$$\begin{aligned}\frac{dA}{dt} &= -\int_{\partial G} \mu K \, ds \\ &= -\mu \int_{\partial G} \frac{d\theta}{ds} \, ds \\ &= -\mu \int_{\partial G} d\theta.\end{aligned}$$

$\int_{\partial G} d\theta$ is not 2π if $n > 0$, since at each triple point junction, there is a 60° loss of angle due to the 120°–120°–120° equilibrium angle condition (Figure 11) that exists at triple points resulting from the necessity of force balance of surface tension at the triple point. The interior grain G has n such triple points. We thus obtain (21).

In Figure 12 we show the initial grid for a five grain columnar microstructure with the axis running top to bottom. A central square-columnar grain G (light in

color) is surrounded by four trapezoidal-columnar grains G_1, G_2, G_3, G_4 (darker in color). The area A of the top surface of the central grain is $1 \times 1 = 1$, while the total top area of all five grains put together is $2 \times 2 = 4$. GRAIN3D was run on this initial microstructure using the same parameter values as in the regularized run of section 7.1. Nodes on the exterior planes were allowed to slide on the planes, nodes on exterior edges were allowed to slide on the edges, and the eight corner nodes were fixed. For this run, each of the four interfaces between the four surrounding grains (i.e., $G_1 \cap G_2$, $G_2 \cap G_3$, $G_3 \cap G_4$, and $G_4 \cap G_1$) was associated with and pinned to a vertical exterior edge.

Figure 12 also shows the evolved microstructure, as computed by GRAIN3D, at $t = 0.2$. The grains have remained columnar and the top surface of the central grain has shrunk. The four triple points on the top surface have adopted the correct 120°–120°–120° angle at this time. In Figure 13, we see computed and exact curves for $-\frac{dA}{dt}$ vs. t and A vs. t, where we have used (21) with $n = 4$ to obtain the "exact" value $-\frac{dA}{dt} = \frac{2\pi}{3}$. (We have called this run N = 16, since A is bounded by 16 computational nodes.) For this run, the computed and exact A vs. t are virtually identical. For most of the run, the computed and exact values for $-\frac{dA}{dt}$ are virtually indistinguishable. At the beginning of the run there is a "blip," where $-\frac{dA}{dt}$ is significantly higher than the theoretical value of $\frac{2\pi}{3}$. We attribute this to the state of the four triple points in the initial data. These four triple points are 90°–135°–135° in the initial data, which is not the 120°–120°–120° force balance equilibrium that should exist. Thus the initial blip represents rapid relaxation to the force balance state; essentially the code has accepted "incorrect" initial data (with forces not in balance at the triple points) and relaxed it to an acceptable state on a rapid time scale. The final blip in $-\frac{dA}{dt}$ occurs near the terminal time $t = \frac{3}{2\pi}$ when the central grain has been reduced to an extremely thin column, by which time the recoloring algorithm would have changed the topology, had it been activated. This blip is not surprising since the slender grain carries with it virtually no energy (due to its vanishingly small surface area), and therefore errors present (i.e., due to the nonphysical regularization forces) may well be magnified at this advanced stage.

In Figure 14, we see $-\frac{dA}{dt}$ vs. t and A vs. t for an $N = 32$ geometry (i.e., at twice the grid density of the one pictured in Figure 12). The initial anomalously high $-\frac{dA}{dt}$ blip is shorter lived, and the final blip at the terminal time has disappeared.

A study of "pure" GWMFE error (with regularization turned off) is impossible for this problem. It was possible in the highly symmetrical sphere collapse problem of section 7.1, but in this problem, which exhibits less symmetry and which has changing triple point angles, the regularization forces play a crucial role in preserving the positivity of all triangle areas and tetrahedral volumes throughout the computation.

8. Numerical example with 3-D microstructure. In this section, we show how GRAIN3D evolves a truly 3-D microstructure to steady-state, successfully maintaining mesh quality and performing necessary topological changes during the course of the calculation. In Figure 15(a), we show an initial five grain microstructure provided by the LaGriT tetrahedral mesh generator. We evolve the mesh with GRAIN3D, using the same regularization parameters as in the previous regularization runs. The nodes are allowed to slide on the external surfaces, as in section 7.2. Every $\Delta t_{\text{massage}} = 0.04$ time units, the mesh topology is massaged: edges that are too long are bisected, unneeded nodes are eliminated, and connections are swapped if necessary. Whenever we detect that a topological component is about to collapse (such as a grain, a grain-grain interface, or a triple line) in the next $\Delta t_{\text{collapse}} = 0.002$ time

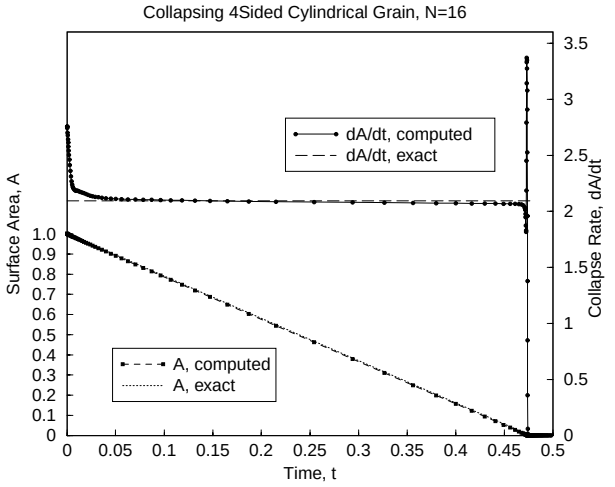


FIG. 13. *Results for* 16*-node run.*

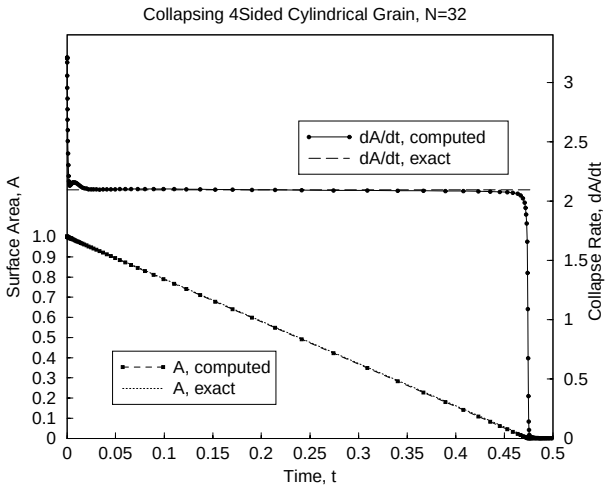
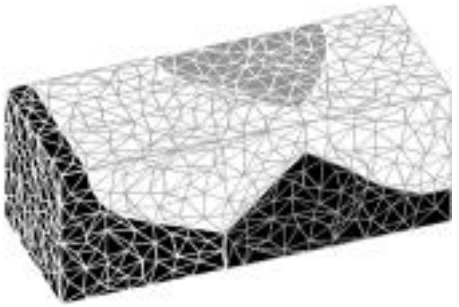


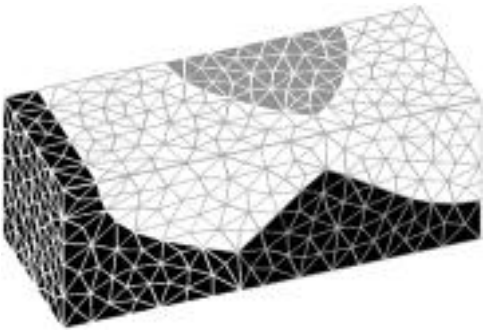
FIG. 14. *Results for* 32*-node run.*

units, we call the recoloring algorithm to effectively perform the necessary topological change.

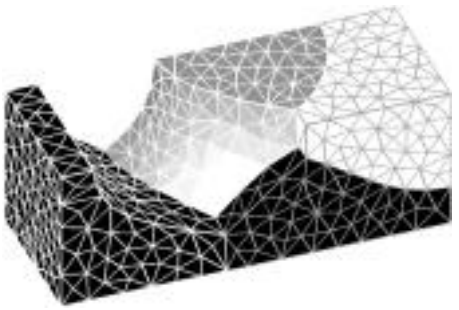
The massage algorithm is called immediately after $t = 0.00$, and the regularization forces (especially perimeter dependent surface energy (12)) fatten the elements, so by $t = 0.02$ (Figure 15(b)), the mesh is much improved. In 15(c), we have made one grain graphically transparent to reveal triples lines and quadruple points. At $t = 0.20$ (15(d)), the invisible grain has shrunk. At $t = 0.30$ (15(e)), the invisible grain has detached from the rear wall. In 15(f), the invisible grain is near collapse. By 15(g), the grain has been eliminated by the recoloring algorithm. This figure also shows a small rear grain about to collapse. In 15(i) this grain has collapsed and we are down to three grains. In fact, the microstructure is now columnar (the axis runs front to back) and would be perfectly well modeled by a 2-D front tracking code, such as that described in [5]. At $t = 1.50$ (15(j)), the light colored grain has disappeared, leaving



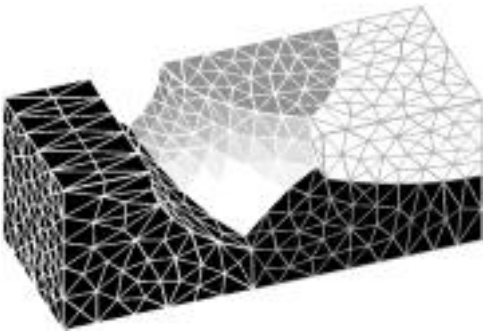
(a) $t = 0.00$. Initial grid is from LaGriT.



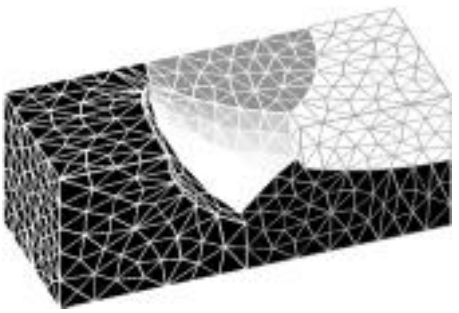
(b) $t = 0.02$. Grid improved by regularization and massage.



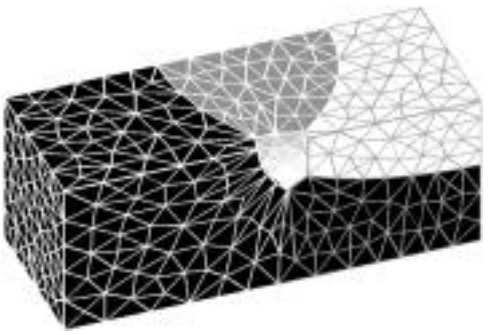
(c) $t = 0.02$. Grid with grain graphically removed for illustration.



(d) $t = 0.02$. Invisible grain has shrunk.



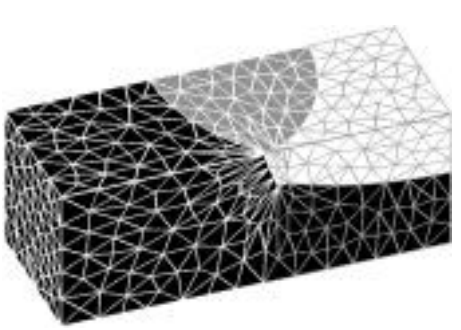
(e) $t = 0.30$. Invisible grain has detached from rear wall.



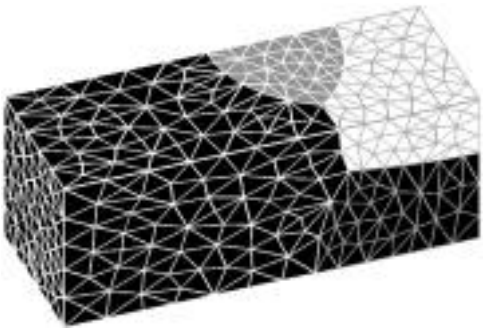
(f) $t = 0.45$. Invisible grain about to collapse.

FIG. 15 (a)–(f). *Grain evolution time sequence computed by GRAIN3D.*

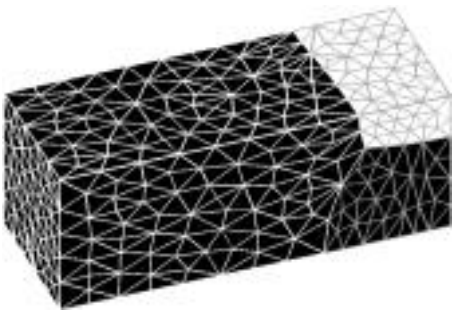
a collapsing circular-columnar grain and a larger complementary grain. At $t = 1.75$ (15(k)), the circular-columnar grain has collapsed, leaving the entire computational domain covered by a single grain. Evidence of the grain collapse is visible in that the grid is somewhat finer in the bottom right-hand corner. Finally, at $t = 1.80$ (15(l)), massage and regularization have coarsened the grid in the vicinity of the last collapsed grain. Of course, the difference between the grids at $t = 1.75$ and $t = 1.80$ is



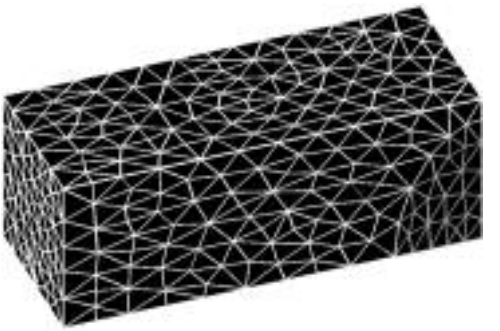
(g) $t = 0.47$. Invisible grain has collapsed.



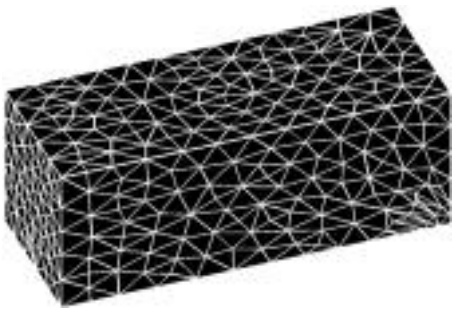
(h) $t = 0.80$. Surface triple points draw apart.



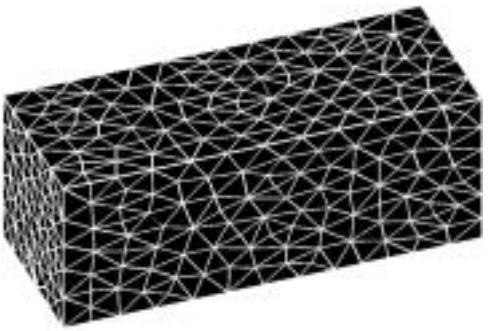
(i) $t = 1.10$. A second grain has collapsed. The microstructure is now 2-D.



(j) $t = 1.50$. A third grain has collapsed.



(k) $t = 1.75$. Fourth grain has collapsed. One grain left.



(l) $t = 1.80$. Grid improved by regularization and massage.

FIG. 15 (g)–(l). *Grain evolution time sequence computed by GRAIN3D.*

purely nonphysical—the grain is the same, but the grid is different. The run is truly completed at $t = 1.75$ when the microstructure evolution has ceased, but it isn't until $t = 1.80$ that the grid has settled down.

Acknowledgments. The author would like to acknowledge the invaluable support of the rest of the Los Alamos T-1 Grain Growth team: Denise George helped with enhancements to LaGriT, Tinka Gammel generated the initial microstructure used in section 8, Neil Carlson (now with Motorola, Inc.) provided assistance with his

GWMFE2DS code, and Galen Straub provided necessary motivation and resources. I also thank the referees for their detailed comments and corrections.

REFERENCES

[1] M.J. Baines, *Moving Finite Elements,* Oxford University Press, London, 1994.
[2] K. Brakke, *The surface evolver,* Experiment. Math., 1 (1992), pp. 141–165.
[3] N.N. Carlson and K. Miller, *Design and application of a gradient-weighted moving finite element code* I: *In one dimension,* SIAM J. Sci. Comput., 19 (1998), pp. 728–765.
[4] N.N. Carlson and K. Miller, *Design and application of a gradient-weighted moving finite element code* II: *In two dimensions,* SIAM J. Sci. Comput., 19 (1998), pp. 766–798.
[5] H.J. Frost, C.V. Thompson, C.L. Howe, and J. Whang, *A two-dimensional computer simulation of capillarity-driven grain growth: Preliminary results,* Scripta Met., 22 (1988), pp. 65–70.
[6] K. Fuchizaki and K. Kawasaki, 3-*dimensional computer modeling of grain-growth: A vertex model approach,* Materials Science Forum, 204 (1996), pp. 267–278.
[7] J.T. Gammel and A. Kuprat, *Modeling Metallic Microstructure: Incorporating Grain Boundary Orientation Dependence,* Tech. report LA-UR-98-1150, in the Special Features Supplement to the Theoretical Division Self-Assessment, Los Alamos National Laboratory, Los Alamos, NM, Spring 1998.
[8] D.C. George, *LaGriT User's Manual,* http://www.t12.lanl.gov/~lagrit.
[9] K. Kawasaki, T. Nagai, and K. Nakashima, *Vertex models for two-dimensional grain growth,* Phil. Mag. B, 60 (1989), p. 399.
[10] A. Kuprat, *Creation and Annihilation of Nodes for the Moving Finite Element Method,* Ph.D. thesis, University of California, Berkeley, May 1992.
[11] A. Liu and B. Joe, *Relationship between tetrahedron shape measures,* BIT, 34 (1994), pp. 268–287.
[12] P. Marcellini and K. Miller, *Regularization for prescribed mean curvature and for motion by mean curvature,* in Curvature Flows and Related Topics, Proceedings of the Second International Conference at Levico, Italy, 1994, A. Damlamian, J. Spruck, and A. Visintin, eds., Gakkotosho, Tokyo, 1995, pp. 145–158.
[13] P. Marcellini and K. Miller, *Elliptic versus parabolic regularization for the equation of prescribed mean curvature,* J. Differential Equations, 137 (1997), pp. 1–53.
[14] K. Miller, *Moving finite elements.* II, SIAM J. Numer. Anal., 18 (1981), pp. 1033–1057.
[15] K. Miller, *A geometrical-mechanical interpretation of gradient-weighted moving finite elements,* SIAM J. Numer. Anal., 34 (1997), pp. 67–90.
[16] K. Miller and R.N. Miller, *Moving finite elements.* I, SIAM J. Numer. Anal., 18 (1981), pp. 1019–1032.
[17] W.W. Mullins, *Two-dimensional motion of idealized grain boundaries,* J. Appl. Phys., 27 (1956), pp. 900–904.
[18] S. Nazari, *Rotational Surfaces in Euclidean and Hyperbolic Spaces, Mean Curvature Motion, and the Moving Finite Element Method,* Ph.D. thesis, Department of Mathematics, University of California, Berkeley, 1993.
[19] D.A. Porter and K.E. Easterling, *Phase Transformations in Metals and Alloys,* 2nd ed., Chapman and Hall, London, 1992.
[20] M. Rivara, *New longest-edge algorithms for the refinement and/or improvement of unstructured triangulations,* Internat. J. Numer. Methods Engrg., 40 (1997), pp. 3313–3324.
[21] S.J. Ruuth, *A diffusion-generated approach to multiphase motion,* J. Comput. Phys., 145 (1998), pp. 166–192.
[22] Y. Saad and M.H. Schultz, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems,* SIAM J. Sci. Stat. Comput., 7 (1986), pp. 856–869.
[23] J.A. Sethian, *Level Set Methods,* Cambridge University Press, London, 1996.
[24] E.Z. Xaba, *Robust Iterative Solvers for Linear and Nonlinear Finite Element Equations,* Ph.D. thesis, Department of Mathematics, University of California, Berkeley, 1997.
[25] H.-K. Zhao, T. Chan, B. Merriman, and S. Osher, *A variational level set approach to multiphase motion,* J. Comput. Phys., 127 (1996), pp. 179–195.

SIAM J. SCI. COMPUT.
Vol. 22, No. 2, pp. 561–581
© 2000 Society for Industrial and Applied Mathematics

AN APPROXIMATION TO MISCIBLE FLUID FLOWS IN POROUS MEDIA WITH POINT SOURCES AND SINKS BY AN EULERIAN–LAGRANGIAN LOCALIZED ADJOINT METHOD AND MIXED FINITE ELEMENT METHODS*

HONG WANG†, DONG LIANG‡, RICHARD E. EWING§, STEPHEN L. LYONS¶, AND GUAN QIN¶

Abstract. We develop an Eulerian–Lagrangian localized adjoint method (ELLAM)-mixed finite element method (MFEM) solution technique for accurate numerical simulation of coupled systems of partial differential equations (PDEs), which describe complex fluid flow processes in porous media. An ELLAM, which was shown previously to outperform many widely used methods in the context of linear convection-diffusion PDEs, is presented to solve the transport equation for concentration. Since accurate fluid velocities are crucial in numerical simulations, an MFEM is used to solve the pressure equation for the pressure and Darcy velocity. This minimizes the numerical difficulties occurring in standard methods for approximating velocities caused by differentiation of the pressure and then multiplication by rough coefficients.

The ELLAM-MFEM solution technique significantly reduces temporal errors, symmetrizes the governing transport equation, eliminates nonphysical oscillation and/or excessive numerical dispersion in many simulators, conserves mass, and treats boundary conditions accurately. Numerical experiments show that the ELLAM-MFEM solution technique simulates miscible displacements of incompressible fluid flows in porous media accurately with fairly coarse spatial grids and very large time steps, which are one or two orders of magnitude larger than the time steps used in many methods. Moreover, the ELLAM-MFEM solution technique can treat large mobility ratios, discontinuous permeabilities and porosities, anisotropic dispersion in tensor form, and point sources and sinks.

Key words. characteristic methods, Eulerian–Lagrangian methods, miscible fluid flows in porous media with wells, numerical simulation of convection-diffusion equations, reservoir simulation, subsurface contaminant transport

AMS subject classifications. 65M25, 65M60, 76M10, 76S05

PII. S1064827598349215

1. Introduction.

Many difficult problems arise in the numerical simulation of complex fluid flow processes in reservoir simulation, subsurface contaminant transport and remediation, and other applications. The mathematical models used to describe these fluid flow processes are coupled systems of nonlinear partial differential equations (PDEs), which are basically convection/diffusion types with convection being the dominant process. Due to the nonlinearity and couplings of these PDEs, the moving steep fronts present in the solutions of these PDEs, the singularities of the solutions at the point sources and sinks (e.g., injection and production wells), and the

*Received by the editors December 14, 1998; accepted for publication (in revised form) March 13, 2000; published electronically August 3, 2000. This research was supported in part by DOE grant DE-FG05-95ER25266, by ONR DEP-EPA-SCoR grant DAAG55-98-1-0002, and by a gift fund from the Upstream Strategic Research Center, Mobil Technology Company, Dallas, TX.

http://www.siam.org/journals/sisc/22-2/34921.html

†Department of Mathematics, University of South Carolina, Columbia, SC 29208 (hwang@math.sc.edu).

‡School of Mathematics and System Sciences, Shandong University, Jinan, Shandong, 250100, China (dliang@math.sdu.edu.cn).

§Institute for Scientific Computation, Texas A&M University, College Station, TX 77843-3404 (ewing@isc.tamu.edu).

¶Upstream Strategic Research Center, Mobil Technology Company, 13777 Midway Road, Dallas, TX 75244-4390 (steve_l_lyons@email.mobil.com, guan_gin@email.mobil.com).

enormous size of field-scale applications, the numerical treatment of these systems often encounters severe difficulties.

Let $c(\mathbf{x},t)$ be the concentration of an invading fluid, and let $p(\mathbf{x},t)$ and $\mathbf{u}(\mathbf{x},t)$ be the pressure and Darcy velocity of the total fluid mixture. The mass conservation equation for the fluid mixture incorporated with the incompressibility condition, Darcy's law, and the mass conservation equation for the invading fluid lead to the following coupled system of PDEs [3, 15], which models the miscible displacement of one incompressible fluid by another in a porous medium reservoir Ω over a time period of $[0,T]$:

$$\begin{aligned} \nabla\cdot\mathbf{u} &= q, && \mathbf{x}\in\Omega, \quad t\in[0,T], \\ \mathbf{u} &= -\frac{\mathbf{K}}{\mu(c)}(\nabla p-\rho g\nabla d), && \mathbf{x}\in\Omega, \quad t\in[0,T], \end{aligned} \tag{1.1}$$

$$\phi\frac{\partial c}{\partial t}+\nabla\cdot(\mathbf{u}c-\mathbf{D}(\mathbf{x},\mathbf{u})\nabla c)=\bar{c}q, \quad \mathbf{x}\in\Omega, \quad t\in[0,T]. \tag{1.2}$$

In many cases, the thickness of the medium is significantly smaller than its length and width. Hence, it is reasonable to average the medium properties vertically and to assume $\Omega\subset\mathbb{R}^2$ with a nonuniform local elevation. In (1.1)–(1.2), $\mathbf{x}:=(x,y)$. The dependent variables are the pressure $p(\mathbf{x},t)$ and the Darcy velocity $\mathbf{u}(\mathbf{x},t):=(u_x(\mathbf{x},t),u_y(\mathbf{x},t))$ of the fluid mixture, where $u_x(\mathbf{x},t)$ and $u_y(\mathbf{x},t)$ are the x and y components of $\mathbf{u}$, respectively, and the volumetric concentration $c(\mathbf{x},t)$ of the invading fluid. $\mathbf{K}(\mathbf{x})$ is the 2×2 permeability tensor of the medium, $\mu(c)$ is the concentration-dependent viscosity of the fluid mixture, which is determined by some mixing rule

$$\mu(c)=\mu(0)[(1-c)+M^{\frac{1}{4}}c]^{-4}, \tag{1.3}$$

where M is the mobility ratio between the resident and injected fluids, and $\mu(0)$ is the viscosity of resident fluid (oil). ρ is the density of the fluid mixture, g is the magnitude of gravitational acceleration, $d(\mathbf{x})$ is the reservoir depth, $q(\mathbf{x},t)$ is the external source/sink term that accounts for the effect of injection and production wells, $\phi(\mathbf{x})$ is the porosity of the medium, $\mathbf{D}$ is the diffusion-dispersion tensor

$$\mathbf{D}(\mathbf{x},\mathbf{u}):=\phi(\mathbf{x})d_m\,\mathbf{I}+\frac{d_l}{|\mathbf{u}|}\begin{pmatrix} u_x^2 & u_xu_y \\ u_xu_y & u_y^2 \end{pmatrix}+\frac{d_t}{|\mathbf{u}|}\begin{pmatrix} u_y^2 & -u_xu_y \\ -u_xu_y & u_x^2 \end{pmatrix}, \tag{1.4}$$

where d_m is the molecular diffusion coefficient, $\mathbf{I}$ is the 2×2 identity tensor, and d_l and d_t are the longitudinal and transverse dispersivities, respectively. $\bar{c}(\mathbf{x},t)$ is either the specified concentration of the injected fluid at injection wells or $\bar{c}(\mathbf{x},t)=c(\mathbf{x},t)$ is the resident concentration at production wells.

In reservoir simulation, the boundary $\Gamma:=\partial\Omega$ is typically impermeable. Consequently, the associated boundary conditions are given by

$$\begin{aligned} \mathbf{u}\cdot\mathbf{n} &= 0, && (\mathbf{x},t)\in\Gamma\times[0,T], \\ (\mathbf{D}\nabla c)\cdot\mathbf{n} &= 0, && (\mathbf{x},t)\in\Gamma\times[0,T]. \end{aligned} \tag{1.5}$$

In addition, the systems (1.1)–(1.2) also need an initial condition for the concentration

$$c(\mathbf{x},0)=c_0(\mathbf{x}), \quad \mathbf{x}\in\Omega. \tag{1.6}$$

The combination of the first equation in (1.1) and that in (1.5) leads to the following compatibility condition that must be imposed on the data:

$$\int_\Omega q(\mathbf{x},t)d\mathbf{x}=0, \quad t\in[0,T]. \tag{1.7}$$

Equation (1.7) states that for an incompressible flow with an impermeable boundary, the amount of injected fluid should be equal to the amount that is produced. In addition, (1.1) with the no-flow boundary condition (1.5) can only determine the pressure $p(\mathbf{x}, t)$ up to an additive constant for all the time $t \in [0, T]$. However, this indeterminacy is of no consequence since $\mathbf{u}$ is uniquely determined by Darcy's law, and only $\mathbf{u}$ (not p) is needed in (1.2).

2. State of the art in numerical approximations. The principal variable of physical interest in (1.1)–(1.2) is the concentration $c(\mathbf{x}, t)$. In reservoir simulation, it indicates how much of the reservoir is swept by solvent, or equivalently, how much oil is recovered. In subsurface contaminant transport, it shows the movement of the concerned solute in groundwater porous medium flows, which one wants to determine. Because the magnitude of the diffusion-dispersion tensor $\mathbf{D}$ is often much smaller than that of the Darcy velocity $\mathbf{u}$, (1.2) for c is a strongly convection-dominated PDE with small diffusion and dispersion terms indicated by the size of the coefficients d_m, d_l, and d_t in (1.4). Moreover, (1.1)–(1.2) are a coupled system of PDEs which is typically defined on a very large physical domain.

2.1. Numerical methods for elliptic pressure PDEs. One important issue in the simulation of porous medium flows is the manner in which the Darcy velocity $\mathbf{u}$ is calculated. Since the convection and diffusion-dispersion terms in (1.2) are governed by the Darcy velocity, accurate approximation to the concentration c requires an accurate approximation to the Darcy velocity $\mathbf{u}$. However, the properties of the porous medium (e.g., $\mathbf{K}$) often change abruptly with sharp changes in lithology. The viscosity $\mu(c)$ also changes rapidly across fluid interfaces. These sharp changes are accompanied by large changes in the pressure gradient which, in a compensatory fashion, yield a fairly smooth Darcy velocity $\mathbf{u}$. Standard finite difference or finite element methods (FDMs, FEMs, respectively) solve (1.1) for the pressure p, which is not necessarily smooth due to the impact of the rough coefficients. The resulting p is then differentiated numerically and then multiplied by a possibly rough coefficient to determine the Darcy velocity $\mathbf{u}$. Therefore, these methods generate a rough and often inaccurate velocity $\mathbf{u}$, which then reduces the accuracy of the approximation to (1.2). Mixed finite element methods (MFEMs) [6, 24] approximate both p and $\mathbf{u}$ from the system (1.1) simultaneously, yield an accurate velocity field $\mathbf{u}$, and conserve mass. MFEMs have been successfully applied in reservoir simulation [15, 27].

2.2. Numerical methods for convection-diffusion PDEs. Standard FDMs or FEMs tend to generate solutions with severe nonphysical oscillations. In industrial applications, upstream weighting techniques are commonly used to stabilize the numerical approximations in large-scale simulators. However, these methods produce excessive numerical dispersion and spurious effects related to the grid orientation [15]. Two general classes of improved methods can be identified from the literature: the Eulerian methods that use the standard temporal discretization and the characteristic methods that carry out the temporal discretization by a characteristic tracking.

Most Eulerian methods are based on upstream weighting techniques. The optimal test function methods [2, 10] attempt to minimize the spatial error and yield an upstream bias in the resulting schemes. Some Eulerian methods [5] try to reduce the overall truncation error by using a nonzero spatial error to cancel temporal errors. The streamline diffusion FEMs [20, 21] add a numerical diffusion only in the direction of streamlines with no crosswind diffusion introduced. However, an a priori choice of the free parameter in the methods is not clear and is heavily problem de-

pendent. The total variation diminishing methods (TVD), essentially nonoscillatory (ENO) methods, and other high resolution methods [11, 14, 18, 29] are well suited for nonlinear hyperbolic conservation laws and resolve shock discontinuities in the solutions without excessive smearing or spurious oscillations. These methods were extended to solve convection-diffusion PDEs [9].

Because of the hyperbolic nature of convective transport, many characteristic methods have been developed for solving convection-diffusion PDEs [4, 13, 23, 30]. Traditional forward tracking or moving mesh methods advance the grids following the characteristics and greatly reduce temporal errors. But they often severely distort the evolving grids and greatly complicate the solution procedures. The modified method of characteristics (MMOC) [13] tracks the characteristics backward from a fixed grid at the current time step and, hence, avoids the grid distortion problems present in forward tracking methods. The MMOC symmetrizes and stabilizes the convection-diffusion PDEs, greatly reduces temporal errors and so allows for large time steps in a simulation without loss of accuracy, and eliminates the excessive numerical dispersion and grid orientation effects present in many Eulerian methods [15, 27]. However, MMOC and many other characteristic methods fail to conserve mass and have difficulties in treating general boundary conditions.

2.3. Numerical approximations to the systems (1.1)–(1.2). The combination of close couplings and nonlinearities of the governing PDEs, the singularities of the solutions due to the effect of injection and production wells, the moving steep fronts present in the solutions of the governing PDEs, the use of large grid-spacings, and the convection-diffusion feature of the governing PDEs often generate spurious numerical artifacts and inaccurate approximations to the systems. A blind linearization with little regard to the properties of the governing PDEs or the solutions can result in extremely large, ill-conditioned, nonlinear discrete algebraic systems. These issues, if not treated carefully, may destroy the usefulness of the simulation.

Fully explicit methods are computationally local and are very efficient per time step, but they often require extremely fine spatial grids and time steps and result in enormous amounts of overall computations in numerical simulations [15, 31]. Fully coupled and fully implicit methods solve all of the coupled nonlinear PDEs simultaneously in an implicit fashion, and they are unconditionally stable. There has been development of fully implicit FDM or FEM simulators for three phase flows, for steam flooding, and for compositional models [15, 27]. However, these methods are computationally expensive per time step and introduce excessive numerical diffusion and reduced accuracy with large time steps [15, 31]. IMPES methods, which solve (1.1) *implicitly* for *pressure* and (1.2) *explicitly* for *saturation* (or concentration), provide variations to fully explicit methods to obtain better stability without increasing complexity too much. However, for difficult nonlinearities, IMPES methods are often forced to use extremely small time steps which lead to significantly compromised computational efficiency [15].

Douglas, Ewing, and Wheeler [12] presented and analyzed an efficient sequential linearization technique for the miscible displacement of one incompressible fluid by another in a porous medium. In the procedure, an MFEM was used to solve (1.1) for the pressure and Darcy velocity and a Galerkin FEM was used to solve (1.2) for the concentration. However, the use of a Galerkin FEM may generate numerical solutions with severe oscillation. Russell generalized the work in [13] and introduced the MMOC into the Society of Petroleum Engineers (SPE) literature [25] for solving (1.2) in miscible displacement of incompressible fluids in reservoir simulation, where

the pressure equation (1.1) was solved by a biquadratic FEM. Subsequently, Ewing, Russell, and Wheeler in [12] and Russell in [25] combined the ideas and proposed an improved MMOC-MFEM sequential solution technique in the numerical simulation of the miscible displacement of incompressible fluid flows in porous media [16], in which the MMOC was used to solve (1.2) and an MFEM was employed to solve (1.1). The use of an MFEM for the pressure equation yields accurate Darcy velocity fields that conserve mass, while the application of MMOC allows large time steps to be used in the solution of the transport equation without loss of accuracy and eliminates the numerical dispersion and grid orientation effects that are among the major difficulties presented in large-scale reservoir simulators [15, 27].

3. An ELLAM scheme for the transport equation (1.2). The ELLAM was introduced by Celia et al. [8] for solving (one-dimensional constant-coefficient) convection-diffusion PDEs. The ELLAM formalism provides a general characteristic solution procedure for convection-dominated PDEs and a consistent framework for conserving mass and treating general boundary conditions. The ELLAM techniques symmetrize these PDEs, generate accurate numerical solutions even if large time steps are used, and eliminate nonphysical oscillations, numerical dispersion, and grid orientation artifacts. Thus, the ELLAM framework overcomes the principal shortcomings of the previous characteristic methods while maintaining their numerical advantages. In this section, we present an ELLAM scheme for the transport equation (1.2) under the assumption that the Darcy velocity $\mathbf{u}$ is known. An ELLAM-MFEM sequential solution technique for the coupled systems (1.1)–(1.2) will be described in section 5.

3.1. A reference equation. We define a partition of the interval $[0, T]$ by

$$0 =: t_0^c < t_1^c < \cdots < t_n^c < \cdots < t_{N-1}^c < t_N^c := T \quad \text{and} \quad \Delta t_n^c := t_n^c - t_{n-1}^c. \tag{3.1}$$

In the ELLAM framework, we choose the test functions $z(\mathbf{x}, t)$ to be continuous and piecewise smooth on the space-time strip $\Omega \times (t_{n-1}^c, t_n^c]$. We allow them to be discontinuous in time at time t_{n-1}^c so that the ELLAM scheme can be decoupled in time. This in turn permits us to focus on the development of the scheme on the current time interval $[t_{n-1}^c, t_n^c]$ only and to define the test functions by constant extension along the characteristics as we should see below. We refer readers to [1, 8, 31] for details. Multiplying (1.2) by these test functions $z(\mathbf{x}, t)$, and integrating the resulting equation over the space-time strip $\Omega \times (t_{n-1}^c, t_n^c]$, we obtain the following space-time weak formulation for (1.2):

$$\begin{aligned} &\int_\Omega \phi(\mathbf{x}) c(\mathbf{x}, t_n^c) z(\mathbf{x}, t_n^c)\, d\mathbf{x} + \int_{t_{n-1}^c}^{t_n^c} \int_\Omega \nabla z(\mathbf{y}, \theta) \cdot \mathbf{D}(\mathbf{y}, \mathbf{u}(\mathbf{y}, \theta)) \nabla c(\mathbf{y}, \theta)\, d\mathbf{y} d\theta \\ &\quad - \int_{t_{n-1}^c}^{t_n^c} \int_\Omega c(\mathbf{y}, \theta) \left[\phi(\mathbf{y}) \frac{\partial z(\mathbf{y}, \theta)}{\partial \theta} + \mathbf{u}(\mathbf{y}, \theta) \cdot \nabla z(\mathbf{y}, \theta) \right] d\mathbf{y} d\theta \\ &\quad = \int_\Omega \phi(\mathbf{x}) c(\mathbf{x}, t_{n-1}^c) z(\mathbf{x}, t_{n-1}^{c,+})\, d\mathbf{x} + \int_{t_{n-1}^c}^{t_n^c} \int_\Omega \bar{c}(\mathbf{y}, \theta) q(\mathbf{y}, \theta) z(\mathbf{y}, \theta)\, d\mathbf{y} d\theta, \end{aligned} \tag{3.2}$$

where $z(\mathbf{x}, t_{n-1}^{c,+}) := \lim_{t \to t_{n-1}^c,\ t > t_{n-1}^c} z(\mathbf{x}, t)$ to take into account the fact that $z(\mathbf{x}, t)$ is discontinuous in time at time t_{n-1}^c. We replace the dummy variables $\mathbf{x}$ and t in the space-time integrals in (3.2) by $\mathbf{y}$ and θ, and we reserve $\mathbf{x}$ for the point in Ω at time t_n^c or t_{n-1}^c for later convenience.

Different splittings of the adjoint equation of (1.2) are studied in the ELLAM framework. It is shown [8] that the test functions should be chosen to satisfy the adjoint equation of (1.2) to reflect the hyperbolic nature of (1.2):

$$\phi(\mathbf{y})\ \frac{\partial z(\mathbf{y},\theta)}{\partial \theta} + \mathbf{u}(\mathbf{y},\theta)\cdot\nabla z(\mathbf{y},\theta) = 0, \quad \mathbf{y}\in\overline{\Omega}, \quad \theta\in[t_{n-1}^c, t_n^c]. \tag{3.3}$$

Thus, the test functions $z(\mathbf{y},\theta)$ should be constant along the characteristics $\mathbf{y} = \mathbf{r}(\theta;\mathbf{x},t_n^c)$, defined by the initial-value problem of the differential equation

$$\frac{d\mathbf{r}}{d\theta} = \frac{\mathbf{u}(\mathbf{r},\theta)}{\phi(\mathbf{r})} \quad \text{and} \quad \mathbf{r}(\theta;\bar{\mathbf{x}},\bar{t})\Big|_{\theta=\bar{t}} = \bar{\mathbf{x}}. \tag{3.4}$$

For any $(\mathbf{y},\theta)\in\Omega\times[t_{n-1}^c,t_n^c]$, there exists an $\mathbf{x}\in\Omega$ such that $\mathbf{y}=\mathbf{r}(\theta;\mathbf{x},t_n^c)$. We apply the Euler formula at time t_n^c to evaluate the source and sink term in (3.2) to obtain

$$\begin{aligned}
&\int_{t_{n-1}^c}^{t_n^c}\int_\Omega \bar{c}(\mathbf{y},\theta)q(\mathbf{y},\theta)z(\mathbf{y},\theta)\ d\mathbf{y}d\theta \\
&\quad= \int_\Omega\int_{t_{n-1}^c}^{t_n^c} \bar{c}(\mathbf{r}(\theta;\mathbf{x},t_n),\theta)q(\mathbf{r}(\theta;\mathbf{x},t_n),\theta)z(\mathbf{x},t_n)\left|\frac{\partial\mathbf{r}(\theta;\mathbf{x},t_n^c)}{\partial\mathbf{x}}\right| d\theta d\mathbf{x} \\
&\quad= \Delta t_n^c\int_\Omega \bar{c}(\mathbf{x},t_n^c)q(\mathbf{x},t_n^c)z(\mathbf{x},t_n^c)d\mathbf{x} + E_q(\bar{c},z),
\end{aligned} \tag{3.5}$$

where $|\frac{\partial\mathbf{r}(\theta;\mathbf{x},t_n^c)}{\partial\mathbf{x}}| = 1+\mathcal{O}(t_n^c-\theta)$ is the Jacobian of the transformation from $\mathbf{x}$ to $\mathbf{y}=\mathbf{r}(\theta;\mathbf{x},t_n^c)$, and $E_q(\bar{c},z)$ is the local truncation error term.

Likewise, we evaluate the diffusion-dispersion term in (1.2) to obtain

$$\begin{aligned}
&\int_{t_{n-1}^c}^{t_n^c}\int_\Omega \nabla z(\mathbf{y},\theta)\cdot\mathbf{D}(\mathbf{y},\mathbf{u}(\mathbf{y},\theta))\nabla c(\mathbf{y},\theta)\ d\mathbf{y}d\theta \\
&\qquad= \Delta t_n^c\int_\Omega \nabla z(\mathbf{x},t_n^c)\cdot\mathbf{D}(\mathbf{x},\mathbf{u}(\mathbf{x},t_n^c))\nabla c(\mathbf{x},t_n^c)\ d\mathbf{x} + E_{\mathbf{D}}(c,z),
\end{aligned} \tag{3.6}$$

where $E_{\mathbf{D}}(c,z)$ is the local truncation error term.

Substituting (3.5) and (3.6) into (3.2), we obtain a reference equation

$$\begin{aligned}
&\int_\Omega \phi(\mathbf{x})c(\mathbf{x},t_n^c)z(\mathbf{x},t_n^c)\ d\mathbf{x} + \Delta t_n^c\int_\Omega \nabla z(\mathbf{x},t_n^c)\cdot\mathbf{D}(\mathbf{x},\mathbf{u}(\mathbf{x},t_n^c))\nabla c(\mathbf{x},t_n^c)\ d\mathbf{x} \\
&= \int_\Omega \phi(\mathbf{x})c(\mathbf{x},t_{n-1}^c)z(\mathbf{x},t_{n-1}^{c,+})\ d\mathbf{x} + \Delta t_n^c\int_\Omega \bar{c}(\mathbf{x},t_n^c)\ q(\mathbf{x},t_n^c)z(\mathbf{x},t_n^c)\ d\mathbf{x} + E(c,z),
\end{aligned} \tag{3.7}$$

where

$$\begin{aligned}
E(c,z) := \ &\int_{t_{n-1}^c}^{t_n^c}\int_\Omega c(\mathbf{y},\theta)\ \left[\phi(\mathbf{y})\ \frac{\partial z(\mathbf{y},\theta)}{\partial\theta} + \mathbf{u}(\mathbf{y},\theta)\cdot\nabla z(\mathbf{y},\theta)\right]\ d\mathbf{y}d\theta \\
&-E_{\mathbf{D}}(c,z) + E_q(\bar{c},z).
\end{aligned} \tag{3.8}$$

3.2. An ELLAM scheme. We present an ELLAM scheme for (1.2). In reservoir simulations, the physical domain is typically a rectangular domain or a finite union of rectangular domains. For simplicity, we assume $\Omega := (a_x,b_x)\times(a_y,b_y)$ and define a tensor-product spatial partition

$$\begin{aligned}
a_x =: \ & x_0^c < x_1^c < \cdots < x_i^c < \cdots < x_{I-1}^c < x_I^c \ := b_x, \\
a_y =: \ & y_0^c < y_1^c < \cdots < y_j^c < \cdots < y_{J-1}^c < y_J^c \ := b_y.
\end{aligned} \tag{3.9}$$

We define the trial and test function spaces to be the space of continuous and piecewise bilinear polynomials on the partition (3.9). Namely,

$$S^c(\Omega) := M^c_{0,1}[a_x, b_x] \otimes M^c_{0,1}[a_y, b_y], \tag{3.10}$$

where

$$\begin{aligned} M^c_{\alpha,\beta}[a_x, b_x] &:= \Big\{ v \in C^\alpha[a_x, b_x] \;\Big|\; v|_{[x^c_{i-1}, x^c_i]} \in P_\beta[x^c_{i-1}, x^c_i],\ i = 1, \ldots, I \Big\}, \\ M^c_{\alpha,\beta}[a_y, b_y] &:= \Big\{ v \in C^\alpha[a_y, b_y] \;\Big|\; v|_{[y^c_{j-1}, y^c_j]} \in P_\beta[y^c_{j-1}, y^c_j],\ j = 1, \ldots, J \Big\}. \end{aligned} \tag{3.11}$$

Here, $C^0[a, b]$ denotes the space of continuous functions on $[a, b]$ and $P_\beta[a, b]$ is the space of univariate polynomials of degree less than or equal to β, restricted to the interval $[a, b]$.

If we assume that the Darcy velocity $\mathbf{u}(\mathbf{x}, t^c_n)$ in (1.2) is known, then the ELLAM scheme is defined as follows: Find $c(\mathbf{x}, t^c_n) \in S^c(\Omega)$ such that

$$\begin{aligned} &\int_\Omega \phi(\mathbf{x}) c(\mathbf{x}, t^c_n) z(\mathbf{x}, t^c_n)\, d\mathbf{x} + \Delta t^c_n \int_\Omega \nabla z(\mathbf{x}, t^c_n) \cdot \mathbf{D}(\mathbf{x}, \mathbf{u}(\mathbf{x}, t^c_n)) \nabla c(\mathbf{x}, t^c_n)\, d\mathbf{x} \\ &\quad = \int_\Omega \phi(\mathbf{x}) c(\mathbf{x}, t^c_{n-1}) z(\mathbf{x}, t^{c,+}_{n-1})\, d\mathbf{x} + \Delta t^c_n \int_\Omega \bar{c}(\mathbf{x}, t^c_n)\, q(\mathbf{x}, t^c_n) z(\mathbf{x}, t^c_n)\, d\mathbf{x} \\ &\qquad\qquad \forall z(\mathbf{x}, t^c_n) \in S^c(\Omega). \end{aligned} \tag{3.12}$$

Remark 3.1. By using a characteristic tracking, the ELLAM scheme (3.12) symmetrizes the transport PDE (1.2), significantly reduces temporal errors and so generates accurate solutions even if large time steps are used in numerical simulations, and yields a 9-banded, symmetric, and positive-definite coefficient matrix. Furthermore, the ELLAM scheme conserves mass [8, 26], i.e.,

$$\int_\Omega \phi(\mathbf{x}) c(\mathbf{x}, t^c_n) d\mathbf{x} = \int_\Omega \phi(\mathbf{x}) c(\mathbf{x}, t^c_{n-1}) d\mathbf{x} + \int_{t^c_{n-1}}^{t^c_n} \int_\Omega \bar{c}(\mathbf{x}, t) q(\mathbf{x}, t) d\mathbf{x} dt, \tag{3.13}$$

which is of essential importance in applications.

Remark 3.2. Because $c(\mathbf{x}, t^c_n), z(\mathbf{x}, t^c_n) \in S^c(\Omega)$ are standard piecewise bilinear functions at time t^c_n, in (3.12) all the terms except the first one on the right-hand side are standard integrals in FEMs and can be evaluated in a fairly standard way. In the first term on the right-hand side, the value of $c(\mathbf{x}, t^c_{n-1})$ is known from the solution at the previous time step t^c_{n-1}. However, the test functions $z(\mathbf{x}, t^{c,+}_{n-1}) := \lim_{t \to t^c_{n-1},\ t > t^c_{n-1}} z(\mathbf{x}, t) = z(\tilde{\mathbf{x}}, t^c_n)$, where $\tilde{\mathbf{x}} = \mathbf{r}(t^c_n; \mathbf{x}, t^c_{n-1})$ is the point at the head of the characteristic that corresponds to $\mathbf{x}$ at the foot. The evaluation of this term becomes much more challenging in multiple dimensions, due to the multidimensional deformation of each cell $[x^c_{i-1}, x^c_i] \times [y^c_{j-1}, y^c_j]$ on which the test functions are defined as the geometry is backtracked from time step t^c_n to time step t^c_{n-1}.

The most practical approach for evaluating this term is to use a forward tracking algorithm proposed by Russell and Trujillo [26]. In this algorithm, an integration quadrature is enforced at time step t^c_{n-1} with respect to the fixed spatial grids (3.9) on which $c(\mathbf{x}, t^c_{n-1})$ is defined, and the difficult evaluation is the test function $z(\mathbf{x}, t^{c,+}_{n-1})$. Rather than backtracking the geometry and estimating the test functions by mapping the deformed geometry onto the fixed grids (3.9), discrete quadrature points chosen on the fixed grids at time step t^c_{n-1} can be tracked forward to time step t^c_n, where evaluation of $z(\mathbf{x}, t^c_n)$ is straightforward. Because the forward tracking algorithm

is used only to evaluate the first term on the right-hand side of (3.12), it has no effect on the solution grids (3.9) or the data structure of the discrete system for (3.12). Therefore, the forward tracking algorithm used here does not suffer from the complication of distorted grids, which complicates many forward tracking algorithms.

Remark 3.3. For a general velocity field $\mathbf{u}(\mathbf{x}, t)$, one cannot solve the initial-value problem (3.4) analytically to track characteristics. Hence, numerical quadratures (e.g., Euler or Runge–Kutta methods) were previously used to track characteristics in the ELLAM schemes for linear transport PDEs where $\mathbf{u}(\mathbf{x}, t)$ is assumed to be a known smooth function [1, 17, 31]. However, in the current circumstances, the velocity field $\mathbf{u}(\mathbf{x}, t)$ is given as a Raviart–Thomas MFEM solution to (1.1) (refer to section 4), which is only piecewise (cell-by-cell) smooth. Therefore, numerical quadratures would generate an inaccurate characteristic tracking, which in turn affects the accuracy of the solutions of the ELLAM schemes. In applications, within each cell the porosity $\phi(\mathbf{x})$ is constant and $u_x(\mathbf{x}, t_n^c)$ (or $u_y(\mathbf{x}, t_n^c)$) is linear (or constant) in the x direction and constant (or linear) in the y direction. Therefore, we can solve the problem (3.4) analytically to track the characteristics on a cell-by-cell basis [17, 19, 28]. In this way, we also minimize the effect of the well singularities on the characteristic tracking.

Remark 3.4. In the MMOC, which is a typical representative of many previous characteristic methods, (1.2) is rewritten in a nonconservative form

$$\phi \frac{\partial c}{\partial t} + \mathbf{u} \cdot \nabla c - \nabla \cdot (\mathbf{D}(\mathbf{x}, \mathbf{u}) \nabla c) = (\bar{c} - c) q, \quad \mathbf{x} \in \Omega, \quad t \in [0, T], \tag{3.14}$$

where the first equation in (1.1) has been used to get $(\nabla \cdot \mathbf{u})\ c = c\ q$.

Then the first two terms on the left-hand side of (3.14) are combined to form one term through a characteristic tracking [13]

$$\begin{aligned} &\phi(\mathbf{x}) \frac{\partial c(\mathbf{x}, t_n^c)}{\partial t} + \mathbf{u}(\mathbf{x}, t_n^c) \cdot \nabla c(\mathbf{x}, t_n^c) \\ &\quad = \sqrt{\phi^2(\mathbf{x}) + |\mathbf{u}(\mathbf{x}, t_n^c)|^2}\ \frac{dc(\mathbf{r}(t; \mathbf{x}, t_n^c), t)}{dt}\bigg|_{t = t_n^c} \\ &\quad \approx \phi(\mathbf{x}) \frac{c(\mathbf{x}, t_n^c) - c(\mathbf{x}^*, t_{n-1}^c)}{\Delta t_n^c}, \end{aligned} \tag{3.15}$$

where

$$\mathbf{x}^* := \mathbf{x} - \frac{\mathbf{u}(\mathbf{x}, t_n^c)}{\phi(\mathbf{x})} \Delta t_n^c. \tag{3.16}$$

Substituting (3.15) for the first two terms on the left-hand side of (3.14) and integrating the resulting equation against any test functions $v(\mathbf{x}) \in S^c(\Omega)$, one obtains the following MMOC scheme [13, 16] for (3.14):

$$\begin{aligned} &\int_\Omega \phi(\mathbf{x})\ \frac{c(\mathbf{x}, t_n^c) - c(\mathbf{x}^*, t_{n-1}^c)}{\Delta t_n^c}\ v(\mathbf{x}) d\mathbf{x} \\ &\qquad + \int_\Omega \nabla v(\mathbf{x}) \cdot \mathbf{D}(\mathbf{x}, \mathbf{u}(\mathbf{x}, t_n^c)) \nabla c(\mathbf{x}, t_n^c)\ d\mathbf{x} \\ &\quad = \int_\Omega (\bar{c}(\mathbf{x}, t_n^c) - c(\mathbf{x}, t_n^c))\ q(\mathbf{x}, t_n^c) v(\mathbf{x})\ d\mathbf{x} \qquad \forall v(\mathbf{x}, t_n^c) \in S^c(\Omega). \end{aligned} \tag{3.17}$$

Although by (1.5) $\mathbf{u}(\mathbf{x}, t) \cdot \mathbf{n}(\mathbf{x})|_\Gamma = 0$, there might be some $\mathbf{x} \in \Omega$, which is close to Γ, such that the corresponding $\mathbf{x}^*$ determined by (3.16) runs out of the domain

Ω. This leads to implementational and analytical difficulties in the application of the MMOC. In contrast, in the ELLAM scheme (3.12) the tracking algorithm is carried out by solving the problem (3.4) analytically on a cell-by-cell basis. Since the numerical Darcy velocity $\mathbf{u}$ satisfies the no-flow boundary condition, the characteristic tracking never runs out of the domain Ω and so avoids this difficulty. This is an additional advantage of the ELLAM scheme.

4. An MFEM for the pressure equation (1.1). Because they yield an accurate velocity field $\mathbf{u}$ for porous medium flows and conserve mass, MFEMs have been widely applied in porous medium flow problems [27]. In this section, we briefly describe an MFEM for the pressure system (1.1).

4.1. Preliminary notations. Let $L^2(\Omega)$ be the standard function space of all the Lebesgue square integrable functions on Ω. Then we define the Sobolev spaces

$$
\begin{aligned}
H^1(\Omega) &:= \left\{ v(\mathbf{x}) \;\middle|\; \frac{\partial v(x,y)}{\partial x},\ \frac{\partial v(x,y)}{\partial y} \in L^2(\Omega) \right\}, \\
L_0^2(\Omega) &:= \left\{ v(\mathbf{x}) \in L^2(\Omega) \;\middle|\; \int_\Omega v(\mathbf{x}) d\mathbf{x} = 0 \right\}, \\
H(\mathrm{div};\Omega) &:= \left\{ \mathbf{v}(\mathbf{x}) \in (L^2(\Omega))^2, \nabla \cdot \mathbf{v} \in L^2(\Omega) \right\} \\
H_0(\mathrm{div};\Omega) &:= \left\{ \mathbf{v}(\mathbf{x}) \in H(\mathrm{div};\Omega) \;\middle|\; \mathbf{v}(\mathbf{x}) \cdot \mathbf{n}(\mathbf{x}) = 0,\ \mathbf{x} \in \Gamma \right\}.
\end{aligned} \tag{4.1}
$$

Multiplying the second equation in (1.1) by $\mu(c)\ \mathbf{K}^{-1}(\mathbf{x})$ yields

$$
\mu(c(\mathbf{x},t))\mathbf{K}^{-1}(\mathbf{x})\ \mathbf{u}(\mathbf{x},t) + \nabla p(\mathbf{x},t) = \rho\ g\ \nabla d(\mathbf{x}), \quad \mathbf{x} \in \Omega, \quad t \in [0,T]. \tag{4.2}
$$

Integrating (4.2) against any test functions $\mathbf{v} \in H_0(\mathrm{div};\Omega)$ and applying the divergence theorem to the ∇p term, we obtain the first equation in (4.2). Then integrating the first equation in (1.1) against any test functions $w(\mathbf{x}) \in L_0^2(\Omega)$, we obtain the second equation in the following system:

$$
\begin{aligned}
\int_\Omega \mu(c)\ \mathbf{K}^{-1}\ \mathbf{u} \cdot \mathbf{v}\ d\mathbf{x} - \int_\Omega p\ \nabla \cdot \mathbf{v}\ d\mathbf{x} &= \int_\Omega \rho\ g\ \nabla d \cdot \mathbf{v}\ d\mathbf{x}, \\
\int_\Omega w\ \nabla \cdot \mathbf{u}\ d\mathbf{x} &= \int_\Omega q(\mathbf{x},t)\ w\ d\mathbf{x} \\
\forall \mathbf{v}(\mathbf{x}) \in H_0(\mathrm{div};\Omega) \quad & \forall w(\mathbf{x}) \in L_0^2(\Omega), \quad t \in [0,T].
\end{aligned} \tag{4.3}
$$

Equation (4.3) is a saddle-point problem which has been proven [6] to have a unique solution $(\mathbf{u}(\mathbf{x},t), p(\mathbf{x},t)) \in H_0(\mathrm{div};\Omega) \times L_0^2(\Omega)\ \forall\ t \in [0,T]$.

4.2. An MFEM for (1.1). We define the following space-time partition for the pressure system (1.1):

$$
\begin{aligned}
0 =: &\ t_0^p < t_1^p < \cdots < t_m^p < \cdots < t_{M-1}^p < t_M^p && := T, \\
a_x =: &\ x_0^p < x_1^p < \cdots < x_k^p < \cdots < x_{K-1}^p < x_K^p && := b_x, \\
a_y =: &\ y_0^p < y_1^p < \cdots < y_l^p < \cdots < y_{L-1}^p < y_L^p && := b_y.
\end{aligned} \tag{4.4}
$$

At each time step t_m^p, we define the trial and test function spaces to be the lowest order Raviart–Thomas MFEM space on the partition (4.4)

$$
\begin{aligned}
S^p(\Omega) &:= \Big(M^p_{0,1}[a_x,b_x]\times M^p_{-1,0}[a_y,b_y]\Big)\times\Big(M^p_{-1,0}[a_x,b_x]\times M^p_{0,1}[a_y,b_y]\Big),\\
S^p_0(\Omega) &:= \Big\{\mathbf{v}(\mathbf{x})\in S^p(\Omega)\ \Big|\ \mathbf{v}(\mathbf{x})\cdot\mathbf{n}(\mathbf{x})=0,\quad \mathbf{x}\in\Gamma\Big\},\\
W^p(\Omega) &:= M^p_{-1,0}[a_x,b_x]\times M^p_{-1,0}[a_y,b_y],\\
W^p_0(\Omega) &:= \left\{v(\mathbf{x})\in W^p(\Omega)\ \Big|\ \int_\Omega v(\mathbf{x})d\mathbf{x}=0\right\},
\end{aligned}
\tag{4.5}
$$

with

$$
\begin{aligned}
M^p_{\alpha,\beta}[a_x,b_x] &:= \Big\{v\in C^\alpha[a_x,b_x]\ \Big|\ v|_{[x^p_{k-1},x^p_k]}\in P_\beta[x^p_{k-1},x^p_k],\ k=1,\dots,K\Big\},\\
M^p_{\alpha,\beta}[a_y,b_y] &:= \Big\{v\in C^\alpha[a_y,b_y]\ \Big|\ v|_{[y^p_{l-1},y^p_l]}\in P_\beta[y^p_{l-1},y^p_l],\ l=1,\dots,L\Big\}.
\end{aligned}
\tag{4.6}
$$

Here, $C^{-1}[a,b]$ is the space of piecewise continuous functions. $C^0[a,b]$ and $P_\beta[a,b]$ are defined in (3.11).

If we assume that $c(\mathbf{x},t_m^p)$ is known, an MFEM for the pressure equation (1.1) can be formulated as follows: Find $\mathbf{u}(\mathbf{x},t_m^p)\in S^p(\Omega)$ and $p(\mathbf{x},t_m^p)\in W_0^p(\Omega)$ such that

$$
\begin{aligned}
\int_\Omega \mu(c(\mathbf{x},t_m^p))\ \mathbf{K}^{-1}\ \mathbf{u}(\mathbf{x},t_m^p)\cdot\mathbf{v}\ d\mathbf{x}&\\
-\int_\Omega p(\mathbf{x},t_m^p)\ \nabla\cdot\mathbf{v}\ d\mathbf{x} &= \int_\Omega \rho\ g\ \nabla d\cdot\mathbf{v}\ d\mathbf{x}\quad \forall\mathbf{v}(\mathbf{x})\in S^p(\Omega),\\
\int_\Omega w\ \nabla\cdot\mathbf{u}(\mathbf{x},t_m^p)\ d\mathbf{x} &= \int_\Omega q(\mathbf{x},t_m^p)\ w\ d\mathbf{x}\quad \forall w(\mathbf{x})\in W_0^p(\Omega).
\end{aligned}
\tag{4.7}
$$

Remark 4.1. In the pressure system (1.1), the sharp changes of the permeability tensor $\mathbf{K}(\mathbf{x})$ and the viscosity coefficient $\mu(c)$ across fluid interfaces often lead to large changes in the pressure gradient which, in a compensatory fashion, yield a fairly smooth Darcy velocity $\mathbf{u}$. By approximating both p and $\mathbf{u}$ from system (1.1) simultaneously, MFEMs generate more accurate Darcy velocities than standard FDMs or FEMs. Moreover, they conserve mass. Furthermore, the MFEM approximations to the Darcy velocity $\mathbf{u}(\mathbf{x},t_m^p)$ are particularly suited for the semianalytical characteristic tracking used in the ELLAM scheme (3.12) and guarantee that the tracking stays within the physical domain (refer to Remarks 3.3 and 3.4).

Remark 4.2. While they possess various numerical advantages, MFEMs generate an indefinite coefficient matrix for the discrete algebraic system and are more expensive to solve than standard FDMs or FEMs. Moreover, the MFEM function spaces for p and $\mathbf{u}$ must be chosen carefully, so that they satisfy the inf-sup stability condition. Extensive research has been conducted on the efficient solution of MFEM discrete systems and on the development of various generalized MFEM spaces [7]. Furthermore, an additional numerical difficulty for the MFEMs for porous medium flows is the effect of the singular source and sink terms $q(\mathbf{x},t)$ in (4.7) (see [15, 27]).

5. An ELLAM-MFEM sequential solution technique for the systems (1.1)–(1.2). Because of the couplings and nonlinearities of the PDEs in the systems (1.1)–(1.2), the effect of the point sources and sinks, the use of large grid-spacings, and the convection-diffusion feature of (1.2), fully coupled and fully implicit methods are commonly used in large-scale simulators for the systems (1.1)–(1.2). However, they are computationally expensive per time step and introduce excessive numerical diffusion

and greatly reduced accuracy with large time steps. Since its introduction by Douglas, Ewing, and Wheeler [12], the MFEM sequential solution technique has proven to be an efficient procedure for the simulation of miscible fluid flow processes in porous media. The MMOC-MFEM solution technique proposed by Ewing, Russell, and Wheeler [16] eliminates the possible nonphysical oscillations in the MFEM sequential techniques and allows larger time steps to be used in the simulation of the systems (1.1)–(1.2). However, the MMOC-MFEM sequential method fails to conserve mass and has difficulties in treating boundary conditions (recall Remark 3.4).

In this section, we present an ELLAM-MFEM solution technique for the systems (1.1)–(1.2), in which we use the ELLAM scheme (3.12) for (1.2) and the MFEM (4.7) for the pressure system (1.1). In the miscible displacement of incompressible fluid flow processes in a porous medium reservoir, the Darcy velocity field $\mathbf{u}(\mathbf{x}, t)$ often changes less rapidly than the concentration, even if characteristics are taken into account. Therefore, it is appropriate to use coarser spatial grids and time steps (4.4) for the pressure system (1.1) than the spatial grids (3.9) and time steps (3.1) for (1.2). We assume that the spatial grids (3.9) and the time steps (3.1) are obtained by subdividing the spatial grids and the time steps (4.4). Namely, there exist

$$
\begin{array}{l}
0 =: N_0 < N_1 < \cdots < N_m < \cdots < N_{M-1} < N_M := N, \\
0 =: I_0 \ < I_1 \ < \cdots < I_k \ < \cdots < I_{K-1} \ < I_K \ := I, \\
0 =: J_0 \ < J_1 \ < \cdots < J_l \ < \cdots < J_{L-1} \ < J_L \ := J,
\end{array} \tag{5.1}
$$

such that

$$
t^c_{N_m} = t^p_m, \quad x^c_{I_k} = x^p_k, \quad y^c_{J_l} = y^p_l, \quad 0 \le m \le M,\ 0 \le k \le K,\ 0 \le l \le L. \tag{5.2}
$$

For $n = N_{m-1}+1, N_{m-1}+2, \ldots, N_m$, the concentration time step t^c_n relates to the pressure time steps by $t^p_{m-1} < t^c_n \le t^p_m$. Thus, we require a velocity approximation for the ELLAM scheme (3.12) based on $\mathbf{u}(\mathbf{x}, t^p_m)$ and earlier values. We define a velocity approximation by an extrapolation

$$
\begin{aligned}
(E\mathbf{u})(\mathbf{x}, t^c_n) \ &:= \left(1 + \frac{t^c_n - t^p_{m-1}}{t^p_{m-1} - t^p_{m-2}}\right)\mathbf{u}(\mathbf{x}, t^p_{m-1}) - \frac{t^c_n - t^p_{m-1}}{t^p_{m-1} - t^p_{m-2}}\mathbf{u}(\mathbf{x}, t^p_{m-2}), \\
&\qquad n = N_{m-1}+1, N_{m-1}+2, \ldots, N_m, \quad m = 2, 3, \ldots, M, \\
(E\mathbf{u})(\mathbf{x}, t^c_n) \ &:= \mathbf{u}(\mathbf{x}, 0), \quad n = 1, 2, \ldots, N_1, \quad m = 1.
\end{aligned} \tag{5.3}
$$

Equivalently, the first equation in (5.3) can be rewritten as

$$
\begin{aligned}
(E\mathbf{u})(\mathbf{x}, t^c_n) \ &:= \ \frac{t^p_m - t^c_n}{t^p_m - t^p_{m-1}}\mathbf{u}(\mathbf{x}, t^p_{m-1}) + \frac{t^c_n - t^p_{m-1}}{t^p_m - t^p_{m-1}}(E\mathbf{u})(\mathbf{x}, t^p_m), \\
&\qquad n = N_{m-1}+1, N_{m-1}+2, \ldots, N_m, \quad m = 2, 3, \ldots, M,
\end{aligned} \tag{5.4}
$$

where $(E\mathbf{u})(\mathbf{x}, t^p_m)$ is evaluated by (5.3) with $t^c_n = t^p_m$ or $n = N_m$.

We are now in a position to present an ELLAM-MFEM sequential decoupling and linearization solution technique for the coupled systems (1.1)–(1.2).

Initialization:
for $m = 0$ and $n = 0$ **do**

A1: Define $c(\mathbf{x}, 0)$ to be the L^2-projection of $c_0(\mathbf{x})$, which is given in (1.6), to the finite element subspace $S^c(\Omega)$: find $c(\mathbf{x}, 0) \in S^c(\Omega)$, such that

$$\int_\Omega c(\mathbf{x}, 0)v(\mathbf{x})d\mathbf{x} = \int_\Omega c_0(\mathbf{x})v(\mathbf{x})d\mathbf{x} \quad \forall v(\mathbf{x}) \in S^c(\Omega). \tag{5.5}$$

A2: With $c(\mathbf{x}, 0)$ obtained from (5.5), find $\mathbf{u}(\mathbf{x}, 0) \in S^p(\Omega)$ and $p(\mathbf{x}, 0) \in W_0^p(\Omega)$, such that

$$\begin{aligned} &\int_\Omega \mu(c(\mathbf{x}, 0))\ \mathbf{K}^{-1}\ \mathbf{u}(\mathbf{x}, 0) \cdot \mathbf{v}\ d\mathbf{x} \\ &\qquad -\int_\Omega p(\mathbf{x}, 0)\ \nabla \cdot \mathbf{v}\ d\mathbf{x} \quad = \quad \int_\Omega \rho\ g\ \nabla d \cdot \mathbf{v}\ d\mathbf{x} \quad \forall \mathbf{v}(\mathbf{x}) \in S^p(\Omega), \\ &\qquad \int_\Omega w\ \nabla \cdot \mathbf{u}(\mathbf{x}, 0)\ d\mathbf{x} \quad = \quad \int_\Omega q(\mathbf{x}, 0)\ w\ d\mathbf{x} \quad \forall w(\mathbf{x}) \in W_0^p(\Omega). \end{aligned} \tag{5.6}$$

end do
for $m = 1, 2, \ldots, M$ **do**

B1: **for** $n = N_{m-1} + 1, N_{m-1} + 2, \ldots, N_m$ **do**
—Find $c(\mathbf{x}, t_n^c) \in S^c(\Omega)$, such that for $(E\mathbf{u})(\mathbf{x}, t_n^c)$ given in (5.3)

$$\begin{aligned} &\int_\Omega \phi(\mathbf{x})c(\mathbf{x}, t_n^c)z(\mathbf{x}, t_n^c)\ d\mathbf{x} + \Delta t_n^c \int_\Omega\ \nabla z(\mathbf{x}, t_n^c) \cdot \mathbf{D}(\mathbf{x}, (E\mathbf{u})(\mathbf{x}, t_n^c))\nabla c(\mathbf{x}, t_n^c)\ d\mathbf{x} \\ &= \int_\Omega \phi(\mathbf{x})c(\mathbf{x}, t_{n-1}^c)z(\mathbf{x}, t_{n-1}^{c,+})\ d\mathbf{x} + \Delta t_n^c \int_\Omega \bar{c}(\mathbf{x}, t_n^c)\ q(\mathbf{x}, t_n^c)z(\mathbf{x}, t_n^c)\ d\mathbf{x} \\ &\qquad\qquad \forall \mathbf{z}(\mathbf{x}, t_n^c) \in S^c(\Omega). \end{aligned} \tag{5.7}$$

end do

B2: Since $t_{N_m}^c = t_m^p$ and $c(\mathbf{x}, t_m^p) = c(\mathbf{x}, t_{N_m}^c)$ is obtained from (5.7) with $n = N_m$, solve the following equation for $\mathbf{u}(\mathbf{x}, t_m^p) \in S^p(\Omega)$ and $p(\mathbf{x}, t_m^p) \in W_0^p(\Omega)$:

$$\begin{aligned} &\int_\Omega \mu(c(\mathbf{x}, t_m^p))\ \mathbf{K}^{-1}\ \mathbf{u}(\mathbf{x}, t_m^p) \cdot \mathbf{v}\ d\mathbf{x} \\ &\qquad -\int_\Omega p(\mathbf{x}, t_m^p)\ \nabla \cdot \mathbf{v}\ d\mathbf{x} \quad = \quad \int_\Omega \rho\ g\ \nabla d \cdot \mathbf{v}\ d\mathbf{x} \quad \forall \mathbf{v}(\mathbf{x}) \in S^p(\Omega), \\ &\qquad \int_\Omega w\ \nabla \cdot \mathbf{u}(\mathbf{x}, t_m^p)\ d\mathbf{x} \quad = \quad \int_\Omega q(\mathbf{x}, t_m^p)\ w\ d\mathbf{x} \quad \forall w(\mathbf{x}) \in W_0^p(\Omega). \end{aligned} \tag{5.8}$$

end do

Remark 5.1. The ELLAM-MFEM solution technique inherits all the numerical advantages of the ELLAM schemes for linear transport PDEs and the MMOC-MFEM sequential method, while overcoming the shortcomings of the MMOC-MFEM method. It significantly reduces the temporal errors and thus generates accurate numerical solutions even if very large time steps Δt_n^c are used. It symmetrizes (1.2) and yields a symmetric and positive-definite coefficient matrix with a condition number of $\mathcal{O}(1 + |\mathbf{D}|\Delta t_n^c/h_c^2)$, where

$$\begin{aligned} h_c &:= \min_{\{i=1,\ldots,I;\ j=1,\ldots,J\}} \{\Delta x_c := x_i^c - x_{i-1}^c,\ \Delta y_c := y_j^c - y_{j-1}^c\}, \\ h_p &:= \min_{\{k=1,\ldots,K;\ l=1,\ldots,L\}} \{\Delta x_p := x_k^p - x_{k-1}^p,\ \Delta y_p := y_l^p - y_{l-1}^p\}. \end{aligned} \tag{5.9}$$

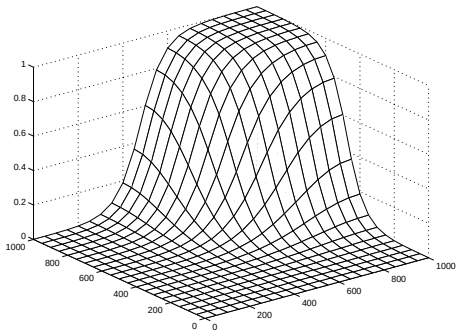
Remark 5.2. The MFEM (4.7) has an indefinite symmetric coefficient matrix with a condition number of $\mathcal{O}(h_p^{-2})$. Hence, it is much more expensive to solve than the ELLAM scheme (3.12), even if preconditioning techniques are used. On the other hand, by utilizing the flow property that the Darcy velocity field $\mathbf{u}$ changes less rapidly than the concentration c, the ELLAM-MFEM (as well as the MFEM-based and the MMOC-MFEM) solution technique allows much coarser spatial grids and time steps for the pressure system (1.1) to be used in a simulation without loss of accuracy. The application of the coarser spatial grids for the pressure reduces the size of the coefficient matrix and therefore its condition number, while the use of coarser time steps reduces the number of times the MFEM (4.7) needs to be solved. In this way, the ELLAM-MFEM sequential technique further enhances the computational efficiency.

Remark 5.3. The ELLAM-MFEM solution technique could be iterated at each pressure time step t_m^p. This would use the computed Darcy velocity $\mathbf{u}(\mathbf{x}, t_m^p)$ from (5.8) to replace the extrapolated Darcy velocity $(E\mathbf{u})(\mathbf{x}, t_m^p)$ in (5.4) in computing $(E\mathbf{u})(\mathbf{x}, t_n^c)$. Then the new velocity $(E\mathbf{u})(\mathbf{x}, t_n^c)$ would be used in the step B1. The present sequential procedure would be considered the first iteration. This was not attempted in this study. An iterated ELLAM-MFEM sequential solution procedure will probably be needed in more complicated applications.

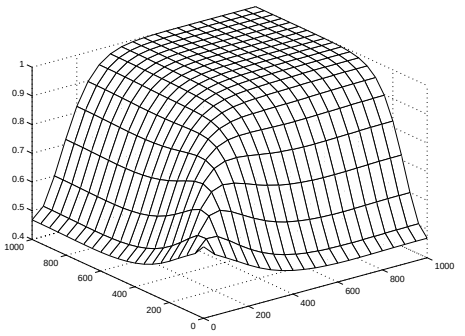
6. Numerical experiments. In this section, we apply the ELLAM-MFEM solution technique to a variety of two-dimensional miscible displacement problems of one incompressible fluid by another in porous media to examine its performance. The test runs include problems with large mobility ratios, anisotropic dispersion in tensor form, discontinuous permeabilities and porosities, and point sources and sinks. In the numerical experiments, we tried to choose test problems with reported data and results in the literature, to justify that the ELLAM-MFEM solution technique generates correct solutions, and to have some approximate comparisons of the ELLAM-MFEM technique with other widely used methods in terms of the spatial grids and time steps used in order to generate comparable solutions.

The numerical experiments simulated miscible displacement within a horizontal reservoir of a thickness of one unit over a period of 10 years (3600 days) for one-quarter of a regular five-spot pattern with injection and production wells at the corners. The spatial domain is $\Omega = (0, 1000) \times (0, 1000)$ ft^2, the time period $[0, T] = [0, 3600]$ days, and the viscosity of the oil is $\mu(0) = 1.0$ cp. The injection well is located at the upper-right corner $(1000, 1000)$ of the domain with an injection rate of $q = 30$ ft^2/day and an injection concentration of $\bar{c} = 1.0$. The production well is located at the lower-left corner $(0, 0)$ with a production rate of $q = 30$ ft^2/day. The initial concentration is $c_0(x, y) = 0$. In the numerical simulation, we use a fairly coarse uniform spatial grid of $\Delta x^c = \Delta y^c = \Delta x^p = \Delta y^p = 50$ ft in both x and y directions, although we understand that a simulation on a nonuniform partition with finer cells around wells probably generate more accurate solutions and our simulator allows such a partition. We also take an extremely large time step of $\Delta t^c = \Delta t^p = 360$ days (*one year*). In contrast, in the numerical results reported previously in the literature, the time steps were chosen from *a few days* for FDM or FEM simulators to about *a month* for MMOC-based simulators [16, 15, 25].

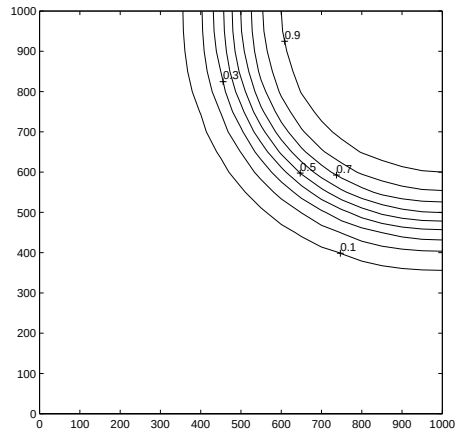
Test 1. We assume that the porous medium is homogeneous and isotropic. The permeability coefficients (diagonal entries of $\mathbf{K}$) are given by $k_x = k_y = 80$ md, and the porosity of the medium is specified as $\phi = 0.1$. Furthermore, we assume that the mobility ratio between the resident and injected fluids is $M = 1$ and that the physical diffusion-dispersion term is given by $D_m := \phi\, d_m = 1.0$ ft^2/day, $D_l := \phi\, d_l = 0.0$ ft,



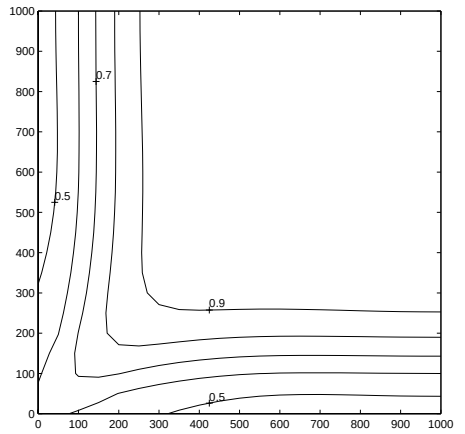
(a) *Surface plot at* $t = 3$ *years.*



(c) *Surface plot at* $t = 10$ *years.*



(b) *Contour plot at* $t = 3$ *years.*



(d) *Contour plot at* $t = 10$ *years.*

FIG. 1. *The concentration of the invading component in Test* 1 *at* 3 *and* 10 *years.*

and $D_t := \phi\ d_t = 0.0$ ft. Namely, only molecular diffusion is present. This example has been widely used in testing the performance of a simulator since the qualitative behavior of its numerical simulation is understood fairly well.

The surface and contour plots for the concentration of the invading fluid at $t =$ 3 years (1080 days) are presented in Figure 1 (a) and (b). The contours of the concentration are a family of concentric circles, which are physically reasonable due to the following reasons: (i) Mobility ratio $M = 1$ implies that the fluid has a constant viscosity $\mu(c) = \mu(0)$. (ii) Together with the facts that $\mathbf{K}$ is a constant tensor and that the reservoir is horizontal, we see that the Darcy velocity $\mathbf{u}$ is actually radial. (iii) Only the molecular diffusion, which is isotropic, is present. In fact, since the model does not include any permeability/viscosity variations or mechanical dispersion effects, any fingering phenomenon, if it happened, would be due to numerical errors and not to the modeling of any physics. Because of the effect of the no-flow boundary conditions and the production wells, the invading fluid moves faster along the diagonal (flow direction) of the reservoir with no fingering phenomena present. This was observed from the surface and contour plots, Figure 1 (c) and (d), of the concentration of the invading fluid at $t = 10$ years (3600 days). These results demonstrate that even

though extremely large time steps have been used in the simulation, the ELLAM-MFEM simulator still generates accurate and physically reasonable solutions.

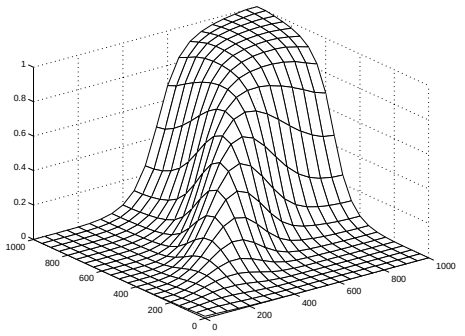
To investigate the mass conservation property of the ELLAM-MFEM solution technique, we calculate the mass balance error numerically. We divide the difference of the left-hand and right-hand sides of (3.13) by the mass $\int_\Omega \phi(\mathbf{x})c(\mathbf{x}, t_n^c)d\mathbf{x}$ at the time t_n^c. At the production well, $\bar{c}(\mathbf{x}, t) = c(\mathbf{x}, t)$. We use a trapezoidal quadrature to evaluate the temporal integral. In the 10-year simulation, the mass balance error is 1.99×10^{-4}. With a refined time step of $\Delta t/10$, the mass balance error is reduced to 2.05×10^{-5}. To indicate the accuracy of the ELLAM-MFEM solution technique, we compare the numerical solution with the solution obtained with a refined grid of $\Delta t/10$ and $\Delta x/2 = \Delta y/2$. The difference in the L^∞- and L^1-norms is 5.71×10^{-2} and 4.88×10^{-3}, respectively.

Test 2. We consider a simulation with an adverse mobility ratio of $M = 41$ and an anisotropic dispersion in tensor form. The physical diffusion-dispersion term is given by $D_m := \phi\ d_m = 0.0$ ft^2/day, $D_l := \phi\ d_l = 5.0$ ft, and $D_t := \phi\ d_t = 0.5$ ft. The permeability and porosity of the medium are taken to be the same as in Test 1. Equations (1.1)–(1.2) were derived via a volume-averaging mechanism and hold only on a macroscopic scale, so they do not model physical behavior on a pore-volume scale. Nevertheless, by including differences in longitudinal versus transverse dispersion levels (which can be viewed as a macroscopic reflection of the microscopic mechanism), (1.1)–(1.2) should model the corresponding (microscopic) behavior of the flow in the form of a macroscopic fingering phenomenon due to varying flow velocities. The macroscopic fingering phenomenon should propagate and grow in a manner akin to viscous fingering on a smaller scale. We refer readers to [15] for detailed descriptions on these concepts.

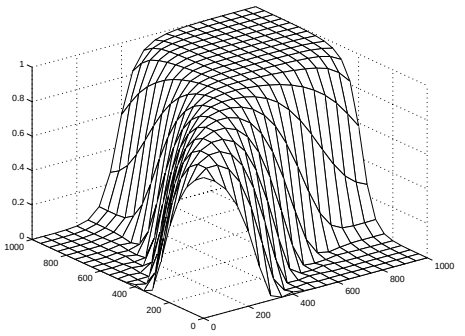
The surface and contour plots for the concentration of the invading fluid at $t = 3$ years (1080 days) and 10 years (3600 days) are presented in Figure 2 (a)–(b) and (c)–(d), respectively. Due to the effect of the large adverse mobility ratio $M = 41$, the viscosity $\mu(c)$ given by the expression (1.3) changes rapidly across the steep fluid interface. Consequently, the Darcy velocity $\mathbf{u}$ has a rapid change across the fluid interface. Moreover, the large differences in longitudinal versus transverse dispersion levels force the fluid flow to move much faster along the diagonal direction (flow direction) from the injection well to the production well. The concentration front moves much faster in the diagonal direction than it did when only the molecular diffusion term was present.

Test 3. We consider the numerical simulation of a miscible displacement in a porous medium with discontinuous permeabilities, which is often encountered in many field applications. In the example run, we use the same data as in Test 2, with an exception that $k_x = k_y = 80$ md is specified on the subdomain $\Omega_L := (0, 1000) \times (0, 500)$ (the lower half of the domain Ω) and that $k_x = k_y = 20$ md is given on the subdomain $\Omega_U := (0, 1000) \times (500, 1000)$ (the upper half of the domain Ω). The surface and contour plots for the concentration of the invading fluid at $t = 3$ years (1080 days) and 10 years (3600 days) are presented in Figure 3 (a)–(b) and (c)–(d), respectively.

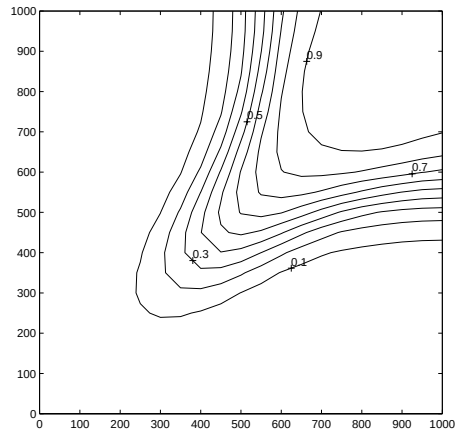
From Figure 3 (a)–(b), we see that the concentration front initially moves faster in the vertical direction than in the horizontal direction, because the subdomain Ω_L has a larger permeability and, thus, a larger Darcy velocity than the subdomain Ω_U. Once the invading fluid reaches Ω_L, it starts to move much faster in the horizontal direction on Ω_L than on Ω_U due to the same reason.



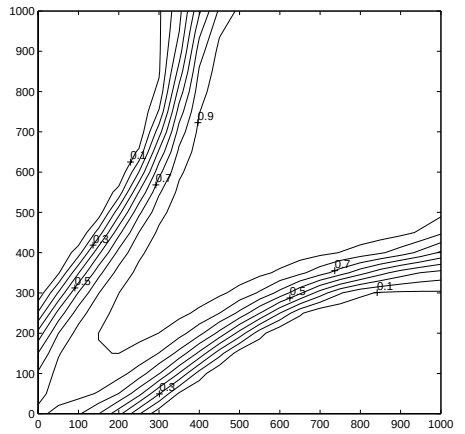
(a) *Surface plot at* $t = 3$ *years.*



(c) *Surface plot at* $t = 10$ *years.*



(b) *Contour plot at* $t = 3$ *years.*

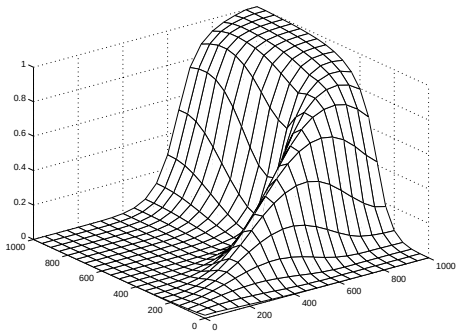


(d) *Contour plot at* $t = 10$ *years.*

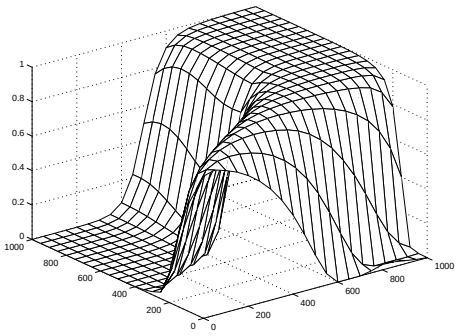
FIG. 2. *The concentration of the invading component in Test* 2 *at* 3 *and* 10 *years.*

Test 4. We consider a simulation in a porous medium with piecewise structures. On the subdomain $\Omega_I := (150, 550) \times (150, 550)$, we specify that the permeability of the medium is $k_x = k_y = 25$ md, that the porosity of the medium is $\phi = 0.09$, and that the physical diffusion-dispersion term is $D_m := \phi\ d_m = 0.0$ ft^2/day, $D_l := \phi\ d_l = 4.5$ ft, and $D_t := \phi\ d_t = 0.45$ ft. On the subdomain $\Omega_O := \Omega - \Omega_I$, we specify that the permeability of the medium is $k_x = k_y = 80$ md, that the porosity of the medium is $\phi = 0.1$, and that the physical diffusion-dispersion term is $D_m := \phi\ d_m = 0.0$ ft^2/day, $D_l := \phi\ d_l = 5.0$ ft, and $D_t := \phi\ d_t = 0.5$ ft. The mobility ratio is still given as $M = 41$. The surface and contour plots for the concentration of the invading fluid at $t = 3$ years (1080 days) and 5 years (1800 days) are presented in Figure 4 (a)–(d), while the corresponding plots at $t = 7$ years (2520 days) and 10 years (3600 days) are presented in Figure 4 (e)–(h).

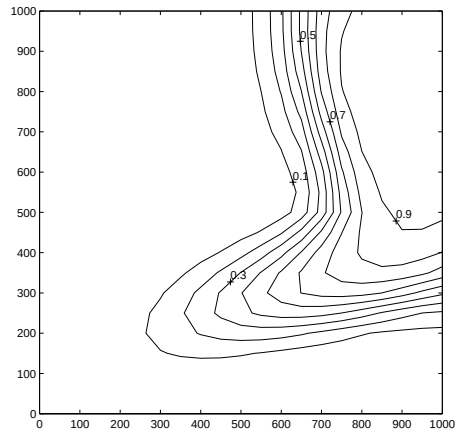
From these numerical results, we have the following observations: (i) The ELLAM-MFEM sequential solution technique can simulate fluid flows in porous media with fairly complex structures, and generates accurate and physically reasonable solutions even though a fairly coarse spatial grid and an extremely large time step are used in the simulation that in turn implies significantly improved computational efficiency.



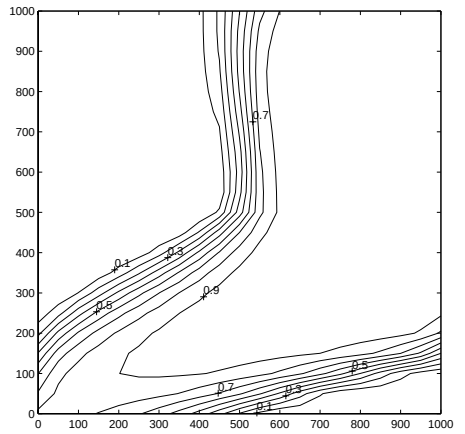
(a) *Surface plot at* $t = 3$ *years.*



(c) *Surface plot at* $t = 10$ *years.*



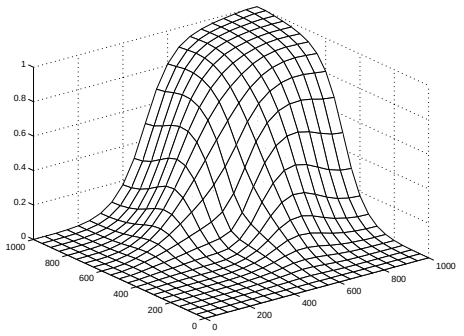
(b) *Contour plot at* $t = 3$ *years.*



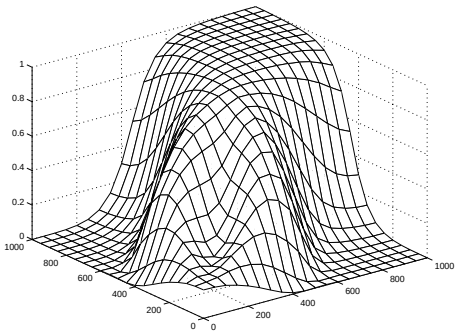
(d) *Contour plot at* $t = 10$ *years.*

FIG. 3. *The concentration of the invading component in Test* 3 *at* 3 *and* 10 *years.*

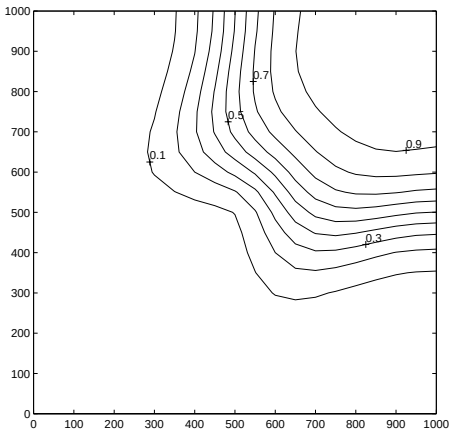
(ii) Comparing these results with those in Test 2, we see that if one has a choice, one should put the production well in a low permeability zone to increase the area swept by the injected fluid. This illustrates how the results of numerical simulations could help the decision making in the petroleum reservoir industry. (iii) A very important technique in enhanced oil recovery is the use of polymers in flooding processes to alter the permeability of the reservoir porous medium to allow the fluid to flow in certain ways. Since the polymers are highly viscous, they can be used to selectively block or reduce the permeabilities of certain pores or flow regions to direct the fluid flow in a manner to optimize the hydrocarbon recovery. Test 4 could also serve as a demonstration for this technique. In this case, one can view that the original fluid and porous medium properties are given as in Test 2. By injecting some polymers in some ways, one alters the properties of the medium to those given in this test. By comparing the corresponding results in Figures 2 and 4, we see that the injected fluid in Figure 4 swept a larger area, which in turn implies a larger output. Since a major cost in the petroleum industry is the cost of drilling (injection and production) wells, these results show the effect of enhanced oil recovery.



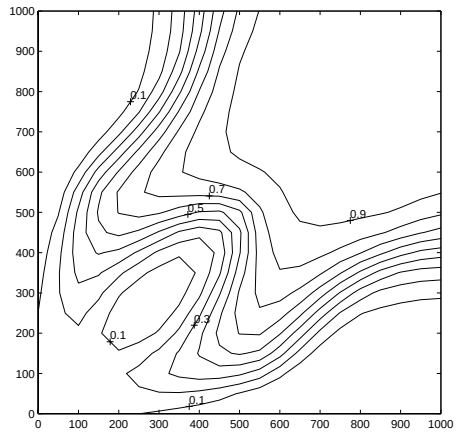
(a) *Surface plot at* $t = 3$ *years.*



(c) *Surface plot at* $t = 5$ *years.*



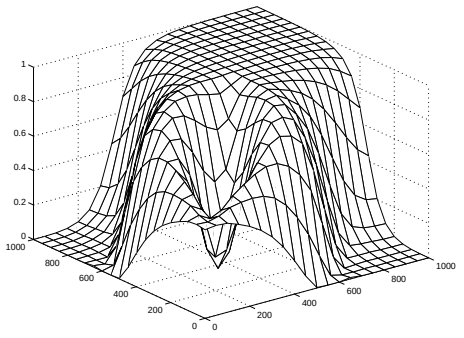
(b) *Contour plot at* $t = 3$ *years.*



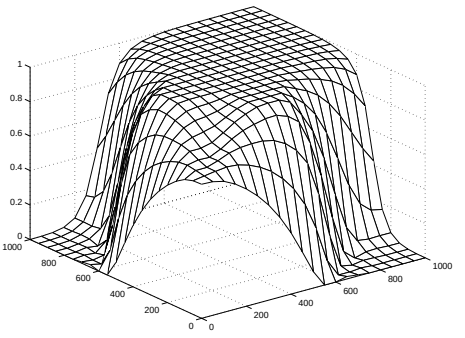
(d) *Contour plot at* $t = 5$ *years.*

FIG. 4. *The concentration of the invading component in Test* 4 *at* 3 *and* 5 *years.*

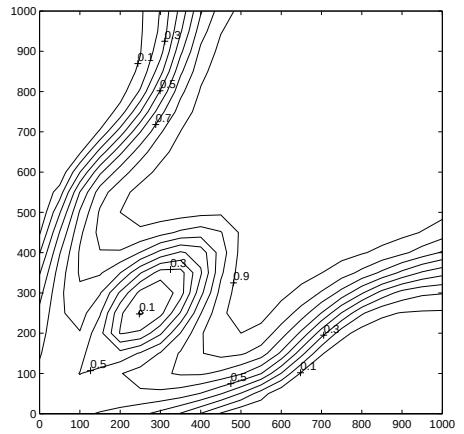
7. Summary and conclusions. We develop an ELLAM-MFEM sequential solution technique for miscible fluid flows in porous media with injection and production wells (point sources and sinks), in which we use an ELLAM to solve the transport equation for concentration and an MFEM to solve the pressure equation for the pressure and Darcy velocity. The ELLAM-MFEM solution technique significantly reduces the temporal truncation errors, and thus generates accurate numerical solutions even if fairly coarse spatial grids and very large time steps are used. It symmetrizes the transport equation and greatly reduces or eliminates nonphysical oscillation and/or excessive numerical dispersion present in many large-scale simulators that are widely used in industrial applications. In this manner, the ELLAM-MFEM solution technique has a greatly improved computational efficiency over many other methods. Furthermore, the ELLAM-MFEM solution technique conserves mass and treats boundary conditions accurately, and therefore overcomes all the shortcomings of the MMOC-MFEM method while maintaining its numerical advantages.



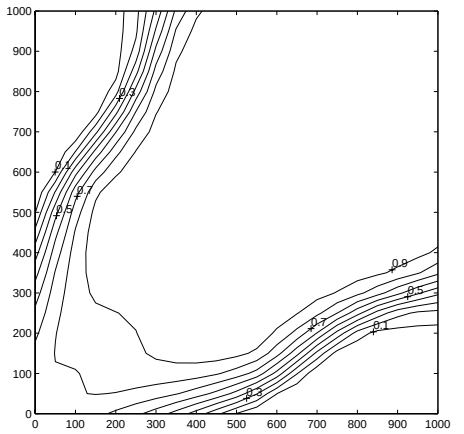
(e) *Surface plot at* $t = 7$ *years.*



(g) *Surface plot at* $t = 10$ *years.*



(f) *Contour plot at* $t = 7$ *years.*



(h) *Contour plot at* $t = 10$ *years.*

FIG. 4(CON'T.) *The concentration of the invading component in Test* 4 *at* 7 *and* 10 *years.*

Previous numerical experiments [1, 31] showed that the ELLAM often outperformed many widely used methods, such as the upwind FDM, various Galerkin and Petrov–Galerkin FEMs [2, 5, 10], the streamline diffusion FEMs [20, 21], and the MUSCL and minmod schemes [11, 14, 18, 29] in the context of linear transport PDEs. The present numerical experiments illustrate that the ELLAM-MFEM solution technique can simulate miscible displacements of incompressible fluid flows in porous media accurately with fairly coarse spatial grids as well as very large time steps, which are much larger than the time steps used in the MMOC-MFEM sequential solution procedure and one or two orders of magnitude larger than those used in many large-scale simulators. The ELLAM-MFEM technique can treat large mobility ratios, discontinuous permeabilities and porosities, anisotropic dispersion in tensor form, and point sources and sinks.

REFERENCES

[1] M. Al-Lawatia, R.C. Sharpley, and H. Wang, *Second-order characteristic methods for advection-diffusion equations and comparison to other schemes*, Advances in Water Resources, 22 (1999), pp. 741–768.

[2] J.W. Barrett and K.W. Morton, *Approximate symmetrization and Petrov-Galerkin methods for diffusion-convection problems*, Comput. Methods Appl. Mech. Engrg., 45 (1984), pp. 97–122.

[3] J. Bear, *Hydraulics of Groundwater*, McGraw-Hill, New York, 1979.

[4] J.P. Benque and J. Ronat, *Quelques difficultés des modeles numeriques en hydraulique*, in Computing Methods in Applied Sciences and Engineering VII, R. Glowinski and J.-L. Lions eds., North-Holland, Amsterdam, New York, 1982, pp. 471–494.

[5] E.T. Bouloutas and M.A. Celia, *An improved cubic Petrov-Galerkin method for simulation of transient advection-diffusion processes in rectangularly decomposable domains*, Comput. Methods Appl. Mech. Engrg., 91 (1991), pp. 289–308.

[6] F. Brezzi, *On the existence, uniqueness and approximation of saddle-point problems arising from Lagrangian multipliers*, RAIRO Modél. Math. Anal. Numér., 8 (1974), pp. 129–151.

[7] F. Brezzi and M. Fortin, *Mixed and Hybrid Finite Element Methods*, Springer Ser. Comput. Math. 15, Springer-Verlag, New York, 1991.

[8] M.A. Celia, T.F. Russell, I. Herrera, and R.E. Ewing, *An Eulerian-Lagrangian localized adjoint method for the advection-diffusion equation*, Advances in Water Resources, 13 (1990), pp. 187–206.

[9] G. Chavent, G. Cohen, J. Jaffre, R. Eymard, D.R. Guerillot, and L.Weiil, *Discontinuous and mixed finite elements for two-phase incompressible flow*, Society of Petroleum Engineers Reservoir Engineering, 5 (1990), pp. 567–575.

[10] I. Christie, D.F. Griffiths, A.R. Mitchell, and O.C. Zienkiewicz, *Finite element methods for second order differential equations with significant first derivatives*, Internat. J. Numer. Methods Engrg., 10 (1976), pp. 1389–1396.

[11] P. Colella, *A direct Eulerian MUSCL scheme for gas dynamics*, SIAM J. Sci. Statist. Comput., 6 (1985), pp. 104–117.

[12] J. Douglas, Jr., R.E. Ewing, and M.F. Wheeler, *A time-discretization procedure for a mixed finite element approximation of miscible displacement in porous media*, RAIRO Modél. Math. Anal. Numér., 17 (1983), pp. 249–265.

[13] J. Douglas, Jr. and T.F. Russell, *Numerical methods for convection-dominated diffusion problems based on combining the method of characteristics with finite element or finite difference procedures*, SIAM J. Numer. Anal., 19 (1982), pp. 871–885.

[14] B. Einfeldt, *On Godunov-type methods for gas dynamics*, SIAM J. Numer. Anal., 25 (1988), pp. 294–318.

[15] R.E. Ewing ed., *The Mathematics of Reservoir Simulation*, Research Frontiers in Appl. Math. 1, SIAM, Philadelphia, 1983.

[16] R.E. Ewing, T.F. Russell, and M.F. Wheeler, *Simulation of miscible displacement using mixed methods and a modified method of characteristics*, SPE 12241, in SPE Annual Conference Proceedings, 1983, pp. 71–81.

[17] R.E. Ewing and H. Wang, *Eulerian-Lagrangian localized adjoint methods for linear advection or advection-reaction equations and their convergence analysis*, Comput. Mech., 12 (1993), pp. 97–121.

[18] A. Harten, B. Engquist, S. Osher, and S. Chakravarthy, *Uniformly high order accurate essentially nonoscillatory schemes*, III, J. Comput. Phys., 71 (1987), pp. 231–241.

[19] R.W. Healy and T.F. Russell, *A finite-volume Eulerian-Lagrangian localized adjoint method for solution of the advection-dispersion equation*, Water Resources Research, 29 (1993), pp. 2399–2413.

[20] T.J.R. Hughes and A.N. Brooks, *A multidimensional upwind scheme with no crosswind diffusion*, in Finite Element Methods for Convection Dominated Flows, ADM 34, T.J.R. Hughes, ed., American Society of Mechanical Engineers (ASME), New York, 1979, pp. 19–35.

[21] C. Johnson, A. Szepessy, and P. Hansbo, *On the convergence of shock-capturing streamline diffusion finite element methods for hyperbolic conservation laws*, Math. Comp., 54 (1990), pp. 107–129.

[22] S.P. Neuman, *An Eulerian-Lagrangian numerical scheme for the dispersion-convection equation using conjugate space-time grids*, J. Comput. Phys., 41 (1981), pp. 270–294.

[23] O. Pironneau, *On the transport-diffusion algorithm and its application to the Navier-Stokes equations*, Numer. Math., 38 (1982), pp. 309–332.

[24] P.-A. Raviart and J.-M. Thomas, *A mixed finite element method for second order elliptic problems*, in Mathematical Aspects of Finite Element Methods, I. Galligani and E. Magenes eds., Lecture Notes in Math. 606, Springer–Verlag, Berlin, 1977, pp. 292–315.

[25] T.F. Russell, *Finite elements with characteristics for two-component incompressible miscible displacement*, SPE 10500, in Proceedings 6th SPE Symposium on Reservoir Simulation, New Orleans, 1982, pp. 123–135.

[26] T.F. Russell and R.V. Trujillo, *Eulerian-Lagrangian localized adjoint methods with variable coefficients in multiple dimensions*, in Computational Methods in Surface Hydrology, Proceedings of the 8th International Conference on Computational Methods in Water Resources, Venice, Italy, Springer-Verlag, Berlin, New York, 1990, pp. 357–363.

[27] T.F. Russell and M.F. Wheeler, *Finite element and finite difference methods for continuous flows in porous media*, in The Mathematics of Reservoir Simulation, Frontiers in Appl. Math. 1, R.E. Ewing, ed., SIAM, Philadelphia, 1983, pp. 35–106.

[28] A.L. Schafer-Perini and J.L. Wilson, *Efficient and accurate front tracking for two-dimensional groundwater flow models*, Water Resoures Research, 27 (1991), pp. 1471–1485.

[29] B. van Leer, *On the relation between the upwind-differencing schemes of Godunov, Engquist-Osher and Roe*, SIAM J. Sci. Statist. Comput., 5 (1984), pp. 1–20.

[30] E. Varoglu and W.D.L. Finn, *Finite elements incorporating characteristics for one-dimensional diffusion-convection equation*, J. Comput. Phys., 34 (1980), pp. 371–389.

[31] H. Wang, R.E. Ewing, G. Qin, S.L. Lyons, M. Al-Lawatia, and S. Man, *A family of Eulerian-Lagrangian localized adjoint methods for multi-dimensional advection-reaction equations*, J. Comput. Phys., 152 (1999), pp. 120–163.

SIAM J. SCI. COMPUT.
Vol. 22, No. 2, pp. 582–603
© 2000 Society for Industrial and Applied Mathematics

FOURIER ANALYSIS OF GMRES(m) PRECONDITIONED BY MULTIGRID*

ROMAN WIENANDS†, CORNELIS W. OOSTERLEE†, AND TAKUMI WASHIO‡

Abstract. This paper deals with convergence estimates of GMRES(m) [Saad and Schultz, *SIAM J. Sci. Statist. Comput.*, 7 (1986), pp. 856–869] preconditioned by multigrid [Brandt, *Math. Comp.*, 31 (1977), pp. 333–390], [Hackbusch, *Multi-Grid Methods and Applications*, Springer, Berlin, 1985]. Fourier analysis is a well-known and useful tool in the multigrid community for the prediction of two-grid convergence rates [Brandt, *Math. Comp.*, 31 (1977), pp. 333–390], [Stüben and Trottenberg, in *Multigrid Methods*, Lecture Notes in Math. 960, K. Stüben and U. Trottenberg, eds., Springer, Berlin, pp. 1–176]. This analysis is generalized here to the situation in which multigrid is a preconditioner, since it is possible to obtain the whole spectrum of the two-grid iteration matrix. A preconditioned Krylov subspace acceleration method like GMRES(m) implicitly builds up a minimal residual polynomial. The determination of the polynomial coefficients is easily possible and can be done explicitly since, from Fourier analysis, a simple block-diagonal two-grid iteration matrix results. Based on the GMRES(m) polynomial, sharp theoretical convergence estimates can be obtained which are compared with estimates based on the spectrum of the iteration matrix. Several numerical scalar test problems are computed in order to validate the theoretical predictions.

Key words. Fourier analysis, multigrid, restarted GMRES

AMS subject classifications. 65N55, 65N12, 65F10

PII. S1064827599353014

1. Introduction.

Nowadays, it has become popular to study the convergence of multilevel methods and to use them in combination with a Krylov subspace acceleration method, in other words, to use multilevel methods as a preconditioner. Often the cycling in a hierarchy of grids is simplified compared to standard multigrid cycling; i.e., the multiplicative cycling between the fine and coarse grids is replaced by an additive (simultaneous) cycling. It is then necessary to use this additive method as a preconditioner to obtain a converging method even for simple problems. Recently, standard (multiplicative) multigrid solvers [1], [8] have been used as preconditioners as well. This application of standard multigrid is beneficial in situations where standard multigrid alone does not converge fully satisfactorily, because certain error frequencies are not reduced well enough. This occurs mainly when complicated PDEs are solved. In general, it is difficult to construct *robust* multigrid solvers for large classes of problems. From this point of view, it makes sense to increase the class of problems for which proven efficient multigrid solvers exist by accelerating (standard) multigrid by a Krylov subspace method. Among many other papers, this approach has been presented in [11], where a symmetric multigrid method was accelerated by CG, in [3] for problems with fine level structures, in [13] for linear singularly perturbed problems, and in [14] for nonlinear problems.

Here, we theoretically analyze one combined solution method, restarted GMRES, GMRES(m), preconditioned by multigrid. We derive quantitative convergence results. The basis for the theoretical convergence estimates is Fourier analysis, which

*Received by the editors March 10, 1999; accepted for publication (in revised form) October 25, 1999; published electronically August 3, 2000.
http://www.siam.org/journals/sisc/22-2/35301.html
†GMD, Institute for Algorithms and Scientific Computing, D-53754 Sankt Augustin, Germany (wienands@gmd.de, oosterlee@gmd.de).
‡C&C Media Research Laboratories, NEC Cooperation, 1-1, Miyazaki 4-chome, Miyamae-ku, Kawasaki, Kanagawa 216-8555, Japan (washio@ccm.cl.nec.co.jp).

is a well-known tool for obtaining two-grid convergence factors (see [1], [19]). The typical use of the two-grid Fourier analysis in a multigrid context is that the spectral radius is obtained theoretically. It is the basis for asymptotic multigrid convergence estimates [19]. Two-grid Fourier analysis transforms a two-grid iteration matrix into a block-diagonal matrix from which more information about the spectrum is easily obtained. Fourier analysis uses unitary basis transformations to simplify the representation of the two-grid iteration matrix. It is therefore suitable for the estimate of the multigrid preconditioned GMRES(m) convergence since unitary transformations do not affect the convergence behavior of GMRES [15]. A preconditioned Krylov subspace acceleration method like GMRES(m) implicitly builds up a minimal residual polynomial. The determination of the polynomial coefficients is easily possible and can be done explicitly, since a block-diagonal matrix results with the help of the Fourier analysis.

In this paper, we restrict ourselves to standard problems coming from linear scalar PDEs discretized with finite differences on uniform Cartesian grids and we keep the multigrid method simple. For example, we restrict ourselves to point smoothers combined with standard grid coarsening and analyze the effect of Krylov subspace acceleration for anisotropic diffusion problems and for problems with mixed derivatives, although it is known that other (more expensive) smoothers or other coarsening strategies are appropriate remedies for the poor convergence of such equations. In this way we gain insight in the behavior of the combination of multigrid and GMRES(m).

In section 2 a multigrid preconditioner is presented and the rigorous two-grid Fourier analysis (see [19], [20]) is repeated briefly for 3D problems. The information on the multigrid preconditioner obtained is used in section 3, where different ways to analyze GMRES(m) preconditioned by multigrid are discussed.

There are actually two types of Fourier analysis available in the multigrid community. The first one is the analysis called "rigorous analysis" here, since it explicitly takes boundary conditions into account and discrete eigenfunctions are related to the discrete mesh size. Discrete sine functions are the basis in case Dirichlet boundary conditions are considered [19]. This analysis is, however, restricted to model situations, i.e., to symmetric problems and standard multigrid components. The second possibility is known as local mode analysis [1]. Exponential functions are the basis for the analysis. A discretization at domain boundaries cannot be taken into account. This analysis can be used for a wider range of PDEs and multigrid components. Local mode two-grid analysis and its generalization to multigrid preconditioned GMRES(m) is explained in section 4. Several numerical tests compare the theoretical convergence estimates with the actual numerical convergence.

For symmetric equations it is possible to construct a symmetric multigrid cycle and to apply CG as Krylov subspace acceleration method. We will not do this in section 3 because CG cannot be applied to the unsymmetric multigrid solvers MG-RB (multigrid solver based on pre- and post-smoothing by red-black Gauss–Seidel relaxation) from section 2 and MG-FF (multigrid solver based on pre- and post-smoothing by lexicographical Gauss–Seidel relaxation) from section 4. Furthermore, we are dealing with unsymmetric equations in sections 4.2 and 4.3. However, the analysis as described in section 3 carries over to an acceleration by CG in a straightforward manner. (See also the first remark in section 4.1.)

Furthermore, a relation to other techniques in which (multistage) parameters for smoothing methods, or acceleration parameters for coarse grid corrections [4], are optimized is presented in section 4.3. Such approaches can easily be viewed in the

framework of Krylov subspace acceleration.

The multigrid purist may not like the approach presented here with multigrid viewed as a preconditioner. It is, of course, also possible to try to construct an optimal multigrid method for each separate problem. For the purist, the analysis might give an indication as to how far his present method is from an optimal multigrid solution method. In other words, if an additional acceleration of the convergence is found by the Fourier analysis of multigrid preconditioned GMRES(m), it must also be possible to tune one of the multigrid components, so that multigrid as a stand alone solver is improved.

2. Rigorous Fourier analysis of multigrid. The Poisson equation, a simple test problem, is chosen to explain the possibilities of convergence estimates by Fourier analysis for multigrid. We apply the standard 5-point discretization on a square ($d = 2$) and the 7-point discretization on a cube ($d = 3$) with Dirichlet boundary conditions using a uniform mesh with meshsize $h = 1/n$:

$$\begin{aligned} A_h u_h(\boldsymbol{x}) = -\Delta_h u_h(\boldsymbol{x}) = b_h(\boldsymbol{x}) \quad &\text{on } \Omega_h = (0,1)^d \cap G_h, \\ u_h(\boldsymbol{x}) = 0 \quad &\text{on } \Gamma_h = ([0,1]^d \setminus \Omega_h) \cap G_h, \end{aligned} \tag{1}$$

with $G_h := \{\boldsymbol{x} = h\boldsymbol{j},\ \boldsymbol{j} \in \mathbb{Z}^d\}$. Equation (1), in stencil notation, looks like

$$A_h u_h(\boldsymbol{x}) = \sum_{\boldsymbol{\kappa} \in J} a_{\boldsymbol{\kappa}} u_h(\boldsymbol{x} + \boldsymbol{\kappa} h) = b_h(\boldsymbol{x}) \ \text{ on } \Omega_h, \tag{2}$$

with $J = \{(-1,0,0),(1,0,0),(0,-1,0),(0,1,0),(0,0,-1),(0,0,1),(0,0,0)\}$ ($d = 3$). We combine stencil, operator, and matrix notation. Multigrid is commonly explained with operators (see, for example, [19], [1]), whereas Krylov subspace methods are often described with matrices. An operator is denoted by a subscript "h" (K_h, Δ_h, M_h^{2h}, etc.) and the corresponding Fourier matrix representation is denoted by a tilde ($\widetilde{K}$, $\widetilde{M}$ etc.).

A well-known efficient multigrid method (see [19], [20]) for problem (1) consists of a red-black Gauss–Seidel smoothing method (GS-RB), full weighting of residuals to obtain the right-hand side on the coarse grids, bi- or tri-linear interpolation (in 2D and 3D, respectively) of corrections from coarse to fine grids, and the coarse grid discretization of the PDE on a grid with meshsize $2h$ in each direction. We call this method the red-black multigrid solver or MG-RB.

For the d-dimensional problem (1) and MG-RB, it is possible to apply rigorous two-grid Fourier analysis, as it was presented for 2D problems in [19] and used for 3D (Poisson-type) problems in [20]. We repeat the 3D analysis in some detail here.

The existence of the basis of discrete eigenfunctions

$$\begin{aligned} \varphi_h^{k,\ell,m}(x,y,z) &= \sin k\pi x \sin \ell\pi y \sin m\pi z, \\ \text{with } \ k,\ell,m &= 1,\ldots,n-1 \ \ ((x,y,z) = \boldsymbol{x} \in \Omega_h) \end{aligned} \tag{3}$$

for the operator (1) is crucial for the 3D discrete Fourier analysis. The scaled basis functions $\varphi_h^{k,\ell,m}(\boldsymbol{x})$ generate the space of all grid functions, $F(\Omega_h)$, and are orthogonal with respect to the discrete inner product on Ω_h:

$$(v_h, w_h) := h^3 \sum_{\boldsymbol{x} \in \Omega_h} v_h(\boldsymbol{x})\, w_h(\boldsymbol{x}) \quad \text{with } v_h, w_h \in F(\Omega_h).$$

The discrete solution u_h and the current approximation u_j can be represented by linear combinations of the basis functions $\varphi_h^{k,\ell,m}(\boldsymbol{x})$. The same holds for the error $v_{j-1} = u_{j-1} - u_h$ before and $v_j = u_j - u_h$ after the jth two-grid cycle. The error v_{j-1} is transformed by a two-grid cycle as follows:

$$v_j = M_h^{2h}\, v_{j-1} \quad \text{with} \quad M_h^{2h} = S_h^{\nu_2} K_h^{2h} S_h^{\nu_1} = S_h^{\nu_2}(I_h - I_{2h}^h(\Delta_{2h})^{-1} I_h^{2h}\Delta_h) S_h^{\nu_1}, \tag{4}$$

where S_h is the smoothing operator, ν_1 and ν_2 indicate the number of pre- and post-smoothing iterations, K_h^{2h} is the coarse grid correction operator, Δ_{2h} the discretization of the operator on a $2h$ coarse grid, I_{2h}^h and I_h^{2h} are transfer operators from coarse to fine grids, and vice versa.

In case of standard multigrid coarsening ($H = 2h$) in 3D, it is convenient to divide $F(\Omega_h)$ into a direct sum of (at most) eight-dimensional subspaces, the *2h-harmonics* [20]:

$$\begin{aligned} E_h^{k,\ell,m} = \operatorname{span}\Big[&\varphi_h^{k,\ell,m}, \varphi_h^{n-k,n-\ell,n-m}, \varphi_h^{n-k,\ell,m}, \varphi_h^{k,n-\ell,n-m}, \varphi_h^{k,n-\ell,m}, \\ &\varphi_h^{n-k,\ell,n-m}, \varphi_h^{k,\ell,n-m}, \varphi_h^{n-k,n-\ell,m}\Big] \quad \text{for} \quad k,\ell,m = 1,\ldots,\frac{n}{2}. \end{aligned} \tag{5}$$

This distinction is motivated by the observation that the *low-frequency* harmonic $\varphi_h^{k,\ell,m}$ ($1 \le k,\ell,m \le \frac{n}{2}$) is also visible on the coarse grid Ω_{2h}, whereas the (at most) seven corresponding *high-frequency* harmonics coincide (up to their sign) with $\varphi_h^{k,\ell,m}$. If one, two, or three of the indices equal $n/2$, the dimension of $E_h^{k,l,m}$ is four, two, or one, respectively. The coarse grid correction operator K_h^{2h} leaves the (at most) eight-dimensional spaces of *2h-harmonics* $E_h^{k,\ell,m}$ with an arbitrary $k,\ell,m = 1,\ldots,\frac{n}{2}$ invariant; see [20]:

$$K_h^{2h} \;:\; E_h^{k,\ell,m} \to E_h^{k,\ell,m} \quad k,\ell,m = 1,\ldots,\frac{n}{2}\,.$$

This is a consequence of the following relations of the transfer and discretization operators:

$$\begin{aligned} \Delta_h \quad &: \quad \operatorname{span}\,[\varphi_h^{k,\ell,m}] \to \operatorname{span}\,[\varphi_h^{k,\ell,m}], \\ I_h^{2h} \quad &: \quad E_h^{k,\ell,m} \to \operatorname{span}\,[\varphi_{2h}^{k,\ell,m}], \\ \Delta_{2h}^{-1} \quad &: \quad \operatorname{span}\,[\varphi_{2h}^{k,\ell,m}] \to \operatorname{span}\,[\varphi_{2h}^{k,\ell,m}], \\ I_{2h}^h \quad &: \quad \operatorname{span}\,[\varphi_{2h}^{k,\ell,m}] \to E_h^{k,\ell,m}. \end{aligned}$$

Because of this invariance property, the corresponding matrix representation of K_h^{2h} with respect to the spaces $E_h^{k,\ell,m}$ leads to a block-diagonal matrix $\widetilde{K}$:

$$\widetilde{K} := \Big[\widehat{K}(k,\ell,m)\Big]_{k,\ell,m=1,\ldots,\frac{n}{2}} \triangleq K_h^{2h} \quad \text{with} \quad \widehat{K}(k,\ell,m) \triangleq K_h^{2h}|_{E_h^{k,\ell,m}}.$$

The same invariance property is true for the GS-RB relaxation, $S_h = S_h^{BLACK} \cdot S_h^{RED}$ [20], where S_h^{RED} and S_h^{BLACK} are the partial step operators, leading to a matrix $\widetilde{S}$:

$$\widetilde{S} := \Big[\widehat{S}(k,\ell,m)\Big]_{k,\ell,m=1,\ldots,\frac{n}{2}} \triangleq S_h \quad \text{with} \quad \widehat{S}(k,\ell,m) \triangleq S_h|_{E_h^{k,\ell,m}}.$$

Hence, M_h^{2h} is orthogonally equivalent to a block matrix $\widetilde{M}$ consisting of (at most) 8×8 blocks (see [19], [20]):

$$\widetilde{M} := \Big[\widehat{M}(k,\ell,m)\Big]_{k,\ell,m=1,\ldots,\frac{n}{2}} \stackrel{\wedge}{=} M_h^{2h} \quad \text{with} \quad \widehat{M}(k,\ell,m) \stackrel{\wedge}{=} M_h^{2h}|_{E_h^{k,\ell,m}}.$$

The tilde $\widetilde{\ }$ indicates the *Fourier matrix representation* of the related operator. By M we denote the matrix representation of the multigrid operator M_h^{2h}, and by A we denote the matrix representation of Δ_h with respect to the Euclidean basis. We have

$$\widetilde{M} = UMU^{-1} \tag{6}$$

with a unitary matrix U. Eight consecutive rows of U are given by the $(n-1)^3$-dimensional orthogonal eigenvectors of A related to the eigenfunctions from (5):

$$\begin{aligned} &\varphi^{k,\ell,m},\ \varphi^{n-k,n-\ell,n-m},\ \varphi^{n-k,\ell,m},\ \varphi^{k,n-\ell,n-m},\ \varphi^{k,n-\ell,m},\ \varphi^{n-k,\ell,n-m}, \\ &\varphi^{k,\ell,n-m},\ \varphi^{n-k,n-\ell,m} \quad \text{for} \quad k,\ell,m = 1,\ldots,\frac{n}{2}, \quad \text{with, for example,} \\ &\varphi^{k,\ell,m} = (\ \sin k\pi h \sin \ell\pi h \sin m\pi h, \ldots, \sin k\pi(1-h)\sin \ell\pi(1-h)\sin m\pi(1-h)\ )^T. \end{aligned} \tag{7}$$

Using discrete Fourier frequencies $\boldsymbol{\theta} = \pi h(k,\ell,m)$ $(k,\ell,m = 1,\ldots,n-1)$, this gives

$$A\,\varphi^{k,\ell,m} = \lambda(\boldsymbol{\theta},h)\,\varphi^{k,\ell,m} \quad \text{with} \quad \lambda(\boldsymbol{\theta},h) = \sum_{\boldsymbol{\kappa}\in J} a_{\boldsymbol{\kappa}} \cos(\boldsymbol{\theta}\boldsymbol{\kappa}). \tag{8}$$

The dimension of $\widetilde{M}$ is given by $(n/2-1)^3\cdot 8 + 3(n/2-1)^2\cdot 4 + 3(n/2-1)\cdot 2 + 1\cdot 1 = (n-1)^3$ due to the dimensions of the $2h$-*harmonics* (5). The representation of $\widehat{M}(k,\ell,m) = \widehat{S}^{\nu_2}(k,\ell,m)\cdot\widehat{K}(k,\ell,m)\cdot\widehat{S}^{\nu_1}(k,\ell,m)$ $(k,\ell,m = 1,\ldots,\frac{n}{2})$ is obtained by straightforward generalization of the 2D case in [19]. From the block matrices $\widehat{M}(k,\ell,m)$, we obtain the two-grid convergence factor by

$$\rho_F := \rho(\widetilde{M}) = \max_{1\le k,\ell,m\le \frac{n}{2}} \rho(\widehat{M}(k,\ell,m)). \tag{9}$$

It is the asymptotic two-grid convergence which is a theoretical estimate of the multigrid convergence. Furthermore, it is possible to determine the whole spectrum and thus the eigenvalue distribution of $\widetilde{M}$ by calculating all eigenvalues of $\widehat{M}(k,\ell,m)$ for $k,\ell,m = 1,\ldots,\frac{n}{2}$. We use this distribution in section 3 for the analysis of the convergence of the multigrid preconditioned GMRES(m) method.

The *smoothing factor* μ_F is based on the ideal coarse grid correction operator Q_h^{2h} from [19] which annihilates the low-frequency error components and leaves the high-frequency components unchanged. Q_h^{2h} is a projection operator onto the space of high frequencies represented by a block-diagonal matrix $\widetilde{Q}$ consisting of (at most) 8×8 diagonal blocks: $\widehat{Q} = \mathrm{diag}(0,1,1,1,1,1,1,1)$. This leads to the following definition of the smoothing factor [19]:

$$\mu_F := \rho(\widetilde{S}^{\nu_2}\widetilde{Q}\widetilde{S}^{\nu_1}) = \rho(\widetilde{Q}\widetilde{S}^{\nu_1+\nu_2}) = \max_{1\le k,\ell,m\le \frac{n}{2}} \rho(\widehat{Q}\widehat{S}(k,\ell,m)^{\nu_1+\nu_2}). \tag{10}$$

The smoothing factor indicates the smoothing effect of the relaxation method under consideration. μ_F is a realistic estimate of the two-grid convergence factor ρ_F as long as the ideal coarse grid correction operator is a good approximation of K_h^{2h}.

2.1. Fourier results for multigrid. In [23] it was shown that an overrelaxation parameter ($\omega > 1$) improves the smoothing properties of GS-RB (leading to ω-GS-RB) and therefore improves the convergence of MG-RB for d-dimensional Poisson-type equations. The extra computational work for performing the overrelaxation in the smoother is worthwhile if $d = 3$. With the 3D Fourier two-grid and smoothing analysis explained above, we are able to estimate this convergence improvement for (1) very accurately; see Table 1. The average numerical convergence rate ρ_h^{mg} from MG-RB (after 100 multigrid iterations) is compared to ρ_F in Table 1. The number of multigrid levels used is 5, 6, and 6, respectively, for the 32^3, 64^3, and 96^3 problems.

TABLE 1
W(1,1)-cycle multigrid convergence ρ_h^{mg}, predicted convergence ρ_F, and smoothing factor μ_F for the 3D Poisson equation.

	$\omega = 1$			$\omega = 1.1$			$\omega = 1.15$ [23]		
Grid	ρ_h^{mg}	ρ_F	μ_F	ρ_h^{mg}	ρ_F	μ_F	ρ_h^{mg}	ρ_F	μ_F
32^3	0.192	0.194	0.194	0.089	0.091	0.090	0.070	0.072	0.088
64^3	0.196	0.197	0.197	0.091	0.092	0.092	0.074	0.074	0.088
96^3	0.196	0.198	0.197	0.091	0.093	0.093	0.074	0.075	0.088

In [23] it was shown furthermore that ω-GS-RB also improves the convergence factor of the 2D anisotropic diffusion equation,

$$A_h u_h(\boldsymbol{x}) = \frac{1}{h^2}\begin{bmatrix} & -1 & \\ -\varepsilon & 2\varepsilon+2 & -\varepsilon \\ & -1 & \end{bmatrix} u_h(\boldsymbol{x}) = b_h(\boldsymbol{x}) \ \text{ on } \Omega_h, \tag{11}$$
$$u_h(\boldsymbol{x}) = 0 \quad \text{on } \Gamma_h.$$

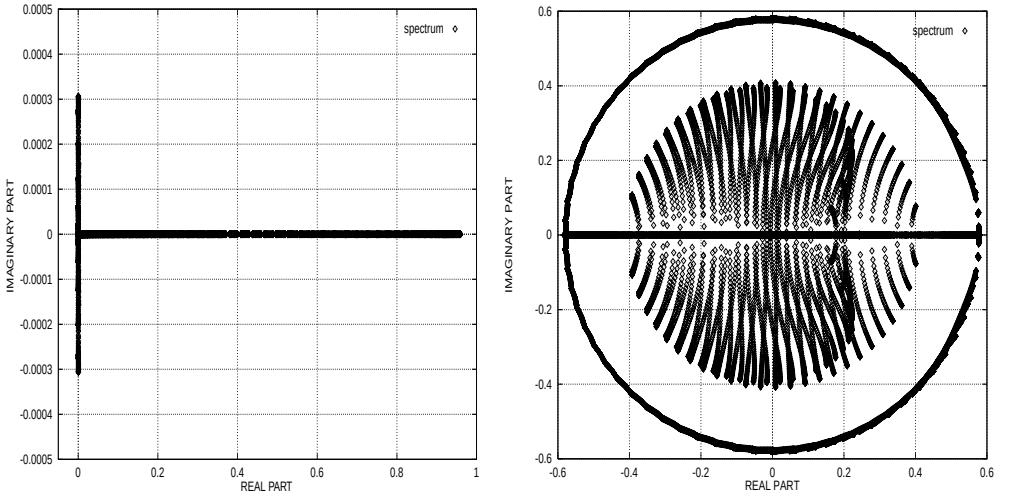
For moderate values of ε, like $100 \geq \varepsilon \geq 0.01$, the extra computational work pays off. Anisotropic problems can be solved with line smoothers in the multigrid process [19], by which much better convergence rates are obtained. However, line smoothers cannot be applied to unstructured grids in a straightforward way and their parallelization can be expensive. Here, we investigate the possibility to improve a multigrid method with a point smoother for moderate anisotropies with GMRES(m) acceleration. (Results are presented in the next sections.)

The rigorous Fourier analysis also applies for (11). In 2D we have to deal with (at most) four-dimensional $2h$-harmonics (see (5)) and therefore $\widetilde{M}$ consists of (at most) 4×4 blocks [19]. We perform numerical tests with $\varepsilon = 0.1$ and $\varepsilon = 0.01$. In both cases a large overrelaxation parameter (ω_{opt}=1.41, 1.76) has to be selected (for these values, see [23]) to obtain an impressive multigrid convergence improvement. The numerical multigrid convergence rates ρ_h^{mg} and the Fourier values μ_F and ρ_F are shown in Table 2. The W-cycle with $\nu_1 = \nu_2 = 1$ (W(1,1)) is chosen with seven multigrid levels for problem (11) discretized on a 128×128 grid.

TABLE 2
W(1,1)-cycle multigrid convergence ρ_h^{mg}, predicted convergence ρ_F, and smoothing factor μ_F for the 2D anisotropic diffusion equation, $h = 1/128$.

	$\omega = 1.0$			$\omega = \omega_{opt}$		
ε	ρ_h^{mg}	ρ_F	μ_F	ρ_h^{mg}	ρ_F	μ_F
$\varepsilon = 0.1$	0.679	0.682	0.682	0.193	0.210	0.219
$\varepsilon = 0.01$	0.957	0.960	0.960	0.566	0.583	0.590

In Figure 1, we show the effect of an overrelaxation parameter $\omega = 1.76$ on the eigenvalue distribution of the two-grid iteration matrix for the anisotropic ($\varepsilon = 0.01$) diffusion problem. In contrast to the almost real-valued spectrum for $\omega = 1.0$, which is uniformly distributed on the interval $[0.0, 0.960]$, we find complex eigenvalues clustered around the origin and uniformly distributed on a circle by the overrelaxation.



1 (a). $\omega = 1.0$ 1 (b). $\omega = 1.76$

FIG. 1. *Eigenvalue spectra of the red-black two-grid 2D anisotropic diffusion solver from rigorous Fourier analysis;* $\nu_1 = \nu_2 = 1, \varepsilon = 0.01, h = 1/128$. (*Note the different scaling of the axes in Figure* 1 (a).)

From the comparison of the predicted convergence rates ρ_F and the numerically obtained ρ_h^{mg}, it is clear that the two-grid Fourier analysis provides excellent quantitative estimates for the actual multigrid convergence of MG-RB for Poisson-type equations. These results encourage the following considerations concerning Fourier analysis of the multigrid preconditioned GMRES(m).

3. Fourier analysis of GMRES(m) preconditioned by multigrid. In this section we discuss two approaches for the analysis of GMRES(m) preconditioned by multigrid. For both approaches it is crucial that the discrete Fourier analysis uses unitary basis transformations (6) which simplify the representation of the multigrid iteration matrix.

We consider the linear system related to (2),

$$Au = b. \tag{12}$$

A two-grid (or more generally, a multigrid) cycle for solving (12) is described by the following matrix splitting:

$$Cu_j + (A - C)u_{j-1} = b, \tag{13}$$

where u_j represents a new and u_{j-1} a previous approximation. This formulation is equivalent to

$$u_j = u_{j-1} + C^{-1}(b - Au_{j-1}), \qquad r_j = (I - AC^{-1})r_{j-1}, \tag{14}$$

with r_j, r_{j-1} the residual vectors and $I - AC^{-1}$ represents the two-grid (multi-grid) iteration matrix.

The multigrid method (13) considered was described in the previous section. It is used as a *right preconditioner* for GMRES(m). The parameter m represents the number of search vectors stored after which the GMRES(m) algorithm will restart. GMRES(m) searches for a solution u_j in the following subspace:

$$\begin{aligned} C(u_j - u_{j-m}) &\in \operatorname{span}[r_{j-m}, (AC^{-1})r_{j-m}, \ldots, (AC^{-1})^{m-1}r_{j-m}] \\ &=: K^m(AC^{-1}, r_{j-m}), \end{aligned} \tag{15}$$

where $K^m(AC^{-1}, r_{j-m})$ is the Krylov subspace. It selects a new approximation u_j by minimizing the corresponding residual r_j in the L_2-norm

$$\min_{C(u_j - u_{j-m}) \in K^m(AC^{-1}, r_{j-m})} ||b - Au_j||_2. \tag{16}$$

Any element w in the affine subspace $u_{j-m} + C^{-1}K^m(AC^{-1}, r_{j-m})$ (see (14), (15)) can be represented by

$$w = u_{j-m} + C^{-1}(\alpha_1 r_{j-m} + \alpha_2 AC^{-1}r_{j-m} + \cdots + \alpha_m (AC^{-1})^{m-1} r_{j-m}). \tag{17}$$

By substituting (17) into the residual equation, we obtain

$$\begin{aligned} r_w &= b - Aw \\ &= r_{j-m} - \alpha_1 AC^{-1} r_{j-m} - \alpha_2 (AC^{-1})^2 r_{j-m} - \cdots - \alpha_m (AC^{-1})^m r_{j-m} \\ &= P_m(AC^{-1}) r_{j-m}, \end{aligned} \tag{18}$$

where P_m is an mth order polynomial defined by $P_m(\lambda) = 1 - \sum_{k=1}^{m} \alpha_k \lambda^k$. Comparing (16) and (18), we see that the L_2-norm of r_j satisfies the following minimization property:

$$||r_j||_2 = \min\{||P_m(AC^{-1})r_{j-m}||_2 \ \mid \ P_m \in \mathbb{P}_m\}, \tag{19}$$

where $\mathbb{P}_m$ denotes the set of all polynomials of degree at most m with $P_m(0) = 1$.

Since we are interested in the convergence of multigrid preconditioned GMRES with a restart after m iterations, we consider the residuals $r_m, r_{2m}, \ldots, r_{i \cdot m}$. This leads to the following definition of the reduction factors ρ_i for a *complete* ith iteration, consisting of m multigrid preconditioned GMRES(m) steps:

$$\rho_i := \left[\left(\frac{||r_{i \cdot m}||_2}{||r_0||_2} \right)^{1/m} \right]^{1/i}.$$

In subsection 3.1, we review estimates for ρ_i which are based on an analysis of the spectrum σ of AC^{-1} (see [5], [6], [15], [16], [17]). In general, these estimates exhibit a qualitative character, and concrete values are rarely found in the literature because the spectrum is usually not available. However, with the Fourier analysis these estimates can be calculated explicitly since the spectrum is easily obtained if multigrid is used as a preconditioner. In subsection 3.2 we introduce a sharper estimate, which is based on an analysis of the GMRES(m) polynomial.

3.1. Analysis with the spectrum of the iteration matrix. The usual way to analyze the convergence of GMRES is to exploit information about the spectrum of the iteration matrix AC^{-1}. If we compare the error transformation (4) and the residual transformation (14) by a two-grid cycle, the Fourier matrix $\widetilde{M}$ (6) is related to the two-grid iteration matrix $I - AC^{-1}$ via the identity

$$\widetilde{M} = UA^{-1}(I - AC^{-1})AU^{-1} \Longleftrightarrow \widetilde{M} = I - UC^{-1}AU^{-1}, \tag{20}$$

with the unitary matrix U from (6). With (20) we have

$$UAC^{-1}U^{-1} = I - UAU^{-1}\widetilde{M}(UAU^{-1})^{-1} = I - \widetilde{A}\widetilde{M}\widetilde{A}^{-1}. \tag{21}$$

$\widetilde{A}$ and $\widetilde{A}^{-1}$ are diagonal matrices since the rows of U are eigenvectors of A (see (7) and section 2). This means that $UAC^{-1}U^{-1}$ is a block-diagonal matrix which consists of (at most) 8×8 blocks in the 3D case and of (at most) 4×4 blocks in the 2D case. The eigenvalues of these block matrices are easily calculated.

We are dealing with positive real iteration matrices AC^{-1} (due to the multigrid preconditioner), so the symmetric part of AC^{-1} is positive definite. It is shown in [6], [5] that GMRES(m) always converges for this type of matrices. More generally, it is sufficient that the *field of values* of the iteration matrix $\{\bar{x}^T AC^{-1}x/\bar{x}^T x : x \in \mathbb{C}^{(n-1)^d}\}$ lies in an open half-plane $\{z : \mathrm{Re}(e^{-i\theta}z) > 0\}$. This is called the *half-plane condition* [12]. In particular the following error bound can be established for any restart parameter m and iteration index i [6]:

$$||r_{i\cdot m}||_2 \leq (1 - \alpha/\beta)^{m/2}||r_{(i-1)m}||_2, \tag{22}$$

with $\alpha = (\,\lambda_{min}(\frac{1}{2}(AC^{-1}+(AC^{-1})^T))\,)^2$ and $\beta = \lambda_{max}(\,(AC^{-1})^T AC^{-1}\,)$. Using (22), we obtain

$$||r_{i\cdot m}||_2 \leq (1 - \alpha/\beta)^{i\cdot m/2}||r_0||_2 \quad \Longrightarrow \quad \rho_i \leq (1 - \alpha/\beta)^{1/2} =: \rho_{hpc} < 1. \tag{23}$$

In general, the m- and i-independent value ρ_{hpc} is not a realistic quantitative estimate of ρ_i, but the residual bound from (23) is an important qualitative result because it ensures the convergence of GMRES(m) for all m.

Next, we discuss considerations on the convergence of GMRES(m) from [15], [16], [17], which are also based on the spectrum of the iteration matrix. From (19) it follows that

$$||r_{i\cdot m}||_2 \leq ||P_m(AC^{-1})||_2\,||r_{(i-1)m}||_2 \quad \text{for all} \quad P_m \in \mathbb{P}_m. \tag{24}$$

Suppose that all eigenvalues of AC^{-1} are located in an ellipse $E(c, d, a)$ which excludes the origin. c denotes the center, d the focal distance, and a the major semi-axis. Then it is known [17] that the absolute value of the polynomial

$$t_m(z) := T_m\Big(\frac{c}{d} - \frac{1}{d}\,z\Big)/T_m\Big(\frac{c}{d}\Big) = T_m(\widehat{z})/T_m\Big(\frac{c}{d}\Big) \quad \text{with} \quad z,\,\widehat{z} := \Big(\frac{c}{d} - \frac{1}{d}\,z\Big) \in \mathbb{C}$$

is small on the spectrum of AC^{-1}. Here, T_m represents the Chebychev polynomial of degree m of the first kind which is given by the three-term recurrence relation

$$T_0(\widehat{z}) = I\,, \quad T_1(\widehat{z}) = \widehat{z}\,, \quad T_{m+1}(\widehat{z}) = 2T_m(\widehat{z}) - T_{m-1}(\widehat{z}) \quad \text{for} \quad m \geq 1.$$

If AC^{-1} is diagonalizable, $AC^{-1} = XDX^{-1}$; then (24) yields

$$||r_{i\cdot m}||_2 \leq ||t_m(AC^{-1})||_2 \, ||r_{(i-1)m}||_2 \leq \left[||t_m(AC^{-1})||_2\right]^i \, ||r_0||_2 \tag{25}$$

$$\leq \left[||X||_2 \, ||X^{-1}||_2 \max_{\lambda_j \in \sigma} |t_m(\lambda_j)|\right]^i \, ||r_0||_2 \tag{26}$$

$$\leq \left[\kappa_2(X)\, T_m\left(\frac{a}{d}\right) / T_m\left(\frac{c}{d}\right)\right]^i \, ||r_0||_2, \tag{27}$$

where $\kappa_2(X)$ denotes the spectral condition number of the transformation matrix X (see [16], [17]). Inequality (27) uses the fact that the maximum modulus of a complex analytical function is attained on the boundary of the domain, which is due to Liouville's theorem [16]. Heuristically, we expect this inequality to be sharp if the interior of the ellipse $E(c, d, a)$ is well covered by the spectrum σ. Using (25) and (27), we obtain the following well-known estimates [16], [17] for ρ_i:

$$\rho_i \leq N_m^E \leq (\kappa_2(X))^{1/m} \, T_m^E, \tag{28}$$

$$\text{with} \quad N_m^E := (||t_m(AC^{-1})||_2)^{1/m} \quad \text{and} \quad T_m^E := \left(T_m\Big(\frac{a}{d}\Big)/T_m\Big(\frac{c}{d}\Big)\right)^{1/m}. \tag{29}$$

T_m^E can be further approximated by

$$T_m^E = \left(\frac{\left(\frac{a}{d} + \sqrt{(\frac{a}{d})^2 - 1}\right)^m + \left(\frac{a}{d} + \sqrt{(\frac{a}{d})^2 - 1}\right)^{-m}}{\left(\frac{c}{d} + \sqrt{(\frac{c}{d})^2 - 1}\right)^m + \left(\frac{c}{d} + \sqrt{(\frac{c}{d})^2 - 1}\right)^{-m}}\right)^{1/m} \approx \frac{a + \sqrt{a^2 - d^2}}{c + \sqrt{c^2 - d^2}}. \tag{30}$$

Approximation (30) is sharp if $a \gg d$ and $c \gg d$ or $m \gg 1$ holds. The first inequality is fulfilled if the ellipse $E(c, d, a)$ tends to a circle which means that the focal distance d becomes very small compared to a (in our tests it is usually sufficient if approximately $a \geq 2d$). In all cases considered, the multigrid preconditioner ensures that the spectrum σ is clustered around 1, which means that the second inequality is satisfied. If (30) is sharp due to a circular shape of σ or due to a large m, T_m^E is independent of the restart parameter m. In this case, it is difficult to further accelerate the multigrid method. A large Krylov subspace (15) does not lead to additional acceleration.

The main drawback of the estimates (28) for the reduction factors ρ_i is their i-independence. The relative amount of residual decrease from early GMRES cycles compared to later ones depends on both the matrix and the initial residual. The reduction by early cycles may be very different from the reduction by later ones. More precisely, the estimates from (28) are based on the worst previous residual that is possible due to the definition of the matrix norm $||A||_2 := \sup\{||Ar||_2/||r||_2 : r \in \mathbb{C}^{(n-1)^d}\}$. Thus, in general they cannot be expected to be sharp approximations for large i because $r_{(i-1)m}$ can be completely different from the worst residual since many iterations have already been performed and the worst search directions have already been removed from the current approximation $u_{(i-1)m}$ by GMRES(m). Furthermore, $\kappa_2(X)$ can be very large if the iteration matrix AC^{-1} is far from normal. But it is found in [12] that the condition number $\kappa_2(X)$ is not always relevant for the convergence of GMRES. In [17] it is stated that (28) is an asymptotic result and that the actual residual norm should behave like T_m^E without $\kappa_2(X)$. Therefore, the explicit values for T_m^E are also given in subsection 3.3 and compared with the asymptotic numerical convergence results. However, T_m^E is a heuristic estimate for $\rho_{i \gg 1}$ and not an upper bound like N_m^E.

3.2. Analysis with the GMRES-polynomial. To obtain a quantitative result for the convergence of GMRES(m), we present a second approach to estimate ρ_i, which explicitly depends on the iteration index i. In this way, it is possible to take the benefits of the previous iterations into account, in contrast to the above estimates. An initial residual r_0 is prescribed, and the GMRES(m)-polynomials P_m^i are explicitly determined for every i. The choice of r_0 does not influence the average reduction factor for $i \gg 1$ in which we are interested.

For any unitary matrix U, and in particular for U from (6), we have $||r_{i\cdot m}||_2 = ||Ur_{i\cdot m}||_2$. In order to find the coefficients α_k^i $(k = 1, \ldots, m)$ of the GMRES(m)-polynomials P_m^i, the following function f is minimized:

$$\begin{aligned} f(\alpha_1^i, \ldots, \alpha_m^i) :=& \Big(UP_m^i(AC^{-1})r_{(i-1)m}, UP_m^i(AC^{-1})r_{(i-1)m} \Big) \\ =& \Big(P_m^i(UAC^{-1}U^{-1})Ur_{(i-1)m}, P_m^i(UAC^{-1}U^{-1})Ur_{(i-1)m} \Big) \\ =& \sum_{l,k=1}^{m} \alpha_l^i \alpha_k^i \Big((UAC^{-1}U^{-1})^l Ur_{(i-1)m}, (UAC^{-1}U^{-1})^k Ur_{(i-1)m} \Big) \\ &+ 2\sum_{l=1}^{m} \alpha_l^i \Big((UAC^{-1}U^{-1})^l Ur_{(i-1)m}, Ur_{(i-1)m} \Big). \end{aligned}$$

The α_k^i are obtained by solving the following linear system of equations:

$$\begin{aligned} \frac{\partial f}{\partial \alpha_l^i} =& 2\sum_{k=1}^{m} \alpha_k^i \Big((UAC^{-1}U^{-1})^l Ur_{(i-1)m}, (UAC^{-1}U^{-1})^k Ur_{(i-1)m} \Big) \\ &+ 2\Big((UAC^{-1}U^{-1})^l Ur_{(i-1)m}, Ur_{(i-1)m} \Big) = 0 \quad \text{for } l = 1, \ldots, m. \end{aligned} \tag{31}$$

In (21) it is seen that the argument $UAC^{-1}U^{-1} = I - \widetilde{A}\widetilde{M}\widetilde{A}^{-1}$ of the GMRES(m)-polynomials P_m^i has a simple block structure. Therefore, due to the sparse structure of $(UAC^{-1}U^{-1})^l$ $(l = 1, \ldots, m)$, the solution of the linear system (31) is easily computed for every complete iteration i, as soon as the previous transformed residual $Ur_{(i-1)m}$ is given. We prescribe right-hand side $b = 0$ and a randomly chosen initial approximation Uu_0, which yield a transformed initial residual $Ur_0 = -\widetilde{A}Uu_0$. More precisely, we select an initial solution $u_0 = \sum_{k,\ell,m=1}^{n-1} \beta(k,\ell,m)\varphi^{k,\ell,m}$, where the amplitudes $\beta(k,\ell,m)$ of the eigenvectors are randomly chosen. Hence, we get

$$Uu_0 = \Big(\beta(1,1,1), \beta(n-1,n-1,n-1), \ldots, \beta\Big(\frac{n}{2}, \frac{n}{2}, \frac{n}{2}\Big) \Big)^T;$$

see (7). From Ur_0 it is possible to evaluate the α_k^1 by solving (31) for $i = 1$. This gives

$$Ur_m = P_m^1(UAC^{-1}U^{-1})Ur_0.$$

It follows that $Ur_{i\cdot m}$ is easily calculated for every i. This leads to the definition of an average reduction factor ρ_i^U obtained by the Fourier analysis:

$$\rho_i^U := \left[\left(\frac{||Ur_{i\cdot m}||_2}{||Ur_0||_2} \right)^{1/m} \right]^{1/i}. \tag{32}$$

A straightforward way to compute ρ_i^U is to run GMRES(m) directly for the prescribed residual r_0 and the transformed matrix $UACU^{-1}$. However, it turns out (see subsection 4.3) that an explicit calculation of the coefficients α_k^i $(k = 1, \ldots, m)$ of the GMRES(m)-polynomials gives an interesting insight into the behavior of GMRES(m).

In most of the tests, presented in the next subsection, ρ_i^U becomes constant for $i \geq 20$. Thus, ρ_{20}^U is expected to match numerical reference values very well and, in particular, to be sharper than the estimates from subsection 3.1.

3.3. Fourier results of multigrid preconditioned GMRES(m). Here, we investigate the quality of the theoretical considerations by comparing them to the numerical convergence of the multigrid preconditioned GMRES(m) method for the test problems from subsection 2.1. We compare the average accelerated multigrid convergence rate, ρ_h^{acc}, from numerical experiments with the theoretical prediction $\rho_{i=20}^U$, the heuristic estimate T_m^E, and the upper bound N_m^E. Insight into the behavior of the acceleration method is gained by considering different combinations of the relaxation parameter ω and the GMRES(m) restart parameter m.

In order to make sure that the asymptotic convergence behavior is observed, we present ρ_h^{acc} after 40 or after 100 multigrid preconditioned GMRES(m) iterations. These numbers of iterations match $\rho_{i=20}^U$ for $m = 2$ and for $m = 5$, respectively. However, the numerical convergence stabilizes much earlier and often the asymptotic values are obtained after only a few complete iterations.

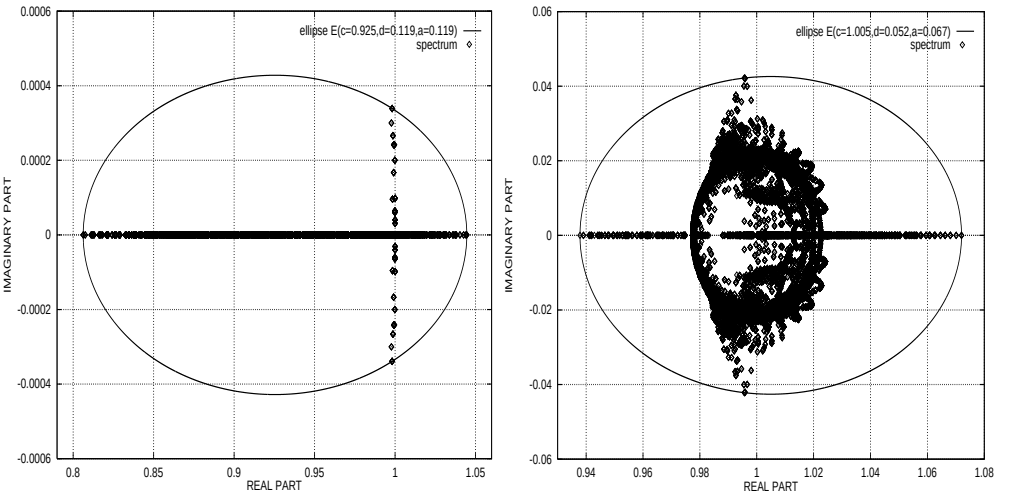
The results for the 3D Poisson equation with right-hand side $b_h(\boldsymbol{x}) = 0$ and initial guess $u_0 = (1, \ldots, 1)^T$ are presented in Table 3. We observe that $\rho_{i=20}^U$ and T_m^E provide accurate predictions of ρ_h^{acc} for different overrelaxation parameters ω. The upper bound N_m^E becomes sharper with increasing m as T_m^E and N_m^E coincide for large m, because the condition number $\kappa_2(X)$ cancels out (see (28)). Comparing Tables 1 and 3, we observe a satisfactory convergence improvement for $\omega = 1$. If overrelaxation is selected, the benefits of the GMRES(m) acceleration are small and the convergence cannot be improved significantly by using a larger Krylov space (15). This behavior is expected from the spectral picture in Figure 2. The ellipses become more orbital (note the different scaling of the axes) with the overrelaxation so that approximation (30) gets sharper.

TABLE 3

W(1,1)-cycle multigrid preconditioned GMRES(m) convergence ρ_h^{acc} and convergence estimates $\rho_{i=20}^U$, T_m^E, N_m^E for the 3D Poisson equation, $h = 1/32$.

Relaxation parameter	Acc. MG-RB, $m = 2$				Acc. MG-RB, $m = 5$			
	ρ_h^{acc}	$\rho_{i=20}^U$	T_m^E	N_m^E	ρ_h^{acc}	$\rho_{i=20}^U$	T_m^E	N_m^E
$\omega = 1.0$	0.085	0.090	0.091	0.186	0.070	0.072	0.074	0.115
$\omega = 1.1$	0.045	0.054	0.055	0.133	0.042	0.045	0.047	0.082
$\omega = 1.15$ [23]	0.050	0.054	0.056	0.129	0.045	0.049	0.055	0.077

Similar statements hold for the 2D anisotropic diffusion equation, where right-hand side $b_h(\boldsymbol{x}) = 0$ and initial guess $u_0 = (1, \ldots, 1)^T$ are chosen, as in the 3D case. Theoretical predictions $\rho_{i=20}^U$ match very well with ρ_h^{acc}, as can be seen from Table 4. In most cases the same observation can be made for the values T_m^E and ρ_h^{acc}. N_m^E is not a good approximation for ρ_h^{acc} but it improves with increasing m. If we choose an optimal overrelaxation parameter for $\varepsilon = 0.1$ and $\varepsilon = 0.01$, it is not really possible to further accelerate the ω-MG-RB method due to the circular shape of the corresponding spectrum σ shown in Figure 3. For the same reason further



2 (a). $\omega = 1.0$ 2 (b). $\omega = 1.15$

FIG. 2. *Eigenvalue spectra of the red-black two-grid 3D Poisson solver from Fourier analysis;* $\nu_1 = \nu_2 = 1$, $h = 1/32$. *(Note the different scaling of the axes.)*

TABLE 4

W(1,1)*-cycle multigrid preconditioned GMRES(m) convergence* ρ_h^{acc} *and convergence estimates* $\rho_{i=20}^U$, T_m^E, N_m^E *for the 2D anisotropic diffusion equation,* $h = 1/128$.

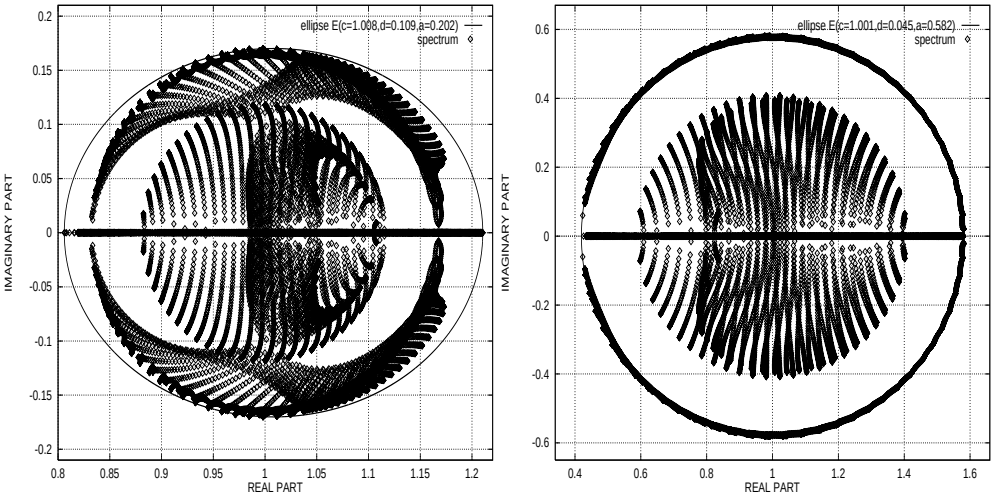
Acc.	$\varepsilon = 0.1$, $\omega = 1.0$				$\varepsilon = 0.1$, $\omega = 1.41$			
MG-RB	ρ_h^{acc}	$\rho_{i=20}^U$	T_m^E	N_m^E	ρ_h^{acc}	$\rho_{i=20}^U$	T_m^E	N_m^E
$m = 2$	0.350	0.392	0.405	0.734	0.173	0.184	0.186	0.724
$m = 5$	0.300	0.327	0.333	0.409	0.169	0.180	0.185	0.354

Acc.	$\varepsilon = 0.01$, $\omega = 1.0$				$\varepsilon = 0.01$, $\omega = 1.76$			
MG-RB	ρ_h^{acc}	$\rho_{i=20}^U$	T_m^E	N_m^E	ρ_h^{acc}	$\rho_{i=20}^U$	T_m^E	N_m^E
$m = 2$	0.795	0.800	0.861	1.460	0.568	0.579	0.580	2.385
$m = 5$	0.723	0.740	0.763	1.054	0.566	0.579	0.580	1.182

improvement cannot be achieved by a larger Krylov space. For $\omega = 1.0$ a satisfactory convergence improvement can be observed comparing Table 2 and Table 4.

As mentioned before, the Fourier analysis for multigrid is based on $\widetilde{M} = I - UC^{-1}AU^{-1}$ (see (20)), the *error* reduction in a two-grid algorithm, whereas Krylov methods minimize the *residual* norm and are based on AC^{-1}. Thus, we have for the corresponding spectra: $\sigma(AC^{-1}) = 1 - \sigma(\widetilde{M})$ (compare Figure 1 (b) and Figure 3 (b)). As a consequence the iteration matrices AC^{-1} are always positive real here because the spectrum of the multigrid iteration matrix $\sigma(\widetilde{M})$ is mainly clustered around the origin. In particular $\rho(\widetilde{M}) = \rho_F < 1$ holds because the multigrid preconditioner is already a convergent method. Thus, any eigenvalue of AC^{-1} lies in the right half-plane and we have $\rho_{hpc} < 1$.

Remark. In this connection it is interesting to note that the *residual* reduction norm $||\widetilde{A}\widetilde{M}\widetilde{A}^{-1}||_2$ decreases with an increasing number of post-relaxations ν_2 in contrast to the *error* reduction norm $||\widetilde{M}||_2$ which decreases with an increasing ν_1 [19]. According to this observation, one should choose similar values for ν_1 and ν_2 and prefer $\nu_1 \geq \nu_2$ to $\nu_1 \leq \nu_2$ when multigrid is selected as a solver [19]. The reversed



3 (a). $\varepsilon = 0.1$, $\omega = 1.41$ 3 (b). $\varepsilon = 0.01$, $\omega = 1.76$

FIG. 3. *Eigenvalue spectra of the red-black two-grid* $2D$ *anisotropic diffusion solver from Fourier analysis;* $\nu_1 = \nu_2 = 1$, $h = 1/128$.

statement (prefer $\nu_1 \leq \nu_2$ to $\nu_1 \geq \nu_2$) holds if we apply multigrid as a preconditioner for a Krylov acceleration technique.

From Tables 3 and 4, it follows that a significant acceleration of the ω-MG-RB method with the help of GMRES(m) is possible if $\omega = 1.0$. This acceleration is observed if we compare the numerical results ρ_h^{mg} and ρ_h^{acc} from subsections 2.1 and 3.3 and if we compare the Fourier estimates ρ_F and $\rho_{i=20}^U$. If, in some sense, an *optimal multigrid method* is selected by using the optimal overrelaxation parameter, a further improvement is almost impossible.

$\rho_{i=20}^U$ can be considered as a quantitative value which is found to be close to the numerical convergence rate ρ_h^{acc}. The heuristic estimate T_m^E gives good insight into the asymptotic convergence behavior, whereas N_m^E is too pessimistic an approximation in general. The qualitative value ρ_{hpc} is important for theoretical purposes, but it doesn't help in choosing between solution methods. In most cases ρ_{hpc} is close to 1 and far away from ρ_h^{acc}.

4. Local mode analysis of multigrid preconditioned GMRES(m). The Fourier analysis as presented in section 2 is restricted to model operators A_h, for which the discrete functions (3) form a basis of eigenfunctions. Furthermore, it is not possible to apply this analysis for multigrid methods based on Gauss–Seidel relaxations with a lexicographical ordering of the grid points (GS-LEX). The eigenfunctions (3) are no longer invariant under the GS-LEX operators. Instead they are intermixed, which leads to a full matrix $\widetilde{S}$. For a general linear operator A_h with constant or frozen coefficients it is, however, possible to use the *local mode Fourier analysis* [1] for analyzing multigrid methods based on GS-LEX as the smoother. The theoretical foundations (for example, presented in [18], [2]) for the local mode analysis are not as straightforward as they are for the analysis from section 2. Basically the local mode analysis is valid if the influence of a domain boundary is negligible [2]. Often, this requirement can be satisfied by performing extra local boundary relaxation. In practice, the local mode (or infinite grid) two-grid and smoothing analysis gives satisfactory

sharp estimates of the actual multigrid convergence.

Instead of considering the finite domain with discrete eigenfunctions (3) of A_h an infinite domain $\overline{\Omega}_h := \{\boldsymbol{x} = (j_x h, j_y h, j_z h) : j_x, j_y, j_z \in \mathbb{Z}\}$ is considered with continuous eigenfunctions

$$\phi_h(\boldsymbol{\theta}, \boldsymbol{x}) = e^{i\boldsymbol{x}\boldsymbol{\theta}/h} = e^{i\boldsymbol{j}\boldsymbol{\theta}} = e^{i(j_x\theta_x + j_y\theta_y + j_z\theta_z)}, \tag{33}$$

where the Fourier frequencies $\boldsymbol{\theta} = (\theta_x, \theta_y, \theta_z)$ vary continuously in $\mathbb{R}^3$. The corresponding eigenvalues are given by $\lambda(\boldsymbol{\theta}, h) = \sum_{\boldsymbol{\kappa}\in J} a_{\boldsymbol{\kappa}} e^{i\boldsymbol{\theta}\boldsymbol{\kappa}}$ (see (8)). Fourier components with $|\boldsymbol{\theta}| := \max\{|\theta_x|, |\theta_y|, |\theta_z|\} \geq \pi$ are not visible on $\overline{\Omega}_h$, since they coincide with components $e^{i\boldsymbol{j}\hat{\boldsymbol{\theta}}}$ where $\hat{\boldsymbol{\theta}} = \boldsymbol{\theta} (\mathrm{mod}\, \pi)$. Therefore, the Fourier space

$$\boldsymbol{\varepsilon}^h = \mathrm{span}\{e^{i\boldsymbol{j}\boldsymbol{\theta}} : \boldsymbol{\theta} \in \Theta = (-\pi, \pi]^3\}$$

contains any infinite grid function on $\overline{\Omega}_h$ [19]. As in the discrete Fourier analysis, we divide the Fourier space $\boldsymbol{\varepsilon}^h$ into eight-dimensional subspaces $\boldsymbol{\varepsilon}^h_{\boldsymbol{\theta}}$, which consist of one low-frequency harmonic ($\boldsymbol{\theta} \in \Theta^l := (-\pi/2, \pi/2]^3$) and seven corresponding high-frequency harmonics ($\boldsymbol{\theta} \in \Theta^h := \Theta \backslash \Theta^l$) (5):

$$\boldsymbol{\varepsilon}^h_{\boldsymbol{\theta}} = \mathrm{span}[\phi(\boldsymbol{\theta}^{\alpha_x\alpha_y\alpha_z}, \boldsymbol{x}) = e^{i\boldsymbol{j}\boldsymbol{\theta}^{\alpha_x\alpha_y\alpha_z}} : \alpha_x, \alpha_y, \alpha_z \in \{0, 1\}], \quad \text{with } \boldsymbol{x} \in \overline{\Omega}_h,$$
$$\boldsymbol{\theta}^{000} \in \Theta^l, \text{ and } \boldsymbol{\theta}^{\alpha_x\alpha_y\alpha_z} = (\theta_x - \alpha_x \mathrm{sign}(\theta_x)\pi, \theta_y - \alpha_y \mathrm{sign}(\theta_y)\pi, \theta_z - \alpha_z \mathrm{sign}(\theta_z)\pi).$$

These spaces $\boldsymbol{\varepsilon}^h_{\boldsymbol{\theta}}$ of $2h$-harmonics are invariant under the coarse grid correction operator K_h^{2h} (4) and also under a GS-LEX relaxation operator. In order to obtain a well-defined two-grid operator we replace the Fourier space $\boldsymbol{\varepsilon}^h$ by a slightly shrunk subspace $\tilde{\boldsymbol{\varepsilon}}^h$ such that $(A_h)^{-1}$ exists and also $(A_{2h})^{-1}$ can be reasonably defined on the coarser grid, as in [19]:

$$\tilde{\boldsymbol{\varepsilon}}^h = \boldsymbol{\varepsilon}^h \backslash \bigcup_{\boldsymbol{\theta}\in\Psi} \boldsymbol{\varepsilon}^h_{\boldsymbol{\theta}} \quad \text{with} \quad \Psi := \{\boldsymbol{\theta} : \lambda(\boldsymbol{\theta}, h) = 0 \text{ or } \lambda(2\boldsymbol{\theta}, 2h) = 0\}.$$

Hence, we have a well-defined operator M_h^{2h} (4) on $\tilde{\boldsymbol{\varepsilon}}^h$ with the invariance property

$$M_h^{2h} : \boldsymbol{\varepsilon}^h_{\boldsymbol{\theta}} \longrightarrow \boldsymbol{\varepsilon}^h_{\boldsymbol{\theta}} \quad \text{with} \quad \boldsymbol{\theta} \in \tilde{\Theta}^l := \Theta^l \backslash \Psi,$$

equivalent to a block-diagonal matrix $\widetilde{M}$. The spectral radius $\rho(\widetilde{M})$ of the corresponding Fourier matrix is determined similarly as in the previous Fourier analysis (9):

$$\rho_F := \rho(\widetilde{M}) = \sup_{\boldsymbol{\theta}\in\tilde{\Theta}^l} \rho(\widehat{M}(\boldsymbol{\theta})) \quad \text{with} \quad \widehat{M}(\boldsymbol{\theta}) \triangleq M_h^{2h}|_{\boldsymbol{\varepsilon}^h_{\boldsymbol{\theta}}}. \tag{34}$$

We consider operators with $\Psi = \{(0, 0, 0)\}$, which is an ellipticity requirement [2]. Then, it is possible to keep the suprenum (34) finite by an appropriate choice of the transfer operators I_h^{2h} and I_{2h}^h [9].

The whole spectrum is obtained similarly as in section 2. We, however, always observe a spectrum from a discrete mesh, whereas we have a continuously defined set of eigenfunctions (33) in order to get h-independent upper bounds for the convergence rates.

The definition of the smoothing factor (10) carries over from the rigorous analysis. The coarse grid correction operator is replaced by an ideal operator Q_h^{2h} which is orthogonally equivalent to a block matrix $\widetilde{Q}$ with blocks $\widehat{Q} = \mathrm{diag}(0, 1, 1, 1, 1, 1, 1, 1)$.

$$\mu_F := \rho(\widetilde{S}_2^{\nu_2}\widetilde{Q}\widetilde{S}_1^{\nu_1}) = \rho(\widetilde{Q}\widetilde{S}_1^{\nu_1}\widetilde{S}_2^{\nu_2}) = \sup_{\boldsymbol{\theta}\in\tilde{\Theta}^l} \rho(\widehat{Q}\widehat{S}_1(\boldsymbol{\theta})^{\nu_1}\widehat{S}_2(\boldsymbol{\theta})^{\nu_2}) \quad \text{with} \quad \widehat{S}_*(\boldsymbol{\theta}) \triangleq S_h^*|_{\boldsymbol{\varepsilon}^h_{\boldsymbol{\theta}}}.$$

TABLE 5

Multigrid convergence ρ_h^{mg}, predicted convergence ρ_F, and smoothing factor μ_F for the 2D anisotropic diffusion equation; $h = 1/128$.

Solution method	$\omega = 1.0$			$\omega = \omega_{opt}$		
ε	ρ_h^{mg}	ρ_F	μ_F	ρ_h^{mg}	ρ_F	μ_F
MG-FF, $\varepsilon = 0.1$	0.693	0.696	0.697	0.410	0.433	0.492
MG-FB, $\varepsilon = 0.1$	0.693	0.697	0.697	0.437	0.440	0.492
MG-FF, $\varepsilon = 0.01$	0.957	0.961	0.961	0.755	0.758	0.769
MG-FB, $\varepsilon = 0.01$	0.957	0.962	0.961	0.750	0.759	0.769

TABLE 6

Multigrid preconditioned GMRES(m) convergence ρ_h^{acc} and convergence estimates $\rho_{i=20}^U$, T_m^E, N_m^E for the 2D anisotropic diffusion equation; $h = 1/128$.

Acc.	$\varepsilon = 0.1$, $\omega = 1.0$				$\varepsilon = 0.1$, $\omega = 1.40$			
MG	ρ_h^{acc}	$\rho_{i=20}^U$	T_m^E	N_m^E	ρ_h^{acc}	$\rho_{i=20}^U$	T_m^E	N_m^E
FF, $m = 2$	0.475	0.515	0.529	0.911	0.410	0.420	0.430	0.524
FF, $m = 5$	0.450	0.469	0.519	0.907	0.410	0.420	0.430	0.441
FB, $m = 2$	0.370	0.402	0.408	0.718	0.185	0.185	0.203	0.445
FB, $m = 5$	0.305	0.331	0.331	0.487	0.158	0.160	0.165	0.202

Acc.	$\varepsilon = 0.01$, $\omega = 1.0$				$\varepsilon = 0.01$, $\omega = 1.75$			
MG	ρ_h^{acc}	$\rho_{i=20}^U$	T_m^E	N_m^E	ρ_h^{acc}	$\rho_{i=20}^U$	T_m^E	N_m^E
FF, $m = 2$	0.850	0.895	0.907	0.952	0.753	0.750	0.757	0.923
FF, $m = 5$	0.800	0.837	0.896	0.897	0.753	0.750	0.757	0.813
FB, $m = 2$	0.784	0.850	0.865	1.403	0.397	0.380	0.479	0.717
FB, $m = 5$	0.723	0.755	0.768	1.102	0.362	0.370	0.392	0.457

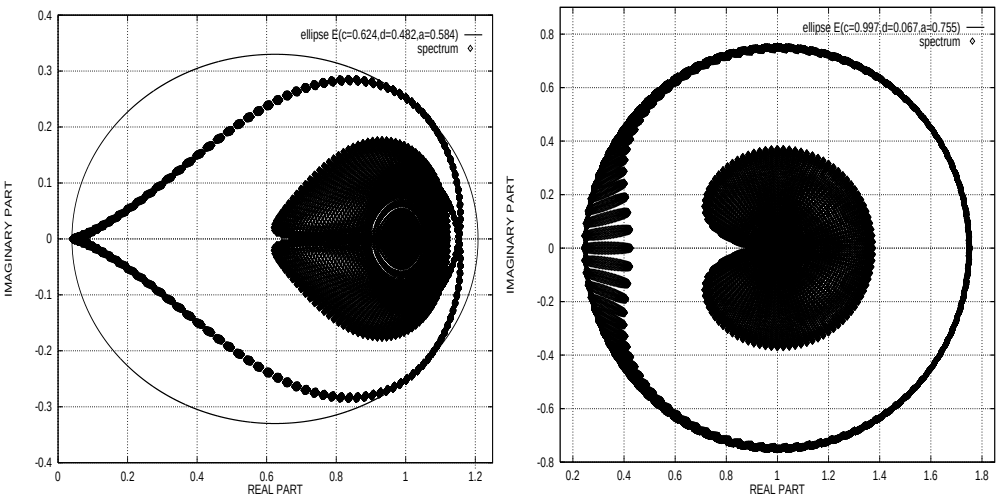
From the local mode analysis, the analytical values $\rho_{i=20}^U$, T_m^E, N_m^E, and ρ_{hpc} for the multigrid preconditioned GMRES(m) method are obtained by straightforward modifications of the definitions (32), (29), and (23) taken from the rigorous case.

4.1. Results for Poisson-type equations. The first example of the local-mode Fourier analysis of multigrid preconditioned GMRES(m) deals again with the anisotropic diffusion equation (2). We analyze the difference in convergence between performing two smoothing iterations ($\nu_1 = \nu_2 = 1$) of the forward lexicographical point Gauss–Seidel (GS-fLEX) method, and one iteration of GS-fLEX and one iteration of backward lexicographical point Gauss–Seidel (GS-bLEX). The other multigrid components are identical to the previously introduced multigrid method MG-RB. In the following, we call these multigrid methods MG-FF and MG-FB, which stand for multigrid forward forward and multigrid forward backward, respectively.

Local mode analysis gives sharp quantitative predictions for the multigrid convergence of MG-FF and MG-FB for the 2D anisotropic diffusion equation with $\varepsilon = 0.1$ and $\varepsilon = 0.01$, as can be seen in Table 5. In the anisotropic case, it is also beneficial to introduce an overrelaxation parameter for these lexicographical smoothers leading to ω-MG-FF and ω-MG-FB. The improvement is less impressive than for ω-MG-RB from [23], even if the optimal overrelaxation parameter ($\omega_{opt} = 1.40$ for $\varepsilon = 0.1, \omega_{opt} = 1.75$, or $\varepsilon = 0.01$) is selected (see Table 5).

In contrast to ω_{opt}-MG-FF (and ω_{opt}-MG-RB), it is possible to further accelerate ω_{opt}-MG-FB significantly by GMRES(m). This can be observed by comparing Table 5 and Table 6 and can be explained as follows. If A is a symmetric matrix, then MG-FB results in an *A-symmetric iteration matrix* due to the construction of the coarse grid correction (FW as the restriction and bilinear interpolation as the prolongation

operator) and the two relaxations which are adjoint to each other [8]. This means that the multigrid iteration matrix M is symmetric with respect to the A-induced inner product $(\,.\,,\,.\,)_A := (A\,.\,,\,.\,)$ (see, for example, [10]). The spectrum of the iteration matrix is real-valued, and uniformly distributed on the interval $[\lambda_{min}, \lambda_{max}]$. This feature is maintained if ω_{opt} is introduced, whereas for ω_{opt}-MG-FF an acceleration by GMRES(m) is not possible because the spectrum becomes orbital. This behavior of ω-MG-FF can be observed in Figure 4, where the spectra of the iteration matrices for $\omega = 1.0$ and $\omega = 1.75$ are shown for $\varepsilon = 0.01$. For other applications it is found in [10] that a symmetrization is not beneficial if Bi-CGstab [21] is used as the Krylov subspace acceleration method.



4 (a). $\omega = 1.0$ 4 (b). $\omega = 1.75$

FIG. 4. *Eigenvalue spectra of the MG-FF two-grid 2D anisotropic diffusion solver from local mode analysis;* $\varepsilon = 0.01, h = 1/128$. (*Note the different scaling of the axes in Figure* 4(a).)

Remark. For A-symmetric matrices it is possible to use the CG iteration as the acceleration method, in which the residual is minimized in the A-norm instead of the 2-norm (see (16)). The asymptotic convergence rate for the preconditioned CG iteration is given by the well-known formula [16]

$$\rho_{cg} := \Big(\min_{P\in\mathbb{P}_m} \max_{\lambda_j\in[\lambda_{min},\lambda_{max}]} |P(\lambda_j)| \Big)^{1/m} = \left(\max_{\lambda_j\in[\lambda_{min},\lambda_{max}]} \frac{T_m(1+2\frac{\lambda_{min}-\lambda_j}{\lambda_{max}-\lambda_{min}})}{T_m(1+2\frac{\lambda_{min}}{\lambda_{max}-\lambda_{min}})} \right)^{1/m}$$
$$= \left(2/T_m\left(\frac{\lambda_{max}+\lambda_{min}}{\lambda_{max}-\lambda_{min}}\right) \right)^{1/m} \le 2\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1} \qquad \text{with} \quad \kappa := \kappa_A(AC^{-1}) = \frac{\lambda_{max}}{\lambda_{min}}.$$

If we consider the definition T_m^E (29) for a real-valued spectrum $\sigma \subset [\lambda_{min}, \lambda_{max}]$, then the ellipse $E(c,d,a)$ deteriorates to a line. In particular we have $a = d = (\lambda_{max} - \lambda_{min})/2$ and $c = \lambda_{min} + a$, which leads to $T_m^E = (T_m(\frac{a}{a})/T_m(\frac{c}{a}))^{1/m} = \rho_{cg}$ (see (29), (30)). As the iteration matrix is A-symmetric, we find $\kappa_A(X) = 1$, where $\kappa_A(X)$ denotes the condition number of the transformation matrix X (26) with respect to the A-norm. In general, this does not imply that $\kappa_2(X) = 1$, but $\kappa_2(X)$ cancels out of (28) for large m. It can be expected that GMRES(m) has a similar asymptotic convergence as CG.

TABLE 7
Multigrid and multigrid preconditioned GMRES(m) convergence (different m) for the 3D Poisson equation and forward and backward lexicographical Gauss–Seidel smoothers.

Solution method	Grid 32^3	Grid 64^3	Grid 96^3	Fourier values
MG-FF	0.25	0.26	0.26	$\rho_F = 0.266$
Acc. MG-FF, $m = 2$	0.22	0.22	0.22	$\rho^U_{i=20} = 0.233$
Acc. MG-FF, $m = 5$	0.22	0.21	0.21	$\rho^U_{i=20} = 0.233$
MG-FB	0.29	0.29	0.29	$\rho_F = 0.294$
Acc. MG-FB, $m = 2$	0.098	0.10	0.10	$\rho^U_{i=20} = 0.121$
Acc. MG-FB, $m = 5$	0.086	0.085	0.086	$\rho^U_{i=20} = 0.099$

The analytical prediction $\rho^U_{i=20}$ shows a sharp quantitative character again, and the estimate T^E_m gives a reliable indication of the average accelerated convergence rate presented in Table 6. N^E_m is an interesting quantity for large m. For the 3D Poisson equation also, the multigrid convergence of MG-FF and MG-FB is similar. The convergence estimate ρ_F is 0.266 for MG-FF and 0.294 for MG-FB, whereas the spectra are completely different (as in 2D). It can be seen from Table 7 that the convergence rates with and without GMRES(m) are h-independent. Table 7 shows that very satisfactory convergence rates are obtained by the Krylov subspace acceleration of MG-FB. Moreover, due to the symmetry, we can predict the convergence rate of the accelerated MG-FB accurately from ρ_{cg} for different m.

Remark. With this knowledge, a symmetrization of the red-black multigrid Poisson solver is performed by doing the post-smoothing in black-red order. Indeed, the iteration matrix is A-symmetric, but the asymptotic convergence factor ρ_F increases to 0.440, which is also the value for MG-RB with only one smoothing step (see, for example, [7]). In this case the nonsymmetric multigrid iteration matrix leads to much better convergence factors with and without Krylov acceleration.

4.2. Results for a problem with mixed derivative. Next, we discuss an unsymmetric equation where reasonable smoothing properties can be achieved with point smoothers, but the coarse grid correction turns out to be problematic in combination with standard coarsening [19]. This equation is tackled in order to demonstrate that GMRES can also be applied to overcome coarse grid difficulties. We investigate the 2D differential equation with a mixed derivative $-\Delta u - \tau u_{xy} = b$ discretized by the $O(h^2)$ 9-point operator

$$A_h u_h(\boldsymbol{x}) = \frac{1}{h^2} \begin{bmatrix} \frac{\tau}{4} & -1 & -\frac{\tau}{4} \\ -1 & 4 & -1 \\ -\frac{\tau}{4} & -1 & \frac{\tau}{4} \end{bmatrix} u_h(\boldsymbol{x}) = b_h(\boldsymbol{x}) \quad \text{on} \quad \Omega_h.$$

It is no longer elliptic if $|\tau| = 2$ and the efficiency of all previously considered multigrid methods (MG-RB, MG-FF, MG-FB) deteriorates for $|\tau| \to 2$ (see, for example, Table 8 and [19]). This behavior can be explained if we perform a simplified two-grid analysis where only the very low Fourier frequencies along the characteristic direction of the differential operator (first differential approximation (FDA) [22]) are considered. The transfer operators act almost as identities for very low-frequency harmonics and no reduction of amplitudes can be achieved by the Gauss–Seidel relaxation. Thus, we obtain the following two-grid amplification factor (see (4)):

$$\rho^{FDA}(\boldsymbol{\theta}) := 1 - \frac{\lambda(\boldsymbol{\theta}, h)}{\lambda(2\boldsymbol{\theta}, 2h)} \quad \text{for } |\boldsymbol{\theta}| := \max\{|\theta_1|, |\theta_2|\} \longrightarrow 0. \tag{35}$$

TABLE 8
Multigrid and two-grid convergences ρ_h^{mg} and ρ_h^{tg}, predicted convergence ρ_F, and smoothing factor μ_F for the τ-problem, MG-FB, $h = 1/64$.

$\tau = -1.90$				$\tau = -1.99$			
ρ_h^{mg}	ρ_h^{tg}	ρ_F	μ_F	ρ_h^{mg}	ρ_h^{tg}	ρ_F	μ_F
0.747	0.654	0.662	0.499	0.839	0.730	0.745	0.518

TABLE 9
Multi- and two-grid preconditioned GMRES(m) convergence factors ρ_h^{acc} and $\rho_h^{acc}(tg)$ and estimates $\rho_{i=20}^U$, T_m^E for the τ-problem, $h = 1/64$.

Acc.	$\tau = -1.90$				$\tau = -1.99$			
MG-FB	ρ_h^{acc}	$\rho_h^{acc}(tg)$	$\rho_{i=20}^U$	T_m^E	ρ_h^{acc}	$\rho_h^{acc}(tg)$	$\rho_{i=20}^U$	T_m^E
$m = 2$	0.420	0.333	0.372	0.373	0.533	0.404	0.415	0.463
$m = 5$	0.370	0.287	0.302	0.304	0.472	0.348	0.354	0.378

The Fourier symbol $\lambda(\boldsymbol{\theta}, h)$ of the differential operator reads as follows:

$$\begin{aligned}\lambda(\boldsymbol{\theta}, h) &= \frac{1}{h^2}\Big(4 - 2\cos(\theta_1) - 2\cos(\theta_2) + \frac{\tau}{2}\cos(\theta_1 - \theta_2) - \frac{\tau}{2}\cos(\theta_1 + \theta_2)\Big) \\ &= \frac{1}{h^2}\Big(\theta_1^2 + \theta_2^2 + \frac{\tau}{4}(\,(\theta_1 + \theta_2)^2 - (\theta_1 - \theta_2)^2\,) - \frac{\theta_1^4 + \theta_2^4}{12} \\ &\qquad + \frac{\tau}{48}(\,(\theta_1 - \theta_2)^4 - (\theta_1 + \theta_2)^4\,) + \mathrm{O}(|\boldsymbol{\theta}|^6)\Big),\end{aligned}$$

where Taylor's expansion is used. If $\tau \longrightarrow \pm 2$ the characteristic frequencies are given by $\theta_2 = \mp\theta_1$. This gives

$$\begin{aligned}&\lambda(\boldsymbol{\theta}, h) = \frac{1}{h^2}\Big((2 \mp \tau)\theta_1^2 - \Big(\frac{1}{6} \mp \frac{\tau}{3}\Big)\theta_1^4 + \mathrm{O}(\theta_1^6)\Big) \\ \Longrightarrow\quad &\rho^{FDA}(\boldsymbol{\theta}) = 1 - \frac{4}{16} = 0.75 \quad \text{for } |\boldsymbol{\theta}| \longrightarrow 0 \text{ and } \tau \longrightarrow \pm 2.\end{aligned}$$

For $\tau = -1.99$ and MG-FB, this limiting value of 0.75 is almost obtained by the predicted two-grid factor ρ_F and also by the numerically obtained ρ_h^{tg}; see Table 8. The multigrid convergence ρ_h^{mg} increases further since the coarse grid problem occurs on all coarser grids (actually six grids are used).

One way to handle the coarse grid problem is to change the smoother to the more expensive ILU-type relaxation [19], which also takes care of the problematic low-frequency error components. A second way is to replace the grid coarsening by a nonstandard coarsening technique. We keep the point Gauss–Seidel smoother with standard grid coarsening and solve this problem by the Krylov subspace acceleration of MG-FB. In Table 9, a significant improvement of the multigrid convergence due to the acceleration is observed even with a small Krylov subspace. The two-grid convergence $\rho_h^{acc}(tg)$ is very well predicted by $\rho_{i=20}^U$ and T_m^E.

Comparing the accelerated two-grid and multigrid convergences from Table 9, it is expected that it is possible to further improve ρ_h^{acc} by incorporating the Krylov acceleration into the multigrid cycle and to apply it also on the coarse grids as in [14].

4.3. Further results and extensions. Here, we present a relation between Krylov subspace acceleration and a coarse grid correction acceleration as presented for the convection-diffusion equation in [4] and a relation between Krylov subspace acceleration and optimal (multistage) overrelaxation in a smoothing method.

A standard upwind discretization of a convection-dominated ($0 < \epsilon \ll 1$) convection-diffusion equation

$$-\epsilon\Delta u(\boldsymbol{x}) + (\, a_1(\boldsymbol{x})u(\boldsymbol{x})\,)_x + (\, a_2(\boldsymbol{x})u(\boldsymbol{x})\,)_y = b(\boldsymbol{x}), \tag{36}$$

with a rotating convection term ($a_1(\boldsymbol{x}) = -\sin(\pi x)\cos(\pi y)$, $a_2(\boldsymbol{x}) = \sin(\pi y)\cos(\pi x)$), is another well-known equation with a problematic coarse grid correction (see [4], [8]). A similar FDA analysis, as in the previous section, shows that characteristic low frequency error components, that are constant along the characteristics of the advection operator, are not reduced efficiently on coarse grids [4]. The different scaling of convection $(\, a_1(\boldsymbol{x})/h, a_2(\boldsymbol{x})/h\,)$ and diffusion ϵ/h^2 is not approximated properly on the $2h$-grid which leads to a limiting two-grid convergence factor of 0.5 [4]. It is shown in [14] that the problems with rotating convection direction can be solved efficiently by a Krylov subspace acceleration of multigrid (also on the coarser grids of the multigrid preconditioner). In [4] an improved multigrid method is designed for this problem where the residuals in the kth two-grid cycle are overweighted by optimized acceleration parameters η_k. This means that the FDA error amplification factor after the mth cycle is given by the following polynomial (see (35) and [4]):

$$\rho_m^{FDA}(\boldsymbol{\theta}) = \prod_{k=1}^{m} (\, 1 - \eta_k \zeta(\boldsymbol{\theta})\,) \quad \text{with} \quad \zeta(\boldsymbol{\theta}) = \frac{\lambda(\boldsymbol{\theta}, h)}{\lambda(2\boldsymbol{\theta}, 2h)}.$$

The two methods are related. The FDA analysis can also be performed for the argument of the GMRES(m)-polynomials (21). It yields that the block matrix $UAC^{-1}U$ is dominated by $\zeta(\boldsymbol{\theta})$ for very low characteristic frequencies $\boldsymbol{\theta}$. We can construct an initial solution u_0 (necessary to calculate the GMRES(m)-polynomials explicitly; see subsection 3.2), which consists only of characteristic error components and freezes the discretization operator at a fixed (but arbitrary) point $\boldsymbol{x}$. All resulting GMRES(m)-polynomials P_m^i are almost identical for $i \geq 20$. Therefore, always the same coefficients α_k^i ($k = 1, \ldots, m$) are found. Thus, it is possible to identify ρ_m^{FDA} with P_m^i for $i \geq 20$ and we can express the coefficients α_k^i ($i \geq 20$) by the η_k, for example,

$$\alpha_1^i = -\eta_1 \quad \text{for} \quad m = 1\,, \quad \alpha_1^i = -(\eta_1 + \eta_2)\,,\ \alpha_2^i = \eta_1\eta_2 \quad \text{for} \quad m = 2. \tag{37}$$

The above considerations are confirmed by test calculations where such relations as from (37) are established for $i \geq 20$ and also for larger m. For problem (36), for example, we find

$$\alpha_1^i = -1.33 \quad \text{for} \quad m = 1\,, \quad \alpha_1^i = -2.82\,,\ \alpha_2^i = 1.88 \quad \text{for} \quad m = 2,$$

which matches exactly with the reference values η_k; see [4]. We see from this heuristic consideration that an optimal tuning of the coarse grid correction is implicitly done by the Krylov subspace acceleration.

Similarly it is possible to find optimal overrelaxation parameters for smoothing methods, like Jacobi or GS-RB, if the coarse grid correction acts as the optimal operator Q_h^{2h} from sections 2 and 4. For example, consider the multistage Jacobi relaxation for the 2D Poisson equation for which the smoothing operator is given by the polynomial

$$S_h = \prod_{k=1}^{m} (\, I_h - \omega_k \Delta_h/4\,). \tag{38}$$

The corresponding smoothing factor for the multistage relaxation S_h from (38) can be minimized analytically, yielding the optimal ω_k, which are knots of Chebychev polynomials [19]. If we select an initial approximation which consists only of high frequency error components, it is possible to identify S_h with the GMRES(m)-polynomials P_m^i (for $i \geq 20$) as the coarse grid correction leaves the high frequency components almost unchanged. A similar comparison of coefficients ω_k and α_k can be performed as described above for η_k and α_k (37). The related α_k^i, calculated by the analysis from subsection 3.2, match with the optimal ω_k for $i \geq 20$ very well.

For more complicated equations (or smoothing procedures) it is, in general, not possible to determine optimal overrelaxation parameters analytically, but the analysis is easily applicable and the α_k^i can be determined. As an example we consider the discrete biharmonic equation $\Delta_h \Delta_h u_h = b_h$ which is sometimes transformed into a system of two Poisson-type equations in order to achieve better smoothing properties [1]. Here we keep the scalar forth order problem and apply a multistage version of a GS-RB smoothing method

$$S_h^{msRB} = \prod_{k=1}^{m} (I_h - \omega_k (I_h - S_h^{BLACK} \cdot S_h^{RED})),$$

where the multistage parameters ω_k are obtained by an evaluation of the related α_k^i ($i \geq 20$). It can be seen in Table 10 that it is possible to obtain satisfactory smoothing rates also for the scalar biharmonic equation.

TABLE 10

Multistage red-black Gauss–Seidel relaxation parameters and smoothing rate for the biharmonic equation.

	α_1	α_2	α_3	μ_F	$(\mu_F)^{1/m}$
$m = 1$	-1.39			0.501	0.501
$m = 2$	-3.17	2.20		0.152	0.390
$m = 3$	-4.81	7.13	-3.30	0.050	0.369

Finally, it can be concluded that the Krylov subspace acceleration implicitly improves the coarse grid correction or the relaxation procedure, if one of these multigrid components clearly hampers the overall multigrid convergence. Furthermore, the analysis from subsection 3.2 is an easy tool to obtain good overrelaxation parameters for different (and also difficult) problems.

5. Conclusions. In this paper we have presented a way to obtain sharp quantitative convergence estimates for GMRES(m) preconditioned by multigrid on the basis of Fourier analysis. For all the cases considered the estimates are accurate compared to measured numerical convergence of the multigrid preconditioned GMRES(m) method.

It has been shown that it is not easily possible to further accelerate multigrid methods which are optimally tuned with, for example, overrelaxation parameters. In other situations, however, very satisfactory convergence improvement is achieved with the Krylov subspace acceleration. The possibilities for the subspace acceleration of multigrid are not only available in the case of isolated eigenvalues from the multigrid preconditioner, but they depend also on the shape of the spectrum. A spectrum resulting from the preconditioner with a circular shape is not appropriate for an acceleration with the method presented. For a fair comparison one should take the additional work for the Krylov subspace acceleration into account. This, however,

is not the main issue in this paper. The acceleration is, in general, cheap compared to the multigrid method. The proposed acceleration is, of course, also applicable to situations in which it is not easy to tune a multigrid method.

REFERENCES

[1] A. Brandt, *Multi-level adaptive solutions to boundary-value problems*, Math. Comp., 31 (1977), pp. 333–390.

[2] A. Brandt, *Rigorous quantitive analysis of multigrid,* I*: Constant coefficients two-level cycle with L_2-norm*, SIAM J. Numer. Anal., 31 (1994), pp. 1695–1730.

[3] A. Brandt and V. Mikulinsky, *On recombining iterants in multigrid algorithms and problems with small islands*, SIAM J. Sci. Comput., 16 (1995), pp. 20–28.

[4] A. Brandt and I. Yavneh, *Accelerated multigrid convergence and high-Reynolds recirculating flows*, SIAM J. Sci. Comput., 14 (1993), pp. 607–626.

[5] S. C. Eisenstat, H. C. Elman, and M. H. Schultz, *Variational iterative methods for nonsymmetric systems of linear equations*, SIAM J. Numer. Anal., 20 (1983), pp. 345–357.

[6] H. C. Elman, *Iterative Methods for Large Sparse Nonsymmetric Systems of Linear Equations*, Ph.D. thesis, Computer Science Department, Yale University, New Haven, CT, 1982.

[7] W. Hackbusch, *Iterative Solution of Large Sparse Systems of Equations,* translated and revised from the 1991 German original, Appl. Math. Sci. 95, Springer-Verlag, New York, 1994.

[8] W. Hackbusch, *Multi-Grid Methods and Applications*, Springer, Berlin, Germany, 1985.

[9] P. W. Hemker, *On the order of prolongations and restrictions in multigrid procedures*, J. Comput. Appl. Math., 32 (1990), pp. 423–429.

[10] M. Holst and S. Vandewalle, *Schwarz methods: To symmetrize or not to symmetrize*, SIAM J. Numer. Anal., 34 (1997), pp. 699–722.

[11] R. Kettler, *Analysis and comparison of relaxation schemes in robust multigrid and preconditioned conjugate gradient methods*, in Multigrid Methods, Lecture Notes in Math. 960, W. Hackbusch and U. Trottenberg, eds., Springer, Berlin, Germany, 1982, pp. 502–534.

[12] N. M. Nachtigal, S. C. Reddy, and L. N. Trefethen, *How fast are nonsymmetric matrix iterations?*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 778–795.

[13] C. W. Oosterlee and T. Washio, *An evaluation of parallel multigrid as a solver and a preconditioner for singularly perturbed problems*, SIAM J. Sci. Comput., 19 (1998), pp. 87–110.

[14] C. W. Oosterlee and T. Washio, *Krylov subspace acceleration of nonlinear multigrid with application to recirculating flows*, SIAM J. Sci. Comput., 21 (2000), pp. 1670–1690.

[15] Y. Saad and M. H. Schultz, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 856–869.

[16] Y. Saad, *Iterative Methods for Sparse Linear Systems*, PWS Publishing, Boston, 1996.

[17] Y. Saad, *Further Analysis of Minimal Residual Iterations*, Tech. report, available online at ftp://ftp.cs.umn.edu/dept/users/saad/reports/FILES/umsi-97-14.ps.gz (1997).

[18] R. Stevenson, *On the Validity of Local Mode Analysis of Multi-Grid Methods*, Ph.D. thesis, Rijks University Utrecht, The Netherlands, 1990.

[19] K. Stüben and U. Trottenberg, *Multigrid methods: Fundamental algorithms, model problem analysis and applications*, in Multigrid Methods, Lecture Notes in Math. 960, W. Hackbusch and U. Trottenberg, eds., Springer, Berlin, Germany, 1982, pp. 1–176.

[20] C. A. Thole and U. Trottenberg, *Basic smoothing procedures for the multigrid treatment of elliptic* 3*-D operators*, Appl. Math. Comput., 19 (1986), pp. 333–345.

[21] H. A. van der Vorst, *BI-CGSTAB: A fast and smoothly converging variant of BI-CG for the solution of nonsymmetric linear systems*, SIAM J. Sci. Comput., 13 (1992), pp. 631–644.

[22] N. N. Yanenko and Y. I. Shokin, *On the correctness of first differential approximation of difference schemes*, Dokl. Akad. Nauk. SSSR, 182 (1968), pp. 776–778.

[23] I. Yavneh, *On red-black SOR smoothing in multigrid*, SIAM J. Sci. Comput., 17 (1996), pp. 180–192.

SIAM J. SCI. COMPUT.
Vol. 22, No. 2, pp. 604–616
© 2000 Society for Industrial and Applied Mathematics

ADAPTIVE BOUNDARY ELEMENT METHODS BASED ON COMPUTATIONAL SCHEMES FOR SOBOLEV NORMS*

O. STEINBACH†

Abstract. Multilevel methods are applied to compute local and global Sobolev norms arising in adaptive boundary element methods. Combining this approach with an approximate computation of an error function we construct efficient and reliable a posteriori error estimators for arbitrary boundary element solutions related to either a Dirichlet or Neumann boundary value problem.

Key words. adaptivity, boundary element methods, error estimation, multilevel methods

AMS subject classifications. 65N38, 65N50

PII. S1064827599352999

1. Introduction.

Multilevel methods (see, for example, [2, 3, 9, 13, 14, 20, 22, 23]) were originally designed for an efficient solution of numerical schemes related to elliptic boundary value problems, in particular, finite and boundary element methods. In fact, multilevel methods are often used to construct almost optimal preconditioners to be used in iterative methods, i.e., we refer to BPX-type preconditioners first considered in [3, 22]. The numerical analysis of such an approach is essentially based on multilevel representations of equivalent norms in certain Sobolev spaces; see, for example, [14]. In a recent paper by Oswald [15] it was shown that the Sobolev norm in $H^{-1/2}(\Gamma)$ allows an equivalent multilevel representation using piecewise constant trial functions. In this paper we will use this equivalent multilevel approximation to define local and global a posteriori error indicators needed in adaptive boundary element methods. A related approach is to derive a posteriori error estimators using hierarchical basis techniques; see, e.g., [1, 16] for finite elements, [11] for boundary elements, or [12] for a coupling of finite and boundary elements.

For the design of adaptive boundary element methods, it is necessary to estimate and localize the error. This is more complicated, as in finite element methods, since the boundary integral operators involved are, in general, nonlocal. Residual-based error estimators for Galerkin boundary element methods were considered, for example, in [4, 7, 17, 21]. The challenge there is the localization of the Sobolev norm, in particular for the Dirichlet boundary value problem where the related boundary integral operator is of order minus one, and to prove reliability of the resulting local error indicators [8]. Since residual-based error estimators require the evaluation of the residual inside certain quadrature formulae (see, for example, [4]), for some applications these methods may be time consuming and therefore become inefficient; see [18].

In [18, 19] we considered an alternative approach to approximate the error function of an arbitrary given boundary element solution for both the Dirichlet and Neumann boundary value problem. This method is based on a truncated Neumann series

* Received by the editors March 10, 1999; accepted for publication (in revised form) February 14, 2000; published electronically August 3, 2000. This work was done while the author was a Postdoctoral Research Associate at the Institute for Scientific Computation (ISC), Texas A & M University, College Station, TX.

http://www.siam.org/journals/sisc/22-2/35299.html

† Mathematisches Institut A, Universität Stuttgart, Pfaffenwaldring 57, 70569 Stuttgart, Germany (steinbach@mathematik.uni-stuttgart.de).

to solve a second kind boundary integral equation. In [18, 19] we analyzed the convergence of these Neumann series in appropriate energy norms to get reliability and efficiency for the resulting global error estimators. To get local error indicators, we used the local contributions of the energy norm. However, in this case we could prove the efficiency of the local error indicators only. Moreover, the use of energy norms induced by boundary integral operators, in particular by the single layer potential, seems not to be satisfactory with respect to the computational costs.

In this paper we describe a new approach to compute and to localize Sobolev norms by means of multilevel methods. These methods are almost optimal with respect to the amount of work. Therefore, in comparison with the approach described in [18, 19], we get a more efficient algorithm as it will be seen in the numerical example. Moreover, when using a multilevel norm instead of an energy norm, we can prove also reliability of the resulting global and local error estimator.

While in this paper we consider multilevel methods to compute Sobolev norms of an approximated error function, one can use such an approach even to compute and localize Sobolev norms of the residual. This modification is straightforward and will not be considered here.

The rest of the paper is organized as follows. In section 2 we recall the definition and basic properties of multilevel operators and derive a local representation of an equivalent Sobolev norm in $H^{-1/2}(\Gamma)$. The approximation of the error function in case of a Dirichlet boundary value problem is described in section 3, where we apply the results of section 2 to construct local and global error indicators. Section 4 is devoted to the same topics in the case of a Neumann boundary value problem. In section 5 we give a general adaptive boundary element algorithm for the solution of Dirichlet or Neumann boundary values problems. Some numerical examples for the two-dimensional Laplacian are given in section 6, i.e., we compare the proposed new approach with an uniform approach as well as with an adaptive algorithm based on the energy norm as considered in [18, 19].

2. Computational schemes for Sobolev norms. Let $\Omega \subset \mathbb{R}^n$ $(n = 2, 3)$ be a bounded domain with Lipschitz boundary $\Gamma = \partial\Omega$. For a family of regular boundary element meshes Γ_{h_j} $(j = 1, \ldots, J)$ of mesh size h_j and consisting of N_j boundary elements $\Gamma_{j,k}$ $(k = 1, \ldots, N_j)$ we define a nested sequence of trial spaces Z_j of piecewise constant basis functions $\psi_{j,k}$ $(k = 1, \ldots, N_j; j = 1, \ldots, J)$, i.e.,

$$Z_1 \subset Z_2 \subset \cdots \subset Z_J \subset H^{-1/2}(\Gamma).$$

Let Q_j denote the L^2 Galerkin projection onto Z_j defined by

$$\langle Q_j w, \tau \rangle_{L^2(\Gamma)} = \langle w, \tau \rangle_{L^2(\Gamma)} \quad \text{for all } \tau \in Z_j. \tag{2.1}$$

Set $Q_0 = 0$. Then there hold the following relations for $k = 1, \ldots, J$:

$$Q_k Q_\ell = Q_\ell Q_k = Q_\ell \quad \text{for } \ell \le k, \tag{2.2}$$

$$(Q_k - Q_{k-1})Q_\ell = 0 \quad \text{for } \ell < k, \tag{2.3}$$

$$(Q_k - Q_{k-1})^2 = Q_k - Q_{k-1}, \tag{2.4}$$

$$(Q_k - Q_{k-1})(Q_\ell - Q_{\ell-1}) = 0 \quad \text{for } k \neq \ell. \tag{2.5}$$

In fact, (2.3)–(2.5) are simple consequences of (2.2) [22].

For $s \in \mathbb{R}$ and $w \in Z_J$ we define the multilevel operator

$$A^{(s)} w = \sum_{j=1}^{J} h_j^{-2s} \cdot (Q_j - Q_{j-1}) w \,. \tag{2.6}$$

From [15, Theorem 2] it is known that there hold the spectral equivalence inequalities

$$c_1||w||^2_{H^{-1/2}(\Gamma)} \leq \langle A^{(-\frac{1}{2})}w, w\rangle_{L^2(\Gamma)} \leq c_2 J^2||w||^2_{H^{-1/2}(\Gamma)} \quad \text{for all } w \in Z_J \tag{2.7}$$

with positive constants c_1 and c_2 independent of $w \in Z_J$ and $J \geq 1$.

While we can use the operator $A^{(-1/2)}$ to compute an equivalent Sobolov norm in $H^{-1/2}(\Gamma)$ (this is due to (2.7)) we need to compute local norms to be used in the mesh refinement of adaptive boundary element methods. This will be based on the following result which is a consequence of (2.4) and (2.5); see also [22].

PROPOSITION 2.1. *Let $A^{(s)}$ be defined as in* (2.6) *for some $s \in \mathbb{R}$. Then there holds*

$$A^{(s)} = A^{(s/2)} \cdot A^{(s/2)}.$$

Proof. A simple computation shows that

$$\begin{aligned} A^{(s/2)} \cdot A^{(s/2)} &= \sum_{k=1}^{J} h_k^{-s}(Q_k - Q_{k-1}) \cdot \sum_{\ell=1}^{J} h_\ell^{-s}(Q_\ell - Q_{\ell-1}) \\ &= \sum_{k=1}^{J}\sum_{\ell=1}^{J} h_k^{-s} h_\ell^{-s}(Q_k - Q_{k-1})(Q_\ell - Q_{\ell-1}) \\ &= \sum_{k=1}^{J} h_k^{-2s}(Q_k - Q_{k-1}) = A^{(s)} \end{aligned}$$

due to (2.4) and (2.5). ∎

As a conclusion of (2.7) and Proposition 2.1 we get that

$$||A^{(-1/4)}w||^2_{L^2(\Gamma)} = \langle A^{(-1/2)}w, w\rangle_{L^2(\Gamma)} \quad \text{for all } w \in Z_J \tag{2.8}$$

is an equivalent Sobolev norm in $H^{-1/2}(\Gamma)$. Moreover, we can localize (2.8) by

$$||A^{(-1/4)}w||^2_{L^2(\Gamma)} = \sum_{k=1}^{N_J} ||A^{(-1/4)}w||^2_{L^2(\Gamma_{J,k})} \quad \text{for all } w \in Z_J. \tag{2.9}$$

The computation of $A^{(-1/4)}w$ for $w \in Z_J$ via (2.6) requires the solution of the variational problems (2.1) for all $j = 1, \ldots, J$, i.e., the inversion of the Gram matrices G_j defined by $G_j[k,\ell] = \langle \psi_{j,k}, \psi_{j,\ell}\rangle_{L^2(\Gamma)}$ for $k, \ell = 1, \ldots, N_j; j = 1, \ldots, J$. Since we are using piecewise constant trial functions $\psi_{j,k}$ the Gram matrices G_j are just diagonal matrices, and therefore the computation of G_j^{-1} and thus of $Q_j w$ is trivial.

3. An error estimator for the Dirichlet problem. Let $\Omega \subset \mathbb{R}^n$ $(n = 2, 3)$ with Lipschitz boundary $\Gamma = \partial\Omega$. For the numerical solution of the Dirichlet boundary value problem

$$Lu(x) = 0 \quad \text{for } x \in \Omega, \qquad u(x) = g(x) \quad \text{for } x \in \Gamma \tag{3.1}$$

with an elliptic and self-adjoint second-order partial differential operator L we consider the boundary integral equation

$$(Vt)(x) = \left(\frac{1}{2}I + K\right) g(x) \quad \text{for } x \in \Gamma \tag{3.2}$$

with the standard notations for the single layer potential V and the double layer potential K,

$$(Vt)(x) = \int_\Gamma U^*(x,y)t(y)ds_y, \qquad (Ku)(x) = \int_\Gamma u(y)T^*(x,y)ds_y.$$

Here, $U^*(x,y)$ is the fundamental solution of the partial differential operator L, $T^*(x,y) = T_y U^*(x,y)$, and $t(y) = T_y u(y)$, where T_y is the conormal derivative operator corresponding to L.

Let $Z_h := Z_I$ be a trial space of piecewise constant basis functions as introduced in the previous section for some $1 \le I < J$. The Galerkin variational formulation of (3.2) reads to find $t_h \in Z_h$ such that

$$\langle Vt_h, \tau_h \rangle_{L^2(\Gamma)} = \left\langle \left(\frac{1}{2}I + K\right) g, \tau_h \right\rangle_{L^2(\Gamma)} \quad \text{for all } \tau_h \in Z_h. \tag{3.3}$$

It is well known (see, for example, [10]) that the variational problem (3.3) is uniquely solvable and that there holds the a priori error estimate

$$||t - t_h||_{H^{-1/2}(\Gamma)} \le c \cdot \inf_{\tau_h \in Z_h} ||t - \tau_h||_{H^{-1/2}(\Gamma)} \le c \cdot h^{s+\frac{1}{2}} \cdot ||t||_{H^s(\Gamma)} \tag{3.4}$$

if $t \in H^s(\Gamma)$ and $s \le 1$.

The design of adaptive boundary element methods is in general based on global and local a posteriori estimators for the error

$$e_h(x) := (t - t_h)(x) \quad \text{for } x \in \Gamma. \tag{3.5}$$

The use of residual based error estimators requires the computation of

$$r_h(x) := (Ve_h)(x) = \left(\frac{1}{2}I + K\right) g(x) - (Vt_h)(x) \quad \text{for } x \in \Gamma. \tag{3.6}$$

Due to the equivalence inequalities

$$c_1 \cdot ||r_h||^2_{H^{1/2}(\Gamma)} \le ||e_h||^2_{H^{-1/2}} \le c_2 \cdot ||r_h||^2_{H^{1/2}(\Gamma)}$$

one can use $||r_h||^2_{H^{1/2}(\Gamma)}$ to derive both local and global error indicators, i.e., one has to localize an equivalent Sobolev norm in $H^{1/2}(\Gamma)$; see [4, 8]. However, numerical examples in [4, 18] indicate that a residual-based approach to constructing error estimators may lead to unstable and, with respect to computing times, inefficient algorithms. In [18] we proposed an alternative approach to estimate the error (3.5) by solving a second kind boundary integral equation via a truncated Neumann series. The resulting approximation of the error (3.5) can be done with high accuracy, but the computation of global and local error indicators now has to be done in a equivalent Sobolev norm in $H^{-1/2}(\Gamma)$. In [18] we used the energy norm induced by the single layer potential V to compute the global error estimator as well as some local indicators. Although, for the example considered, the approach in [18] was more efficient compared with the method described in [4] and an uniform refinement, the amount of work to compute the energy norm was not satisfactory. Therefore, we consider in this paper an alternative approach to compute both the global and local error indicators appearing in [18] by an equivalent Sobolev norm based on multilevel trial spaces as described in

section 2. Moreover, we also get reliability of local error indicators when using this multilevel approach.

Let $t_h \in Z_h$ be an arbitrary boundary element approximation for $t \in H^{-1/2}(\Gamma)$, for example, we may take the Galerkin solution of (3.3) but any other discretization scheme such as collocation may be used also. Then we can define an approximate solution of the boundary value problem (3.1) by

$$u_h(\widetilde{x}) = \int_\Gamma U^*(\widetilde{x},y)t_h(y)ds_y - \int_\Gamma g(x)T^*(\widetilde{x},y)ds_y \quad \text{for } \widetilde{x} \in \Omega. \tag{3.7}$$

Applying T_x for $x \in \Gamma$ and taking the limit $\Omega \ni \widetilde{x} \to x \in \Gamma$, (3.7) defines a computable function

$$\widetilde{t}(x) = \left(\frac{1}{2}I + K'\right) t_h(x) + (Dg)(x) \quad \text{for } x \in \Gamma \tag{3.8}$$

almost everywhere where K' is the adjoint double layer potential operator and D is the hypersingular integral operator,

$$(K't)(x) = \int_\Gamma T_x U^*(x,y)t(y)ds_y, \quad (Du)(x) = -T_x \int_\Gamma u(x)T_y U^*(x,y)ds_y.$$

It was shown in [18, Lemma 1] that the error (3.5) is then a solution of the second kind boundary integral equation

$$\left(\frac{1}{2}I - K'\right)(t - t_h)(x) = (\widetilde{t} - t_h)(x) \quad \text{for } x \in \Gamma. \tag{3.9}$$

Let $||\cdot||_V$ denote the energy norm in $H^{-1/2}(\Gamma)$ induced by the single layer potential. Then (see [18, Theorem 1])

$$\left\| \left(\frac{1}{2}I + K'\right) w \right\|_V \le c_K \cdot ||w||_V \quad \text{for all } w \in H^{-1/2}(\Gamma) \quad \text{with } c_K < 1. \tag{3.10}$$

Therefore we can compute the error (3.5) by the Neumann series

$$e_h(x) = \sum_{\ell=0}^{\infty} (\frac{1}{2}I + K')^\ell (\widetilde{t} - t_h)(x) \quad \text{for } x \in \Gamma.$$

Using a truncated Neumann series, i.e., for a fixed $q \ge 0$, we may compute the approximate error function

$$e_h^{(q)}(x) = \sum_{\ell=0}^{q} \left(\frac{1}{2}I + K'\right)^\ell e_h^{(0)}(x), \quad e_h^{(0)}(x) = (\widetilde{t} - t_h)(x), \tag{3.11}$$

and derive global and local error indicators from $||e_h^{(q)}||^2_{H^{-1/2}(\Gamma)}$ or an equivalent norm. From (3.11) using (3.10) we conclude the equivalence inequalities

$$\frac{1}{1 + c_K^{q+1}} \, ||e_h^{(q)}||_V \le ||e_h||_V \le \frac{1}{1 - c_K^{q+1}} \, ||e_h^{(q)}||_V. \tag{3.12}$$

For a numerical realization of the successive approximation (3.11) we introduce a trial space $Z_J := Z_{\tilde{h}}$ of dimension N_J with $Z_h \subset Z_{\tilde{h}}$ and compute

$$e_{\tilde{h}}^{(q)}(x) = \sum_{\ell=0}^{q} \left[Q_{\tilde{h}} \left(\frac{1}{2} I + K' \right) \right]^{\ell} e_{\tilde{h}}^{(0)}(x), \quad e_{\tilde{h}}^{0}(x) = Q_{\tilde{h}}(\widetilde{t} - t_h)(x), \tag{3.13}$$

where $Q_{\tilde{h}}$ is the L^2 Galerkin projection onto $Z_{\tilde{h}}$ as defined in (2.1).

Hence, using the energy norm induced by the single layer potential, a global a posteriori error estimator is given by

$$\eta^{(q)} := ||e_{\tilde{h}}^{(q)}||_V. \tag{3.14}$$

Combining the equivalence inequalities (3.12) with the approximation error of $Q_{\widetilde{h}}$ in $H^{-1/2}(\Gamma)$ we get the following result.

THEOREM 3.1 (see [18, Theorem 3.3]). *Let $\eta^{(q)}$ as defined in* (3.14). *Then,*

$$\frac{1}{1 + c_K^{q+1}} \left\{ \eta^{(q)} - c \cdot q \cdot \widetilde{h}^{1/2} ||e_h^{(0)}||_{L^2(\Gamma)} \right\} \leq ||e_h||_V \leq \frac{1}{1 - c_K^{q+1}} \left\{ \eta^{(q)} + c \cdot q \cdot \widetilde{h}^{1/2} ||e_h^{(0)}||_{L^2(\Gamma)} \right\}.$$

We note that, due to our construction, the trial spaces Z_j form a nested sequence

$$Z_1 \subset \cdots \subset Z_I = Z_h \subset \cdots \subset Z_J = Z_{\tilde{h}},$$

therefore we can apply all results from section 2, in particular, we may define the global error estimator

$$\widetilde{\eta}^{(q)} = \sqrt{\langle A^{-1/2} e_{\tilde{h}}^{(q)}, e_{\tilde{h}}^{(q)} \rangle_{L^2(\Gamma)}} \tag{3.15}$$

which is equivalent to $\eta^{(q)}$ as defined in (3.14) and therefore to $||e_h||_V$. Moreover, due to (2.9) we can define local error indicators

$$\widetilde{\eta}_k^{(q)} := ||A^{-1/4} e_{\tilde{h}}^{(q)}||_{L^2(\Gamma_{I,k})} \tag{3.16}$$

satisfying

$$[\widetilde{\eta}^{(q)}]^2 = \sum_{k=1}^{N_I} [\widetilde{\eta}_k^{(q)}]^2.$$

Hence, the local error indicators defined by (3.16) satisfy the equivalence inequalities

$$c_1 \cdot \sqrt{\sum_{k=1}^{N_I} [\widetilde{\eta}_k^{(q)}]^2} \leq ||e_h||_V \leq c_2 \cdot \sqrt{\sum_{k=1}^{N_I} [\widetilde{\eta}_k^{(q)}]^2}. \tag{3.17}$$

The local error indicators (3.16) are local contributions of the global Sobolev norm (2.8) which is an equivalent norm in $H^{-1/2}(\Gamma)$. Moreover, the global error estimator (3.15) can be derived from all local indicators. Hence we get reliability and efficiency for both global and local error indicators. Note that we are localizing a Sobolev norm in $H^{-1/2}(\Gamma)$ to get a posteriori error estimators for the Dirichlet problem. When using a residual-based approach one has to localize a Sobolev norm in $H^{1/2}(\Gamma)$. This can be done, for example, by a window technique as described in [17], by a direct localization of the Sobolev–Slobodeckii norm in $H^{1/2}(\Gamma)$ (see [8]), or by local indicators of Babuška–Rheinboldt type [4, 7].

4. An error estimator for the Neumann problem. Now we consider instead of (3.1) the Neumann boundary value problem

$$Lu(x) = 0 \quad \text{for } x \in \Omega, \qquad (T_x u)(x) = f(x) \quad \text{for } x \in \Gamma. \tag{4.1}$$

Note that f has to satisfy the compatibility condition

$$\int_\Gamma f(x)v(x)ds_x \;=\; 0 \quad \text{for all } v \in \mathcal{K}, \tag{4.2}$$

where $\mathcal{K}$ is the solution space of the homogeneous Neumann boundary value problem,

$$\mathcal{K} \;:=\; \left\{ v \in H^1(\Omega) \;:\; Lv = 0 \text{ in } \Omega, T_x v = 0 \text{ on } \Gamma \right\}, \tag{4.3}$$

and that the solution of (4.1) is unique only modulo $\mathcal{K}$.

Let us denote

$$H_0^{1/2}(\Gamma) \;:=\; \{ v \in H^{1/2}(\Gamma) \;:\; \langle v, w \rangle_{L^2(\Gamma)} = 0 \quad \text{for all } w \in \mathcal{K} \}. \tag{4.4}$$

For the numerical solution of (4.1) we may consider the hypersingular boundary integral equation

$$(Du)(x) \;=\; \left(\frac{1}{2} I - K' \right) f(x) \quad \text{for } x \in \Gamma, \tag{4.5}$$

where the boundary integral operators are defined as in the previous section. If f satisfies (4.2), then there exists a unique solution $u \in H_0^{1/2}(\Gamma)$ satisfying (4.5). Let

$$W_h \;=\; \operatorname{span}\{\varphi_k\}_{k=1}^M \subset H_0^{1/2}(\Gamma) \tag{4.6}$$

be a boundary element trial space of continuous and piecewise linear splines with respect to a triangulation $\Gamma_h = \Gamma_{h_I}$ for some $1 \le I < J$; see section 2. The Galerkin formulation of (4.5) reads to find $u_h \in W_h$ such that

$$\langle Du_h, v_h \rangle_{L^2(\Gamma)} \;=\; \left\langle \left(\frac{1}{2} I - K' \right) f, v_h \right\rangle_{L^2(\Gamma)} \qquad \text{for all } v_h \in W_h. \tag{4.7}$$

It is well known that there exists a unique solution $u_h \in W_h$ of (4.7) satisfying the error estimate

$$||u - u_h||_{H^{1/2}(\Gamma)} \;\le\; c \cdot \inf_{v_h \in W_h} ||u - v_h||_{H^{1/2}(\Gamma)} \;\le\; c \cdot h^{s-\frac{1}{2}} \cdot ||u||_{H^s(\Gamma)} \tag{4.8}$$

if $u \in H^s(\Gamma)$ and $s \le 2$. It was shown in [19] that the error

$$e_h(x) \;=\; (u - u_h)(x) \in H_0^{1/2}(\Gamma) \tag{4.9}$$

is a solution of the second kind boundary integral equation

$$\left(\frac{1}{2} I + K \right) (u - u_h)(x) \;=\; (\widetilde{u} - u_h)(x) \quad \text{for } x \in \Gamma, \tag{4.10}$$

where $\widetilde{u}$ is given as

$$\widetilde{u}(x) \;=\; (Vf)(x) + \left(\frac{1}{2} I - K \right) u_h(x) \quad \text{for } x \in \Gamma.$$

Let $P_{\mathcal{K}} : H^{1/2} \to H_0^{1/2}(\Gamma)$. Then we can compute an approximate error function by the truncated Neumann series

$$e_h^{(q)} = \sum_{\ell=0}^{q} \left[P_{\mathcal{K}} \left(\frac{1}{2} I - K \right) \right]^{\ell} P_{\mathcal{K}} (\widetilde{u} - u_h)(x) \quad \text{for } x \in \Gamma \tag{4.11}$$

for some $q \geq 0$ to be used as an error estimator for (4.9); see [19] for details. As in the case of the Dirichlet problem we introduce a trial space $W_{\tilde{h}}$ with $W_h \subset W_{\tilde{h}}$ to realize the successive approximation (4.11), i.e., we compute

$$\widetilde{e}_h^{(q)} = \sum_{\ell=0}^{q} \left[\hat{Q}_{\tilde{h}} P_{\mathcal{K}} \left(\frac{1}{2} I - K \right) \right]^{\ell} \hat{Q}_{\tilde{h}} P_{\mathcal{K}} (\widetilde{u} - u_h)(x) \quad \text{for } x \in \Gamma, \tag{4.12}$$

where $\hat{Q}_{\tilde{h}}$ is now the L^2 Galerkin projection onto $W_{\tilde{h}}$. We have to assume that $\hat{Q}_{\tilde{h}} : H^1(\Gamma) \to W_{\tilde{h}} \subset H^1(\Gamma)$ is bounded. To ensure this for adaptively refined meshes we need to formulate some additional mesh restrictions [5]. For example, when using piecewise linear trial functions on a curve, the mesh ratio of neighbored elements has to be bounded by 4; see [5, Theorem 2].

Based on the approximated error function (4.12) we can compute a global error estimator by

$$\widetilde{\eta}^{(q)} = ||\widetilde{e}_h^{(q)}||_D, \tag{4.13}$$

where we used the energy norm induced by the hypersingular integral operator D.

THEOREM 4.1 (see [19, Theorem 3.1]). *Let $\widetilde{\eta}^{(q)}$ as defined in* (4.13). *Then,*

$$\frac{1}{1 + c_K^{q+1}} \left\{ \eta^{(q)} - c \cdot q \cdot \widetilde{h}^{1/2} ||\widetilde{e}_h^{(0)}||_{L^2(\Gamma)} \right\} \leq ||\widetilde{e}_h||_D \leq \frac{1}{1 - c_K^{q+1}} \left\{ \eta^{(q)} + c \cdot q \cdot \widetilde{h}^{1/2} ||\widetilde{e}_h^{(0)}||_{L^2(\Gamma)} \right\}.$$

For the computation of all global and local error indicators we consider the case $n = 2$ first. Since the energy norm induced by the hypersingular integral operator of the Laplace equation is spectrally equivalent to any other norm in $H_0^{1/2}(\Gamma)$ it is sufficient to consider the norm induced by the hypersingular integral operator of the Laplacian only. Moreover, for the boundary integral operators related to the Laplace operator it is known that there holds

$$\langle Du, v \rangle_{L^2(\Gamma)} = \left\langle V \frac{d}{ds} u, \frac{d}{ds} v \right\rangle_{L^2(\Gamma)} \quad \text{for all } u, v \in H_0^{1/2}(\Gamma).$$

Since $\widetilde{e}_h^{(q)} \in W_{\tilde{h}}$ we get that $\frac{d}{ds} \widetilde{e}_h^{(q)} =: w_{\tilde{h}}^{(q)} \in Z_{\tilde{h}}$, where $Z_{\tilde{h}}$ is the trial space of piecewise constant splines as introduced in the previous section. Therefore,

$$||\widetilde{e}_h^{(q)}||_D^2 = \langle D \widetilde{e}_h^{(q)}, \widetilde{e}_h^{(q)} \rangle_{L^2(\Gamma)} = \langle V w_{\tilde{h}}^{(q)}, w_{\tilde{h}}^{(q)} \rangle_{L^2(\Gamma)}.$$

Since the single layer potential operator $V : H^{-1/2}(\Gamma) \to H^{1/2}(\Gamma)$ is positive definite and bounded, we conclude that

$$c_1 \cdot ||w_{\tilde{h}}^{(q)}||_{H^{-1/2}(\Gamma)}^2 \leq ||\widetilde{e}_h^{(q)}||_D^2 \leq c_2 \cdot ||w_{\tilde{h}}^{(q)}||_{H^{-1/2}(\Gamma)}^2,$$

and hence it is sufficient to compute and localize an equivalent Sobolev norm in $H^{-1/2}(\Gamma)$ for $w_{\tilde{h}}^{(q)} \in Z_{\tilde{h}}$.

For $n = 3$ we can apply a similar approach. Using (see [6])

$$\begin{aligned}\langle Du, v\rangle_{L_2(\Gamma)} &= \frac{1}{4\pi}\int_\Gamma\int_\Gamma \frac{1}{|x-y|}\vec{\mathrm{curl}}_\Gamma u(x)\cdot\vec{\mathrm{curl}}_\Gamma v(x)ds_x ds_y \\ &= \sum_{i=1}^{3}\langle V(\vec{\mathrm{curl}}_\Gamma u)_i, (\vec{\mathrm{curl}}_\Gamma v)_i\rangle_{L_2(\Gamma)},\end{aligned}$$

the energy norm induced by the hypersingular integral operator for the Laplacian can be written again as energy norm induced by the single layer potential. Then we can follow the same ideas as described above for each component of $\vec{\mathrm{curl}}_\Gamma \widetilde{e}_h^{(q)}$.

5. Adaptive boundary element schemes. Now we are in a position to describe an adaptive boundary element method for the numerical solution of either the Dirichlet boundary value problem (3.1) or the Neumann boundary value problem (4.1) by means of the Galerkin formulation (3.3) or (4.7), respectively.

The adaptive boundary element algorithm reads as follows:

1. For a given boundary element mesh Γ_h compute either a boundary element solution t_h of (3.3) (Dirichlet) or $u_h \in W_h$ of (4.7) (Neumann).
2. Compute the approximate error functions $\widetilde{e}_h^{(q)}$ for some $q \geq 0$ by means of the truncated Neumann series (3.11) (Dirichlet) or (4.12) (Neumann).
3. In case of the Neumann problem (n=2) compute $w_{\tilde{h}}^{(q)} = \frac{d}{ds}\widetilde{e}_h^{(q)} \in Z_{\tilde{h}}$.
4. Compute $v_{\tilde{h}} = A^{(-\frac{1}{4})}\widetilde{e}_h^{(q)}$ (Dirichlet) or $v_{\tilde{h}} = A^{(-\frac{1}{4})}w_{\tilde{h}}^{(q)}$ (Neumann, n=2) via the multilevel representation (2.6).
5. Compute all local error indicators $\widetilde{\eta}_k^{(q)} = ||v_{\tilde{h}}||_{L^2(\Gamma_{I,k})}$ and the global error estimator $\widetilde{\eta}^{(q)} = \sqrt{\sum_{k=1}^{N_I}[\widetilde{\eta}_k^{(q)}]^2}$. Stop, if a required accuracy is reached.
6. Refine all boundary elements Γ_k, where

$$\widetilde{\eta}_k^{(q)} \geq \theta \cdot \max_{1\leq\ell\leq\widetilde{N}} \widetilde{\eta}_\ell^{(q)}$$

is satisfied with some appropriate chosen refinement parameter $\theta \in [0, 1]$.

For the Neumann problem and $n = 3$ some slight modifications in steps 3 and 4 as described in the previous section are needed.

6. Numerical results. As a numerical example we consider for both the Dirichlet and Neumann boundary value problem the two-dimensional Laplace equation, where Ω is the L shaped domain described by the nodes $(0, 0)$, $(0.25, 0)$, $(0.25, 0.25)$, $(-0.25, 0.25)$, $(-0.25, -0.25)$, and $(0, -0.25)$.

Let Z_h and W_h denote the boundary element spaces of piecewise constant and piecewise linear trial functions, respectively. Then we define $Z_{\tilde{h}}$ by piecewise constant basis functions and $W_{\tilde{h}}$ by piecewise linear ones where the mesh size in both cases is $\tilde{h} = h/8$, i.e., we apply three additional refinement steps to construct the underlying boundary element mesh $\Gamma_{\tilde{h}}$ from Γ_h.

In both examples we have chosen $q = 0$ in the truncated Neumann series (3.11) and (4.12), respectively. While the corresponding error functions $\widetilde{e}_h^{(0)}$ already define equivalent error estimators, the constants in the equivalence inequalities and therefore

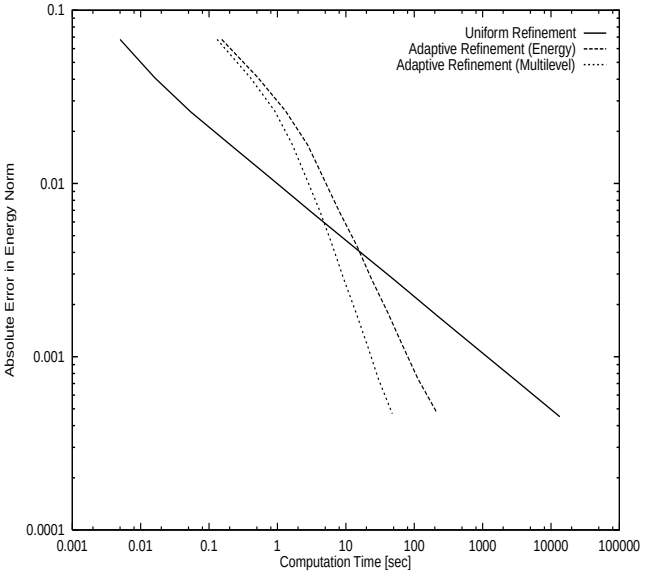


FIG. 1. *Uniform vs. adaptive refinement, Dirichlet problem.*

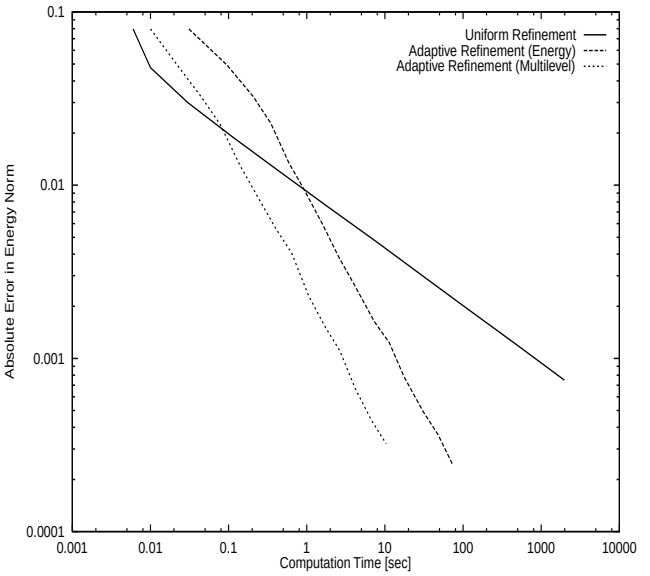


FIG. 2. *Uniform vs. adaptive refinement, Neumann problem.*

the accuracy of the error estimator can be improved when using some $q > 0$; see [18] for a more detailed discussion on this.

6.1. Dirichlet problem. For the Dirichlet boundary value problem the boundary data are given in such a way that the solution is

$$u(x) = u(r,\varphi) = r^{2/3} \sin\left(\frac{2}{3}\varphi\right) \tag{6.1}$$

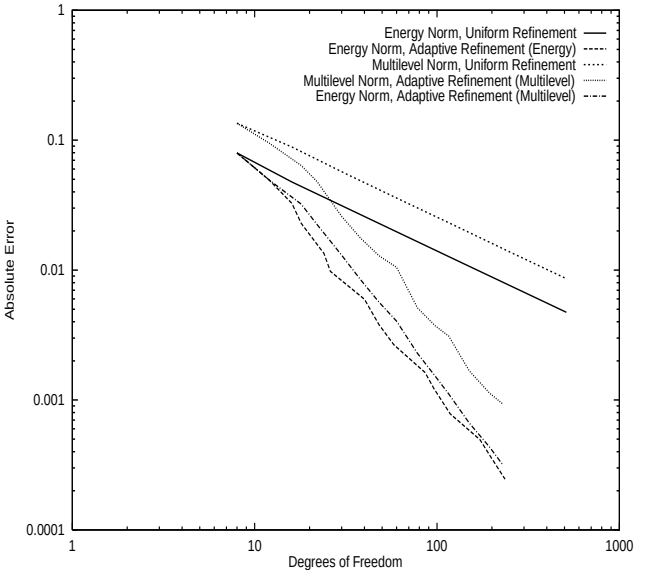


FIG. 3. *Errors for adaptive refinement, Neumann problem.*

TABLE 1
Errors for uniform refinement, Neumann problem.

N	$\|\|u-u_h\|\|_D$	Order	$\|\|A^{-\frac{1}{4}}\frac{d}{ds}(u-u_h)\|\|_{L^2}$	Order
8	7.98 –2		1.35 –1	
16	4.77 –2	0.74	8.86 –2	0.61
32	3.00 –2	0.67	5.47 –2	0.70
64	1.89 –2	0.67	3.44 –2	0.67
128	1.19 –2	0.67	2.17 –2	0.66
256	7.50 –3	0.67	1.36 –2	0.67
512	4.73 –3	0.67	8.59 –3	0.66

when using polar coordinates. In this case we get $t \in H^\sigma(\Gamma)$ with $\sigma < \frac{1}{6}$ for the solution t of the boundary integral equation (3.2). Therefore, in case of an uniform refinement the expected rate of convergence is $\frac{2}{3}$ when measuring the error $t-t_h$ in an equivalent norm in $H^{-1/2}(\Gamma)$; see (3.4). In Table 2 we give the results when using the energy norm $||\cdot||_V$ induced by the single layer potential V and the multilevel norm (2.9). In the case of a regular solution $t \in H^1(\Gamma)$ the expected order of convergence is supposed to be $\frac{3}{2}$. To improve the order of convergence we used the adaptive boundary element algorithm as described in section 5 with the refinement parameter $\theta = 0.05$. The results are given in Figure 4, for comparison we give also the error curves when using the energy norm $||\cdot||_V$ to compute all local and global error indicators; see [18]. Note that the energy error of the solution based on the multilevel approach converges even faster than using the adaptive approach based on the energy norm. On the other hand we can notify the norm equivalence as stated in the theory. Note that *adaptive refinement (energy)* marks the use of the adaptive refinement strategy based on local error indicators computed by the energy norm while *adaptive refinement (multilevel)* describes the results in case of the multilevel computation of the local indicators.

One motivation for the use of multilevel methods in the computation of local and global Sobolev norms was the expected efficiency due to the minimal amount of work

TABLE 2
Errors for uniform refinement, Dirichlet problem.

N	$\|\|t - t_h\|\|_V$	Order	$\|\|A^{-\frac{1}{4}}(t - t_h)\|\|_{L^2}$	Order
8	6.79 –2		1.16 –1	
16	4.08 –2	0.72	7.94 –2	0.55
32	2.59 –2	0.66	4.92 –2	0.69
64	1.65 –2	0.65	3.12 –2	0.66
128	1.06 –2	0.64	2.01 –2	0.63
256	6.74 –3	0.65	1.31 –2	0.62
512	4.30 –3	0.65	8.48 –3	0.63

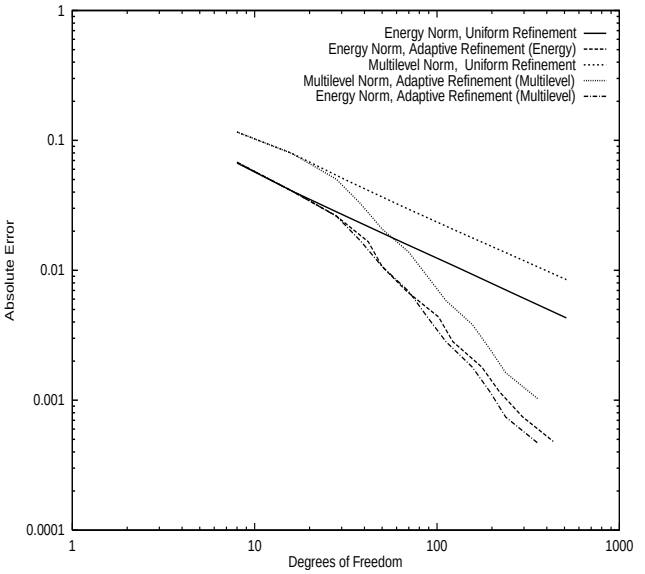


FIG. 4. *Errors for adaptive refinement, Dirichlet problem.*

involved in the computation of Sobolev norms. Therefore we give a comparison of the computing times in Figure 1, where we consider an uniform refinement, an adaptive strategy based on the energy norm [18] and the algorithm based on the multilevel norm as proposed in this paper. Note that in the uniform case the computing time includes the discretization and solution process at one level only, while in the adaptive algorithms the total computing time includes discretization, solution, and error estimation over all previous mesh levels. As expected from the theory the amount of work of the multilevel approach is less than the method based on the energy norm while both algorithms produce almost the same adaptive boundary element mesh.

6.2. Neumann problem. In case of a Neumann boundary value problem for the Laplace equation the boundary conditions are given in such a way that the solution is

$$u(x) \;=\; u(r,\varphi) \;=\; r^{2/3}\;\cos\left(\frac{2}{3}\varphi\right). \tag{6.2}$$

In this case we get $u \in H^{\rho}(\Gamma)$ with $\rho < \frac{7}{6}$ for the solution of (4.5). Hence, the order of convergence is about $\frac{2}{3}$ when measuring the error in an equivalent Sobolev norm in

$H^{1/2}(\Gamma)$; this is well documented numerically in Table 1 when using both the energy norm $||\cdot||_D$ and the multilevel norm (2.9).

For a regular solution $u \in H^2(\Gamma)$ the expected rate of convergence is about $\frac{3}{2}$. Hence, using the adaptive algorithm as described in section 5 with $\theta = 0.25$ we can improve the order of convergence to get almost the optimal behavior. This is documented for both the energy and the multilevel norm in Figure 3. In addition to the results based on the multilevel approach we also give the results for the adaptive algorithm based on the energy norm [19]; see Figure 2.

Acknowledgment. The financial support by the ISC is gratefully acknowledged.

REFERENCES

[1] R. E. Bank, *Hierarchical bases and the finite element method*, Acta Numer., 5 (1996), pp. 1–43.
[2] J. H. Bramble, J. E. Pasciak, and P. S. Vassilevski, *Computational scales of Sobolev norms with applications in preconditioning*, Math. Comp., 69 (2000), pp. 463–480.
[3] J. H. Bramble, J. E. Pasciak, and J. Xu, *Parallel multilevel preconditioners*, Math. Comp., 55 (1990), pp. 1–22.
[4] C. Carstensen and E. P. Stephan, *Adaptive boundary element methods for some first kind integral equations*, SIAM J. Numer. Anal., 33 (1996), pp. 2166–2183.
[5] M. Crouzeix and V. Thomée, *The stability in L_p and W_p^1 of the L_2 projection onto finite element function spaces*, Math. Comp., 48 (1987), pp. 521–532.
[6] R. Dautray and J.-L. Lions, *Mathematical Analysis and Numerical Methods for Science and Technology. Volume* 4: *Integral Equations and Numerical Methods*, Springer, Berlin, 1990.
[7] B. Faermann, *Local a posteriori error estimators for the Galerkin discretization of boundary integral equations*, Numer. Math., 79 (1998), pp. 43–76.
[8] B. Faermann, *Localization of the Aronszajn–Slobodeckij norm and application to adaptive boundary element methods*, *Part* I. *The two–dimensional case*, IMA J. Numer. Anal., 20 (2000), pp. 203–234.
[9] N. Heuer and E. P. Stephan, *Iterative substructuring for hypersingular integral equations in* $\mathbb{R}^3$, SIAM J. Sci. Stat. Comput., 20 (1999), pp. 739–749.
[10] G. C. Hsiao and W. L. Wendland, *A finite element method for some integral equations of the first kind*, J. Math. Anal. Appl., 58 (1977), pp. 449–481.
[11] M. Maischak, P. Mund, and E. P. Stephan, *Adaptive multilevel BEM for acoustic scattering*, Comput. Methods Appl. Mech. Engrg., 150 (1997), pp. 351–367.
[12] P. Mund and E. P. Stephan, *An adaptive two–level method for the coupling of nonlinear FEM–BEM equations*, SIAM J. Numer. Anal., 36 (1999), pp. 1001–1021.
[13] P. Mund, E. P. Stephan, and J. Weisse, *Two–level methods for the single layer potential in* $\mathbb{R}^3$, Computing, 60 (1998), pp. 243–266.
[14] P. Oswald, *Multilevel Finite Element Approximation*, Teubner–Verlag, Stuttgart, 1994.
[15] P. Oswald, *Multilevel norms for* $H^{-1/2}$, Computing, 61 (1998), pp. 235–255.
[16] U. Rüde, *Mathematical and Computational Techniques for Multilevel Adaptive Methods*, Frontiers Appl. Math. 13, SIAM, Philadelphia, 1993.
[17] H. Schulz, *Über lokale und globale Fehlerabschätzungen für adaptive Randelementmethoden*, Doctoral Thesis, Universität Stuttgart, Germany, 1997.
[18] H. Schulz and O. Steinbach, *A new a posteriori error estimator in adaptive direct boundary element methods, the Dirichlet problem*, Calcolo, 37 (2000), pp. 79–96.
[19] H. Schulz and O. Steinbach, *A new a posteriori error estimator in adaptive direct boundary element methods, the Neumann problem*, in Multifield Problems. State of the Art, A. M. Sandig, W. Schiehlen, and W. L. Wendland, eds., Springer, Berlin, 2000, pp. 201–208.
[20] E. P. Stephan and T. Tran, *Additive Schwarz method for the h–version boundary element method*, Appl. Anal., 60 (1996), pp. 63–84.
[21] W. L. Wendland and De-Hao Yu, *Adaptive boundary element methods for strongly elliptic integral equations*, Numer. Math., 53 (1988), pp. 539–558.
[22] J. Xu, *Theory of Multilevel Methods*, Ph.D. thesis, Cornell University, Ithaca, NY, 1989.
[23] J. Xu, *An introduction to multilevel methods*, in Wavelets, Multilevel Methods and Elliptic PDE's, M. Ainsworth, J. Levesley, M. Marletta, and W. A. Light, eds., Numerical Mathematics and Scientific Computation, Clarendon Press, Oxford University Press, New York, 1997, pp. 213–302.

SIAM J. SCI. COMPUT.
Vol. 22, No. 2, pp. 617–632
© 2000 Society for Industrial and Applied Mathematics

OPTIMAL DECOMPOSITION OF THE DOMAIN IN SPECTRAL METHODS FOR WAVE-LIKE PHENOMENA*

CARL ERIK WASBERG† AND DAVID GOTTLIEB‡

Abstract. A strategy for determining the optimal number of grid points and subdomains in a spectral method with domain decomposition on a serial computer is presented. The rapidly growing computational cost for large numbers of grid points in each subdomain is balanced against the exponential convergence for spectral approximation of smooth functions, and the optimum is found as the number of grid points and subdomains that gives the minimal computational cost for a given accuracy. The typical length scale of the problem is found to influence the number of subdomains but not the number of grid points within each subdomain.

Key words. domain decomposition, spectral methods

AMS subject classification. 65M70

PII. S1064827597322306

1. Introduction.

Domain decomposition methods have been extensively used for numerical simulations involving spectral methods, as they can offer more flexibility in treating complicated domains as well as computational savings. In fact, when applied to incompressible flows, the spectral element method [16], based on domain decomposition, seems to be the method of choice. The application of domain decomposition spectral methods to hyperbolic equations has been considered, e.g., by Kopriva [12], Kopriva and Kolias [14], and Quarteroni [17]. The advent of parallel computers has made multidomain methods very popular.

In this paper we consider spectral Chebyshev collocation methods with domain decomposition applied to problems whose solutions display wave-like behavior. By wave-like we mean that the structure is mostly homogeneous over the whole computational domain, or at least that the spatial resolution requirements are the same throughout. This does not necessarily exclude situations with, e.g., boundary layers, if suitable coordinate transformations are applied in the areas where fine structure is expected. Such an approach leads also to natural load-balancing in parallel implementations, as the computational complexity will be similar for all subdomains. The analysis in this paper extends to complex geometries, as long as the domain can be built up by transformed squares, or three-dimensional (3-D).

Our main subject is to find the number of subdomains, d, and number of grid points per subdomain, N, in each spatial direction that gives a solution of given accuracy with minimal computational work. Numerical experiments illustrating this problem can by found, e.g., in [10, 11, 15, 19]. Our approach is based on models for the error and for the complexity of the algorithm as functions of these domain decomposition parameters.

We assume that the total accuracy of an algorithm can be estimated by the spatial approximation accuracy, i.e., that other tasks such as time-stepping or implicit

*Received by the editors June 4, 1997; accepted for publication (in revised form) March 15, 1999; published electronically August 3, 2000.
http://www.siam.org/journals/sisc/22-2/32230.html
†FFI, PO Box 25, 2027 Kjeller, Norway (carl-erik.wasberg@ffi.no).
‡Division of Applied Mathematics, Brown University, Providence, RI 02912 (dig@cfm.brown.edu).
The research of this author was supported by AFOSR grant F49620-96-1-0150 and NSF grant DMS-9500814.

solutions can be done sufficiently accurately. An error estimate for the approximation of sine waves is used, and the desired spatial resolution determines the highest wavenumber it is required to resolve. The approximation error decays exponentially in N (see section 2), so using more points in each subdomain increases the accuracy substantially. Of course, the number of waves that each subdomain "sees" decreases linearly with the number of subdomains.

For a given numerical algorithm, the computational work can be estimated experimentally by timing or theoretically through operation counts. The computational work in a spectral method scales with some power of the number of points in each subdomain, and this growth is usually faster than the growth with an increasing number of subdomains. (See the algorithm examples in section 5.) Thus, it is computationally cheaper to use many subdomains with few points in each subdomain, provided that the required accuracy can be achieved. Our models of computational work do not include costs of memory access (e.g., the use of cache), which could introduce discontinuities in the work functions. We note only that the tensor-product structure of the basis functions in spectral methods and the use of matrix multiplication for evaluation of spatial derivatives facilitate efficient use of cache. Serial machines are considered in this paper, while some issues arising in parallel computations are treated in [5].

Given the models for accuracy and work, we obtain optimal values for the domain decomposition parameters N and d, depending on the required accuracy. The optimal values represent the balance between the high accuracy of a large number of points in each subdomain and the computational gain of many subdomains with fewer points in each. We do the analysis for quite general work functions, and we show examples in sections 5 and 6 of how given algorithms fit into this framework.

The paper is organized as follows. We start with a brief introduction to spectral collocation methods with Chebyshev polynomials in section 2. In section 3 we introduce the models for the error and for the computational work. The problem of minimizing the work for a given accuracy is analyzed in section 4, and the results are discussed and related to algorithm examples in section 5. In section 6, the theoretical results are compared with results of numerical experiments of function approximation and PDE-solution. Finally, we comment on the relation to high-order finite difference methods in section 7 before summing up with the conclusions in section 8.

2. Chebyshev collocation methods. Spectral methods are based on representing the numerical solution on a form that can be differentiated analytically to produce good approximations to the spatial derivatives. The representations used in the early spectral methods were truncated Fourier series, but polynomial expansions have become widely used. In spectral collocation methods, the expansion coefficients are determined from discrete transforms, i.e., they are based only on the values of the numerical solution at discrete points. The resulting representation interpolates the numerical solution at these points.

In more than one spatial dimension, the basis functions are tensor products of the one-dimensional (1-D) ones, and each direction can be scaled or mapped to change the form of the computational domain. In the following, we briefly review some basic theory for Chebyshev spectral methods that can be found, e.g., in the book by Canuto et al. [3].

The Chebyshev polynomial of order k can be written as a cosine function with a change of variable:

$$T_k(x) = \cos(k \arccos x), \qquad k = 0, 1, 2, \ldots, \tag{2.1}$$

and the Chebyshev polynomials are orthogonal in the weighted function space $L^2_w(-1,1)$, with the weight function

$$w(x) = (1-x^2)^{-1/2}. \tag{2.2}$$

The discrete Chebyshev coefficients of a function $v(x)$, defined on $[-1,1]$, with respect to the collocation points

$$x_j = \cos \pi j/N, \qquad j = 0,1,\ldots,N, \tag{2.3}$$

are defined by

$$\widetilde{v}_k = \frac{2}{N\bar{c}_k} \sum_{j=0}^{N} \frac{1}{\bar{c}_j} v(x_j) T_k(x_j), \quad k = 0,1,\ldots,N, \tag{2.4}$$

where $\bar{c}_k = 1$, $k = 1,\ldots,N-1$, $\bar{c}_0 = \bar{c}_N = 2$. Equation (2.4) gives the transformation from the physical space of point values to the spectral space of coefficients. The inverse transformation is the interpolating function

$$I_N v(x) = \sum_{k=0}^{N} \widetilde{v}_k T_k(x). \tag{2.5}$$

An error estimate involving derivatives up to order l can be found in the norm of the weighted Sobolev space $H^l_w(-1,1)$:

$$\|v - I_N v\|_{H^l_w(-1,1)} \le C N^{2l-m} \|v\|_{H^m_w(-1,1)}, \qquad m \ge 1, \quad 0 \le l \le m, \tag{2.6}$$

when $v \in H^m_w(-1,1)$. If $v(x)$ is smooth, then m in (2.6) can be chosen arbitrarily large, with the result that the approximation error decays faster than any algebraic power of N as $N \to \infty$. The approximation is then said to be exponentially or spectrally accurate. An attractive aspect of using the Chebyshev polynomials is that simple expressions for the polynomials and the collocation points are available because of the relation (2.1) to cosine functions, and the transform (2.4) and the evaluation of (2.5) at the collocation points (2.3) can be carried out by fast Fourier transforms (FFTs) for suitable even values of N.

Approximate spatial derivatives can be calculated by differentiating the interpolating function (2.5). The coefficients $\widetilde{v}'_k$ for the derivative are given by

$$\begin{aligned} \widetilde{v}'_N = 0, \qquad & \widetilde{v}'_{N-1} = 2N\widetilde{v}_N, \\ \bar{c}_k \widetilde{v}'_k = \widetilde{v}'_{k+2} + 2(k+1)\widetilde{v}_{k+1}, \qquad & k = 0,1,\ldots,N-2. \end{aligned} \tag{2.7}$$

The differentiation operation can also be expressed as $\mathbf{u}_x = D\mathbf{u}$, where $\mathbf{u}$ is the vector of function values at the collocation points and $\mathbf{u}_x$ is the vector of approximate values for the derivative at the same points. From the description given above about calculating coefficients for the derivative, it is clear that the derivative operator can be written $D = T^{-1}\widetilde{D}T$, where T is the transformation (2.4) from physical space to spectral space, $\widetilde{D}$ is the derivative matrix in spectral space, and T^{-1} is the evaluation of (2.5) at the collocation points with the new coefficients. For small values of N, it is usually more efficient to form the derivative matrix D explicitly and calculate the derivative by matrix/vector multiplication than to use FFTs for the operations T and T^{-1}. The crossover point depends heavily on the computer architecture and the

implementation details. It was found to be around $N = 64$ in the calculations on a CRAY X-MP in [20], while calculations on scalar computers would give smaller numbers. We refer to [4] for a discussion with several timing examples.

This review has concentrated on the spatial approximation properties. Time-dependent PDEs can be solved using the method of lines. This means that the spatial derivatives are approximated as described above, and the resulting semidiscrete problem

$$\mathbf{u}_t = \mathbf{f}(\mathbf{u}), \qquad \mathbf{u} = (u(x_0), \ldots, u(x_N))^T \tag{2.8}$$

is solved by a standard method for systems of ordinary differential equations [9]. Explicit Runge–Kutta methods are used in the example presented in section 6.2, with boundary conditions applied at the end of each time-step.

With explicit time-integration methods, and also many implicit methods, there is a stability restriction on the allowable time-steps, depending on the operator $\mathbf{f}$ in (2.8). For linear and quasi-linear systems, the maximum explicit time-steps with Chebyshev collocation are $O(1/N^2)$ for first-order problems and $O(1/N^4)$ for second-order problems. Explicit time-integration is therefore usually too expensive for nonperiodic problems involving second derivatives.

3. Models for accuracy and computational work. The Chebyshev expansion of a sine function with k wavelengths in the interval $[-1, 1]$ is

$$\sin(k\pi x) = \sum_{n=0}^{\infty} a_n T_n(x), \qquad -1 \le x \le 1, \tag{3.1a}$$

with (Galerkin) expansion coefficients [8, Eq. (3.41)] (or from [1, Eq. (9.1.21)])

$$\begin{aligned} a_n &= \frac{2}{\pi c_n} \int_{-1}^{1} \sin(k\pi x)\, T_n(x)\, w(x)\, dx \\ &= \frac{2}{c_n} J_n(k\pi) \sin(n\pi/2), \qquad n = 0, 1, \ldots, \end{aligned} \tag{3.1b}$$

where $c_0 = 2$, $c_n = 1$ $(n > 0)$, and J_n is the Bessel function of the first kind of order n. Since $J_n(k\pi) \to 0$ exponentially fast as n increases beyond $k\pi$, the estimated truncation error $E(N, k)$ of an N-term Chebyshev expansion of $\sin(k\pi x)$ is given by the upper bound of the coefficient a_N, i.e.,

$$E(N, k) = 2J_N(k\pi), \qquad N > k\pi.$$

Using the asymptotic form (as $\nu \to \infty$) [1, Eq. (9.3.1)]

$$J_\nu(x) \sim \frac{1}{\sqrt{2\pi\nu}} \left(\frac{ex}{2\nu}\right)^\nu,$$

we obtain the estimate

$$E(N, k) = \sqrt{\frac{2}{\pi N}} \left(\frac{e\pi k}{2N}\right)^N. \tag{3.2}$$

The error estimate can also be used for a collocation method, since the interpolation error is asymptotically of the same order as the truncation error. Figure 3.1 shows the estimate (3.2) compared with the measured maximum error for approximation of the

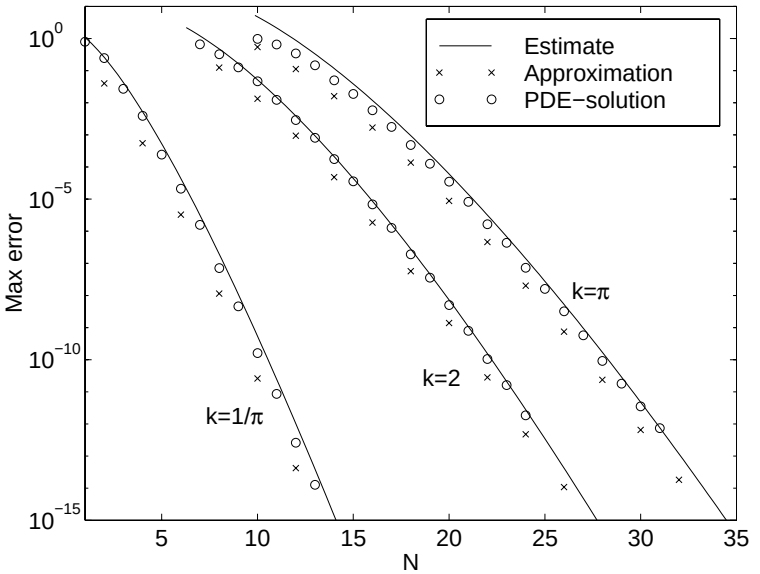


FIG. 3.1. *Estimated* (3.2) *and numerically measured* 1*-domain Chebyshev collocation approximation error for the function* $\sin(kx)$*,* $-1 \le x \le 1$*, with* $k = 1/\pi, 2, \pi$*, compared with the errors in the solution of the* 2*-D advection problem* (6.1) *with the same wavenumbers in the initial and boundary conditions.*

function $\sin(k\pi x)$ for $k = 1/\pi, 2, \pi$ and the maximum error of the solution of the two-dimensional (2-D) advection equation (see section 6.2) with the same wavenumbers in the initial and boundary conditions. While the error estimate is too conservative by factors between 4 and 20 in the approximation case, we note that the estimated values of N for a given accuracy are still generally only 1 grid point error away from the measured values. In the PDE-examples, the error estimates are within a factor of 2 grid point errors of the measured errors for the majority of the data.

For a Legendre expansion, an expression similar to (3.2) can be obtained using [8, Eq. (3.45)] instead of (3.1).

If the computational domain is divided into d subdomains in each spatial direction, the wavenumber on each subdomain becomes k/d, and the new error function is

$$E(N, k, d) = \sqrt{\frac{2}{\pi N}} \left(\frac{e\pi k}{2dN} \right)^N. \tag{3.3}$$

In higher dimensions, N is the number of grid points in each spatial direction in each subdomain.

We assume that the computational work required to solve the problem in question is given by a function $W(N, d)$. The simplest of the models we consider in this paper is

$$W(N, d) = C d^r N^\alpha, \qquad \alpha > r \ge 1. \tag{3.4a}$$

The condition $\alpha > r$ ensures that it is computationally cheaper to increase the total number of points by using more subdomains than by increasing the number of

points within each subdomain. If this was not the case, the single domain approach would be preferred because of the spectral accuracy property. Motivation for domain decomposition would then have to come from aspects other than those considered in this analysis, e.g., parallelization, geometry considerations, or the need for local grid refinement.

Lower order terms in the work function may become significant for small values of N. As an example, evaluation of derivatives in 3-D by matrix multiplication requires $O(N^4)$-operations on each subdomain, while, e.g., adding up derivatives in different directions for all the grid points in a subdomain is an $O(N^3)$-operation. We therefore consider also work functions on the form

$$W(N,d) = Cd^r(N^\alpha + \beta N^{\alpha-1}), \qquad \alpha > r \geq 1, \quad \beta > 0. \tag{3.4b}$$

For larger values of N, it may be more efficient to apply FFTs instead of matrix multiplication for the transformations between the physical and spectral space. In that case the work function includes a logarithmic term:

$$W(N,d) = Cd^r N^{\alpha-1} \log_2 N, \qquad \alpha - 1 > r \geq 1. \tag{3.4c}$$

Examples of the parameters r and α in given algorithms are given in section 5.

We can now formulate the problem described in the introduction more precisely.

PROBLEM 1. *Minimize $W(N,d)$ under the constraint*

$$E(N,k,d) = e^{-\varepsilon}. \tag{3.5}$$

As $\varepsilon = -\ln E$, the values of ε would be in the range between 4 ($E = 1.8 \cdot 10^{-2}$, low accuracy) and 30 ($E = 9.4 \cdot 10^{-14}$, close to standard double precision machine accuracy). The problem is analyzed in the next section for the different work functions given by (3.4a)–(3.4c).

4. Analysis. Combining the error function (3.3) and the constraint (3.5) gives the following expression for the number of subdomains, d:

$$d = \frac{e\pi k}{2N} \left(\frac{2}{\pi N}\right)^{\frac{1}{2N}} e^{\varepsilon/N}. \tag{4.1}$$

This is used in the analysis below to eliminate d from the work functions. For each of the three work functions (3.4a)–(3.4c), examples of the optimal number of points per subdomain and the optimal number of points per wavelength for different values of the parameters r and α are shown in Figures 4.1 and 4.2.

Case (a). Inserting (4.1) into the simple work function (3.4a) and setting $\frac{\partial W}{\partial N} = 0$ gives the following relation for the optimal number of points in each subdomain:

$$(\alpha - r)N = r\varepsilon + \frac{r}{2}\left(1 - \ln\frac{\pi N}{2}\right). \tag{4.2}$$

Thus the optimal N is approximately given by

$$N_{\text{opt}} \approx \frac{r\varepsilon}{\alpha - r} \tag{4.3}$$

and plotted for different values of r and α in Figures 4.1 and 4.2.

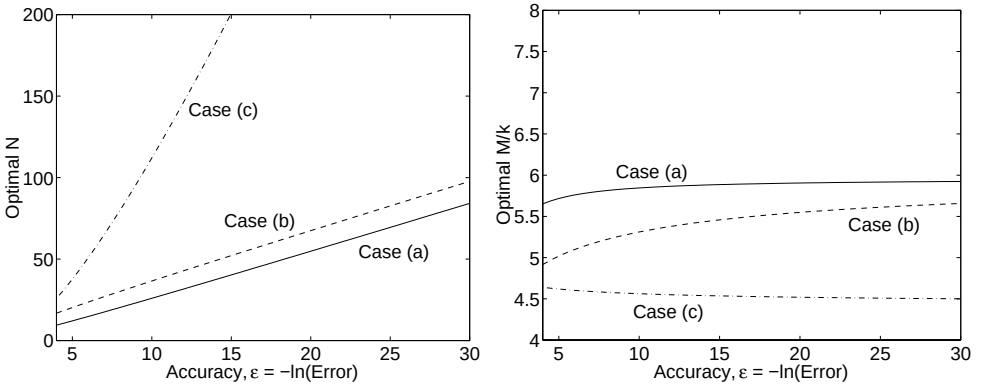


FIG. 4.1. *Optimal number of points in each subdomain (left) and optimal total number of points per wavelength (right) as functions of the accuracy for the different work functions* (3.4) *for* $r = 3$, $\alpha = 4$.

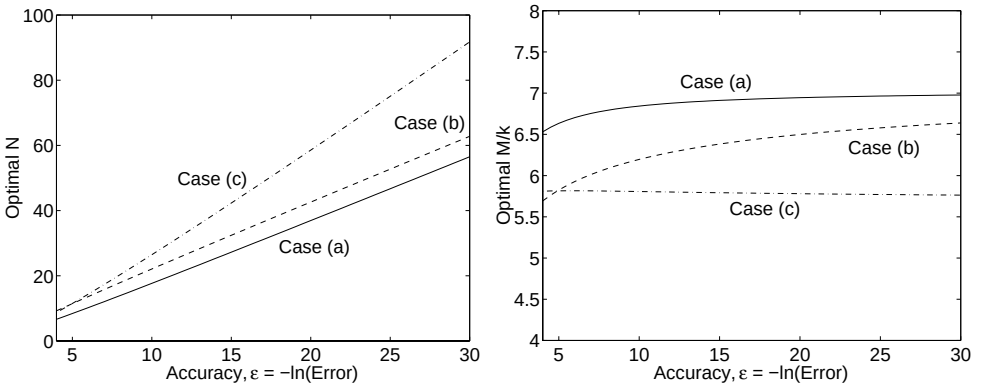


FIG. 4.2. *Optimal number of points in each subdomain (left) and optimal total number of points per wavelength (right) as functions of the accuracy for the different work functions* (3.4) *for* $r = 4$, $\alpha = 6$.

Using (4.1), the total number of grid points is

$$M = dN = \frac{e\pi k}{2}\left(\frac{2}{\pi N}\right)^{\frac{1}{2N}} e^{\varepsilon/N},$$

so by using the approximate N_{opt} from (4.3), we obtain

$$M_{\text{opt}} = d_{\text{opt}} N_{\text{opt}} \approx \frac{1}{2}\pi e^{\alpha}\left(\frac{2(\alpha - r)}{\pi r \varepsilon}\right)^{\frac{\alpha - r}{2r\varepsilon}} k.$$

The result is that the optimal total number of points per wavelength, M_{opt}/k, is increasing very slowly with the accuracy ε, as shown in Figures 4.1 and 4.2.

Case (b). Inserting (4.1) into the work function (3.4b), including a lower order term, and setting $\frac{\partial W}{\partial N} = 0$, gives the relation

$$(\alpha - r)N^2 + (\beta(\alpha - 1 - r) - r\varepsilon)\, N + (N + \beta)\frac{r}{2}\left(\ln\frac{\pi N}{2} - 1\right) - r\varepsilon\beta = 0. \tag{4.4}$$

Compared with the results from the previous section, the effect of the lower order term is slightly higher values of N_{opt}, as shown in Figures 4.1 and 4.2 (with $\beta = 16$ in (4.4)). This means that the spectral accuracy property is more dominant, and more efficient approximations are done in each subdomain. As a consequence the optimal total number of points per wavelength is lower.

Case (c). Inserting (4.1) into the work function (3.4c) including a logarithmic term, and setting $\frac{\partial W}{\partial N} = 0$, gives the relation

$$(\alpha - 1 - r)N + \frac{N}{\ln N} = r\varepsilon + \frac{r}{2}\left(1 - \ln\frac{\pi N}{2}\right). \tag{4.5}$$

Because of the slower growth of the work with N than in the previous cases, N_{opt} is higher. As explained for in Case (b), this gives a lower optimal total number of points per wavelength. These effects are also illustrated in Figures 4.1 and 4.2.

5. Discussion.

Algorithm examples. A spectral element method for the steady Stokes problem

$$\begin{aligned} -\nabla^2\mathbf{u} + \nabla p &= \mathbf{f} \quad \text{in } \Omega, \\ -\nabla\cdot\mathbf{u} &= 0 \quad \text{in } \Omega, \\ \mathbf{u} &= \mathbf{0} \quad \text{on } \partial\Omega \end{aligned} \tag{5.1}$$

is analyzed in [6], including serial work estimates. The algorithm is based on the iterative solution of a Poisson problem by a preconditioned conjugate gradient method. For a discretization in $\mathbf{R}^s$ using d elements in each direction and polynomial order N in each direction within each element, each iteration has a computational complexity of $O(d^s N^{s+1})$, and the number of conjugate gradient iterations is estimated to be of order dN. The solution of the Stokes problem is then obtained by the Uzawa iterative procedure involving an outer conjugate gradient loop that converges in $O(1)$ iterations, leaving the total complexity of the algorithm to be $O(d^{s+1}N^{s+2})$ in s spatial dimensions. This corresponds to $r = s + 1$, $\alpha = s + 2$ in the work functions (3.4a) or (3.4b). The results shown in Figure 4.1 consequently cover the 2-D version of this algorithm.

As another algorithm example, we consider the solution of a hyperbolic system in s spatial dimensions

$$\mathbf{u}_t + \sum_{i=1}^{s} A_i \mathbf{u}_{x_i} = 0 \tag{5.2}$$

with suitable initial and boundary conditions for $\mathbf{u}(\mathbf{x}, t)$. Details of solution of the Euler equations can be found, e.g., in [2, 13]. These algorithms use explicit time-integration, with time-steps proportional to $1/dN^2$. With $O(d^s N^{s+1})$-operations at each time-step, the total work for integrating one time-unit is $O(d^{s+1}N^{s+3})$, corresponding to $r = s + 1$, $\alpha = s + 3$ in the work functions (3.4a) or (3.4b). The results shown in Figure 4.2 therefore illustrate this algorithm in 3-D.

An iteration-by-subdomain method for domain decomposition is described in [18]. For an elliptic problem $Au = f$ (e.g., the Poisson problem), the problem is solved on each subdomain by an iterative method with preconditioning, and for the best preconditioners the number of iterations is independent of N. This gives $O(N^{s+1})$ work per subdomain for each of the d^s subdomains. The number of iterations in the outer iteration-by-subdomain loop is also independent of N but proportional to d.

TABLE 5.1

Factors to multiply by sigma = −ln(Error) to estimate the optimal N for an algorithm with work function $W(N,d) = Cd^{s+1}N^{s+q}$. N is the number of points in each direction in each subdomain, d is the number of subdomains in each direction, and s is the spatial dimension.

	1-D	2-D	3-D
$q = 2$	2	3	4
$q = 3$	1	3/2	2

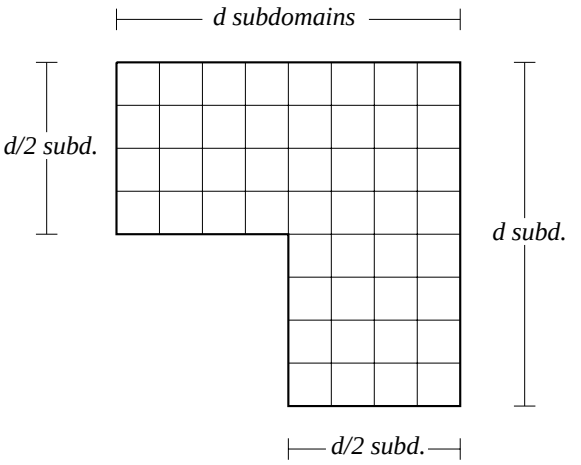


FIG. 5.1. *An L-shaped domain with 3/4 d^2 subdomains.*

This gives a total complexity of $O(d^{s+1}N^{s+1})$, i.e., $r = \alpha$ in the notation used in our work functions. As discussed in section 3, the choice of domain decomposition parameters in such a case is not covered by the analysis in this paper.

Conversely, in the discussion of the iteration-by-subdomain method for hyperbolic problems [17], Quarteroni gives the computational complexity in 1-D as $O(d^2N^3)$, because in this case effective preconditioners for the solution at each subdomain by implicit time-integration are not found. A generalization to s spatial dimensions gives an $O(d^{s+1}N^{s+2})$-algorithm, as in the spectral element example discussed above.

Dimension dependency. In most cases the work function depends on the number of spatial dimensions, s, as

$$W(N,d) = Cd^{s+p}N^{s+q}, \qquad p \geq 0, \quad q > p, \tag{5.3}$$

which gives

$$N_{\text{opt}} \approx \frac{s+p}{q-p}\varepsilon$$

in Case (a). We have seen examples of $p = 1$ and $q = 2, 3$ in the algorithms discussed above. The value of $\frac{s+p}{q-p}$ in these cases is given in Table 5.1 for 1, 2, and 3 spatial dimensions, to show how the estimates of optimal N vary with the number of dimensions.

Complex geometries. The analysis in this paper extends to topologically nonrectangular geometries, as illustrated by the L-shaped domain in Figure 5.1. From the description of wave-like behavior in the introduction, we have (possibly after a mapping) a homogeneous smallest-length scale all through the domain, and the highest

number of waves to be approximated on each subdomain is as before k/d in each direction. Therefore, the accuracy estimate (3.3) is still valid. The number of subdomains in this example is $3/4\ d^2$, and since our analysis is independent of the constant factor in the work functions, it also applies in this case.

Differentiation algorithms. Calculation of derivatives is carried out on most current computer architectures more efficiently by matrix multiplication than by the use of FFTs when the number of grid points in the differentiation direction is less than 32. Figures 4.1 and 4.2 and Table 5.1 show that the optimal N can be less than 32 for a wide range of accuracies and work functions not involving the logarithmic term, but higher values of N may be optimal when very high accuracy is required in two or three dimensions, leaving FFTs as the method of choice.

Total number of points per wavelength. In all the cases analyzed above, the total number of points per wavelength (M/k) approaches constant values when the accuracy is increased.

Optimal balancing of subdomains and points in each subdomain. The analysis shows that the optimal number of points in each subdomain is independent of the wavenumber k, and Figures 4.1 and 4.2 show that it depends almost linearly on the accuracy ε for a given work function $W(N, d)$. As the optimal total number of points per wavelength varies slowly only with ε, the optimal number of subdomains is approximately inversely proportional to ε.

6. Numerical results.

6.1. An approximation example. The Chebyshev collocation approximation to the function $\sin(2\pi^2 x)$ on $-1 \le x \le 1$ is calculated. Matrix multiplication is used to calculate Chebyshev coefficients, and the work function is approximately proportional to dN^2. However, actual timing of the calculations shows that the work is expressed more accurately by $Cd(N^2 + 4N)$, i.e., using the work function from Case (b), with $\beta = 4$.

Maximum approximation error is measured by evaluating the Chebyshev interpolant at a fixed set of equidistant points. The typical tendency is illustrated in Figure 6.1. Many subdomains are most efficient for low accuracy, while high accuracy requires fewer subdomains with more points in each. More detailed results are given in Table 6.1, where the most efficient choices of parameters are marked with boxes around N. For each d and E, the table entry is chosen as the N giving an error closest to E on a logarithmic scale. To find the "most efficient" of two pairs of (N, d) when one of them requires more calculation time but yields higher accuracy, we have considered straight lines between the data points closest to the required accuracy, as drawn in Figure 6.1.

The results in Table 6.1 can be compared with the theoretical optimal values, found by solving (4.2) or (4.4), with $r = 1$, $\alpha = 2$, and ε for the required accuracy to obtain N, and then by inserting this N and ε into (4.1) for the number of subdomains d. The theoretical optimal values of N and d for the three different accuracies are shown in Table 6.2, and the agreement is good for the work function of Case (b) with $\beta = 4$.

6.2. A PDE example. The PDE

$$u_t + u_x + u_y = 0, \qquad 0 < x < 2, \quad 0 < y < 2, \quad t > 0, \tag{6.1a}$$

with initial and boundary conditions

$$u(x, y, 0) = \sin(k\pi x + k\pi y), \qquad 0 < x < 2, \quad 0 < y < 2, \tag{6.1b}$$

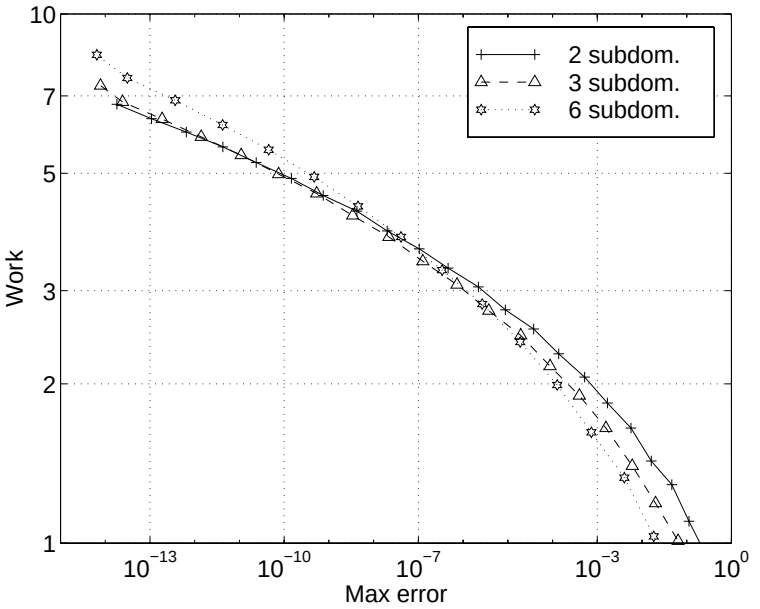


FIG. 6.1. *Approximation of the function* $\sin(2\pi^2 x)$. *Work (CPU-time) as a function of the max error for different number of subdomains.*

TABLE 6.1

Calculation of the numerical approximation with different accuracies to the function $\sin(2\pi^2 x)$, *using different domain decomposition parameters* (N, d). *The preferred parameter choices are indicated by boxes around the optimal values of* N.

	$E = 10^{-3}$			$E = 10^{-7}$			$E = 10^{-13}$		
d	Error	Time	N	Error	Time	N	Error	Time	N
1	$0.84 \cdot 10^{-3}$	2.64	28	$0.97 \cdot 10^{-7}$	4.23	36	$0.79 \cdot 10^{-13}$	6.81	46
2	$1.69 \cdot 10^{-3}$	1.84	16	$1.05 \cdot 10^{-7}$	3.60	23	$1.08 \cdot 10^{-13}$	6.34	31
3	$1.56 \cdot 10^{-3}$	1.65	12	$1.26 \cdot 10^{-7}$	3.41	18	$1.87 \cdot 10^{-13}$	6.33	25
4	$1.27 \cdot 10^{-3}$	1.58	10	$2.49 \cdot 10^{-7}$	3.26	15	$1.31 \cdot 10^{-13}$	6.65	22
5	$0.59 \cdot 10^{-3}$	1.68	9	$0.53 \cdot 10^{-7}$	3.62	14	$0.86 \cdot 10^{-13}$	6.96	20
6	$0.74 \cdot 10^{-3}$	1.62	8	$0.41 \cdot 10^{-7}$	3.79	13	$0.31 \cdot 10^{-13}$	7.57	19
8	$0.46 \cdot 10^{-3}$	1.76	7	$0.93 \cdot 10^{-7}$	3.77	11	$3.59 \cdot 10^{-13}$	7.36	16
10	$0.62 \cdot 10^{-3}$	1.72	6	$0.78 \cdot 10^{-7}$	4.01	10	$1.43 \cdot 10^{-13}$	8.18	15

TABLE 6.2

Optimal pairs of domain decomposition parameters (N, d) *for numerical approximation with different accuracies to the function* $\sin(2\pi^2 x)$, *predicted by the analysis from Case* (a) *and Case* (b) *of section* 4.

	$E = 10^{-3}$		$E = 10^{-7}$		$E = 10^{-13}$	
Work model	N	d	N	d	N	d
Case (a)	6	11	15	5	29	3
Case (b), $\beta = 4$	9	6	18	3	32	2

TABLE 6.3

Calculation of the numerical solution with different accuracies of the problem (6.1) *up to* $t = 2$, *using different domain decomposition parameters* (N, d). *The preferred parameter choices are indicated by boxes around the optimal values of* N.

	$E = 10^{-2}$			$E = 10^{-5}$			$E = 10^{-8}$		
d	Error	Time	N	Error	Time	N	Error	Time	N
1	$1.28 \cdot 10^{-2}$	298	43	$1.23 \cdot 10^{-5}$	681	51	$0.77 \cdot 10^{-8}$	1278	58
2	$1.24 \cdot 10^{-2}$	147	24	$0.63 \cdot 10^{-5}$	495	31	$0.87 \cdot 10^{-8}$	1012	36
3	$0.78 \cdot 10^{-2}$	130	18	$1.34 \cdot 10^{-5}$	406	23	$1.05 \cdot 10^{-8}$	1029	28
4	$1.62 \cdot 10^{-2}$	98	14	$1.78 \cdot 10^{-5}$	395	19	$0.50 \cdot 10^{-8}$	1178	24
5	$1.24 \cdot 10^{-2}$	97	12	$0.87 \cdot 10^{-5}$	462	17	$1.14 \cdot 10^{-8}$	1232	21
6	$0.69 \cdot 10^{-2}$	114	11	$1.58 \cdot 10^{-5}$	452	15	$1.13 \cdot 10^{-8}$	1337	19
7	$0.69 \cdot 10^{-2}$	120	10	$0.96 \cdot 10^{-5}$	524	14	$0.47 \cdot 10^{-8}$	1649	18
8	$1.06 \cdot 10^{-2}$	114	9	$0.96 \cdot 10^{-5}$	562	13	$2.57 \cdot 10^{-8}$	1437	16

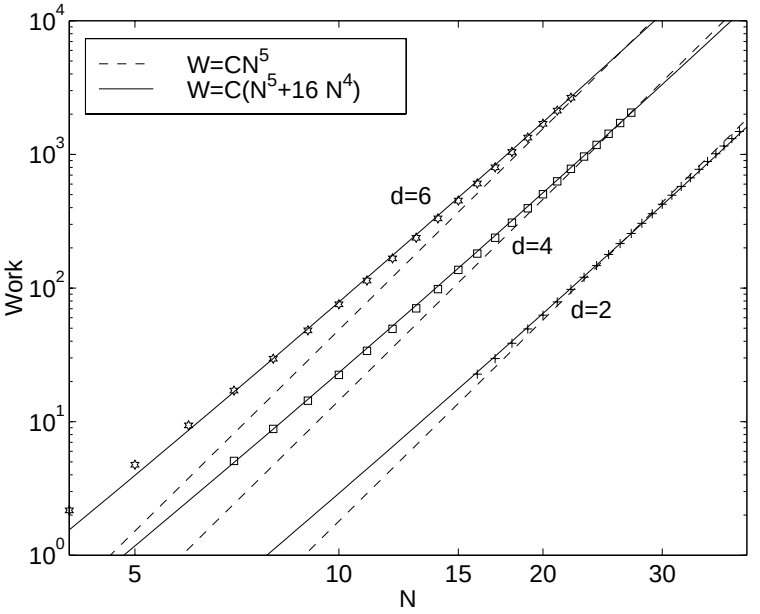


FIG. 6.2. *Measured (points) and modeled (curves) work (CPU-time) for the solution of the PDE in section* 6.2 *with different numbers of subdomains.*

$$u(0, y, t) = \sin\left(-k\pi t + k\pi(y - t)\right), \qquad 0 < y < 2, \quad t > 0, \tag{6.1c}$$

$$u(x, 0, t) = \sin\left(k\pi(x - t) - k\pi t\right), \qquad 0 < x < 2, \quad t > 0, \tag{6.1d}$$

with $k = 5\pi/\sqrt{2}$, is integrated to $t = 2$. The spatial discretization is Chebyshev collocation, while a high-order explicit time-integration scheme (Butcher's sixth-order Runge–Kutta method [9]) is used to eliminate time discretization errors. Timing and accuracy results are given in Table 6.3, where the most efficient choices of parameters are marked with boxes around N. Practically identical results are obtained with the initial function $u_0(x, y) = \sin(k\pi x) \cdot \sin(k\pi y)$, which does not represent a plane wave.

The time-steps are proportional to $1/dN^2$, and as the calculation of derivatives at each time-step involves matrix products of complexity $O(N^3)$ per subdomain, the

TABLE 6.4
Optimal pairs of domain decomposition parameters (N, d) for the numerical solution with different accuracies of the problem (6.1) up to $t = 2$, predicted by the analysis from Case (a) and Case (b) of section 4.

	$E = 10^{-2}$		$E = 10^{-5}$		$E = 10^{-8}$	
Work model	N	d	N	d	N	d
Case (a)	6	14	16	6	26	4
Case (b), $\beta = 16$	8	8	20	4	31	3

number of operations for integration up to a given time is $O(d^3N^5)$. As in the previous example, the actual CPU-time indicates that lower order terms should be included, as seen from Figure 6.2. Using (3.4b) from Case (b) with $\alpha = 4$ and $\beta = 16$ gives a better estimate of the computational work, and the corresponding predictions for N_{opt} and d_{opt} are given in Table 6.4. The predicted optimal values using Case (b) with $\beta = 16$ are close to the experimental results, as there are small differences in efficiency between the best choices of (N, d).

7. Relation to finite difference methods. In some of the cases considered in the previous sections both the theoretical analysis and the numerical experiments give quite small optimal numbers of points in each subdomain. The resulting methods are comparable to high-order finite difference methods, except that the grid points are not uniformly distributed. It is therefore natural to ask whether a finite difference method would be more efficient. Spectral methods are better when high accuracy is required, but in which range is this valid?

To indicate how the efficiency of finite difference methods and spectral methods compare, the first derivative of the function $\sin(\pi^2 x)$ is calculated on the interval $-1 \le x \le 1$. As examples of high-order finite difference schemes we have used the standard fourth-order (FD4) and sixth-order (FD6) centered schemes given, e.g., in [7]. A complication with finite difference schemes is that special operators that reduce the accuracy have to be constructed at and near the boundaries.

Figure 7.1 shows the efficiency and the grid point requirements for these finite difference methods compared with single-domain and multidomain Chebyshev collocation methods. When more than the lowest accuracies are required, the finite difference methods must be of very high order to compete with the spectral methods.

These results are based on derivative approximation and do not include the larger numerical dispersion and diffusion errors in finite difference methods for wave propagation problems. Conversely, the time-step restrictions for explicit time-integration are less severe for finite differences.

Figure 7.2 shows that for the solution of the advection problem

$$\begin{aligned} u_t + u_x &= 0, \qquad 0 < x < 2, \quad t > 0, \\ u(x,0) &= \sin(\pi^2 x), \qquad u(0,t) = -\sin(\pi^2 t), \end{aligned} \tag{7.1}$$

the fourth-order finite difference method is more efficient than the spectral method only for accuracy between two and three digits or less. The time-integration in this example is done by the standard fourth-order Runge–Kutta method [9], and the time-steps are chosen as the most efficient stable values for each method ($3/N$ for FD4 and $6/dN^2$ for the spectral methods).

8. Conclusions. We have presented a strategy for choosing the optimal number of subdomains and grid points in each subdomain in a Chebyshev domain decompo-

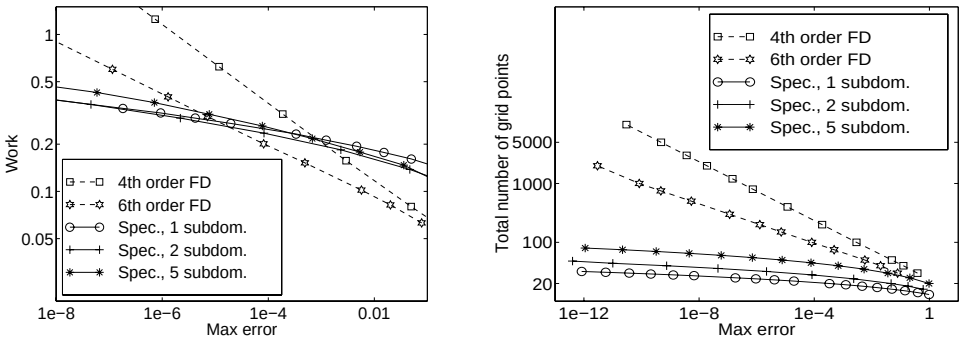


FIG. 7.1. *Approximation of the derivative of the function* $\sin(\pi^2 x)$ *by finite difference and spectral Chebyshev collocation methods. Left: Work (CPU-time) as a function of the max error. Right: The total number of grid points as a function of the max error.*

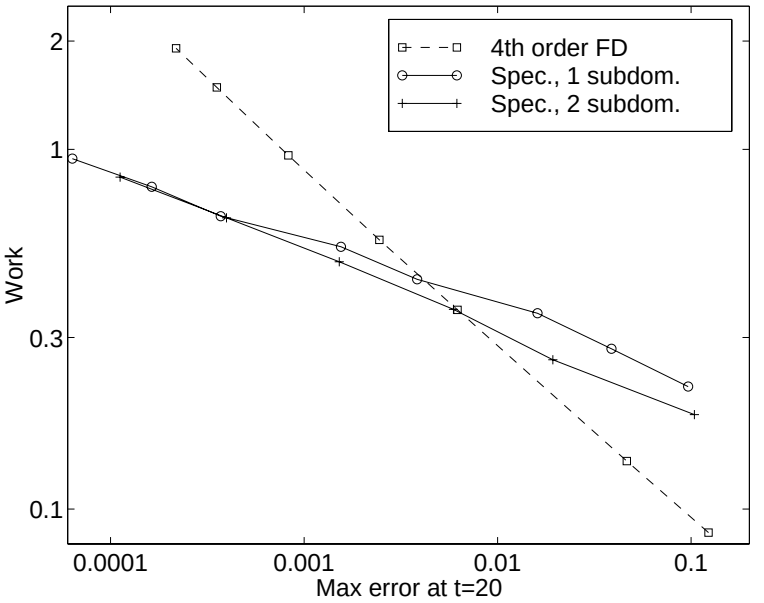


FIG. 7.2. *Measured work (CPU-time) as a function of accuracy for wave propagation by finite difference and spectral Chebyshev collocation methods.*

sition method. It is assumed that the solution is mainly homogeneous throughout the domain, such that a global smallest length scale can be decided a priori. This corresponds to the highest wavenumber to be resolved in the calculations.

The accuracy of the method is derived from properties of spectral approximation, and the computational work to obtain a required accuracy is minimized for work functions of different form, where the work grows faster with increased resolution within each subdomain than with increasing number of subdomains.

These considerations lead to a standard minimization problem, and the results can be summarized as follows:

- The optimal number of points in each subdomain, N_{opt}, is in the simplest case linearly proportional to the negative logarithm of the required accuracy. This optimal number N_{opt} is independent of the complexity (the number of waves) of the problem.
- The optimal number of subdomains increases with the complexity of the problem and is approximately inversely proportional to the negative logarithm of the required accuracy.
- The optimal total number of points per wavelength is slowly varying only with the required accuracy and approaches constant values for high accuracies.

Acknowledgments. The first author thanks Professor Knut S. Eckhoff for initiating the study of this problem and Professor David A. Kopriva for stimulating discussions.

REFERENCES

[1] M. Abramowitz and I. A. Stegun, *Handbook of Mathematical Functions*, Dover, New York, 1970.

[2] Ø. Andreassen, I. Lie, and C. E. Wasberg, *The spectral viscosity method applied to simulation of waves in a stratified atmosphere*, J. Comput. Phys., 110 (1994), pp. 257–273.

[3] C. Canuto, M. Y. Hussaini, A. Quarteroni, and T. A. Zang, *Spectral Methods in Fluid Dynamics*, Springer-Verlag, New York, 1988.

[4] W. S. Don and A. Solomonoff, *Accuracy and speed in computing the Chebyshev collocation derivative*, SIAM J. Sci. Comput., 16 (1995), pp. 1253–1268.

[5] P. Fischer and D. Gottlieb, *On the optimal number of subdomains for hyperbolic problems on parallel computers*, Int. J. Supercomput. Appl. High Perform. Comput., 11 (1997), pp. 65–76.

[6] P. F. Fischer and A. T. Patera, *Parallel spectral element solution of the Stokes problem*, J. Comput. Phys., 92 (1991), pp. 380–421.

[7] B. Fornberg, *Generation of finite difference formulas on arbitrary spaced grids*, Math. Comp., 51 (1988), pp. 699–706.

[8] D. Gottlieb and S. A. Orszag, *Numerical Analysis of Spectral Methods: Theory and Applications*, SIAM, Philadelphia, 1977.

[9] E. Hairer, S. P. Nørsett, and G. Wanner, *Solving Ordinary Differential Equations* I, *Nonstiff Problems*, Springer-Verlag, New York, 1987.

[10] J. S. Hesthaven, *A stable penalty method for the compressible Navier-Stokes equations:* II. *One-dimensional domain decomposition schemes*, SIAM J. Sci. Comput., 18 (1997), pp. 658–685.

[11] J. S. Hesthaven, *A stable penalty method for the compressible Navier-Stokes equations.* III. *Multidimensional domain decomposition schemes*, SIAM J. Sci. Comput., 20 (1999), pp. 62–93.

[12] D. A. Kopriva, *Computation of hyperbolic equations on complicated domains with patched and overset Chebyshev grids*, SIAM J. Sci. Statist. Comput., 10 (1989), pp. 120–132.

[13] D. A. Kopriva, *Multidomain spectral solution of the Euler gas-dynamics equations*, J. Comput. Phys., 96 (1991), pp. 428–450.

[14] D. A. Kopriva and J. H. Kolias, *A conservative staggered-grid Chebyshev multidomain method for compressible flows*, J. Comput. Phys., 125 (1996), pp. 244–261.

[15] E. M. Rønquist, *Optimal Spectral Element Methods for the Unsteady Three-Dimensional Incompressible Navier-Stokes Equations*, Ph.D. thesis, Massachusetts Institute of Technology, Cambridge, MA, 1988.

[16] A. T. Patera, *A spectral element method for fluid dynamics: Laminar flow in a channel expansion*, J. Comput. Phys., 54 (1984), pp. 468–488.

[17] A. Quarteroni, *Domain decomposition methods for systems of conservation laws: Spectral collocation approximations*, SIAM J. Sci. Statist. Comput., 11 (1990), pp. 1029–1052.

[18] A. Quarteroni, *Domain decomposition and parallel processing for the numerical solution of partial differential equations*, Surveys Math. Indust., 1 (1991), pp. 75–118.

[19] C. E. Wasberg, *On the Chebyshev Collocation Approximation and Domain Decomposition*, Tech. report FFI/RAPPORT-92/7032, Norwegian Defence Research Establishment, N-2027 Kjeller, Norway, 1992.

[20] C. E. Wasberg, *Spectral Collocation Methods with Domain Decomposition for Fluid Dynamics*, Doctoral thesis, Department of Mathematics, University of Bergen, Bergen, Norway 1995.

SIAM J. SCI. COMPUT.
Vol. 22, No. 2, pp. 633–655
© 2000 Society for Industrial and Applied Mathematics

HALF-SPACE ANALYSIS OF THE DEFECT-CORRECTION METHOD FOR FROMM DISCRETIZATION OF CONVECTION*

BORIS DISKIN† AND JAMES L. THOMAS‡

Abstract. A novel, comprehensive, discrete, half-space analysis for the defect-correction method has been developed. This analysis plays the same role for nonelliptic-problem solvers as the full-space Fourier mode analysis plays for elliptic-problem solvers. Numerical simulations confirm the accuracy of the half-space analysis. The following important findings about the defect-correction method applied to the Fromm discretization of the two-dimensional convection equation are reported:

1. The initial convergence rate of the defect-correction method is principally a function of the relative accuracy of the operators involved in the defect-correction iterations.
2. The asymptotic convergence rate is about 0.5 per defect-correction iteration.
3. If the driver operator is first-order accurate, then the initial convergence rates may be slow. The number of iterations required to get into the asymptotic convergence regime or/and to converge the algebraic error below the discretization-error level can be proportional to $h^{-1/3}$. This h-dependent delay is a multidimensional phenomenon—it cannot be observed in one-dimensional problems, and it disappears in the case of close alignment between the grid and the convection equation characteristic.
4. If the driver operator is second-order accurate, the defect-correction solver demonstrates the asymptotic convergence rate from the very beginning. Only one defect-correction iteration is required to converge algebraic error substantially below the discretization-error level.

Key words. half-space analysis, convection equation, Fromm discretization, defect-correction method

AMS subject classifications. 65M12, 65T50

PII. S1064827599358637

1. Introduction.

The subject of this paper is the analysis of the convergence properties of the defect-correction method for the Fromm discretization of the scalar constant-coefficient convection equation. This problem is rather simple, but past experience has shown that many important aspects relevant to more general problems can be well captured in such simple models. In the last decade, the defect-correction technique introduced in [2] has been used extensively in computational fluid dynamics (CFD), often in conjunction with multigrid, for convection-diffusion, Euler, and Navier–Stokes problems (see, e.g., [1, 7, 8, 11, 12, 13, 14, 15, 16, 17]). Usually, the efficiency was quite satisfactory. Sometimes, however, fast convergence was observed only after several slowly convergent initial iterations. The analysis of the defect-correction method reported in this paper was motivated by the search for an explanation of convergence properties of existing practical CFD solvers [13, 15].

Although the defect-correction method has been used to solve complicated nonlinear problems, the analytical tool usually applied to confirm experimental findings was the full-space Fourier mode analysis of a related constant-coefficient problem (see,

*Received by the editors July 14, 1999; accepted for publication (in revised form) March 29, 2000; published electronically August 9, 2000.

http://www.siam.org/journals/sisc/22-2/35863.html

†Institute for Computer Applications in Science and Engineering, NASA Langley Research Center, Mail Stop 132C, Hampton, VA 23681-2199 (bdiskin@icase.edu). This research was supported by the National Aeronautics and Space Administration under NASA contract NAS1-97046 while the author was in residence at the Institute for Computer Applications in Science and Engineering, NASA Langley Research Center.

‡Computational Modeling and Simulation Branch, NASA Langley Research Center, Mail Stop 128, Hampton, VA 23681-2199 (j.l.thomas@larc.nasa.gov).

e.g., [8, 14]). This approach is well justified for *elliptic* problems, where the full-space Fourier analysis is known to provide an accurate prediction of the asymptotic convergence rate. The asymptotic convergence rate is especially important in elliptic problems because it provides the upper bound for the convergence rates throughout the solution process.

In *nonelliptic* problems, a good asymptotic convergence rate is often observed in contrast to a pessimistic forecast provided by the modified (zero-mode exclusion) full-space Fourier mode analysis (see [8, 18]). The full-space analysis fails to predict the asymptotic convergence rate correctly because it does not take boundary conditions into account. For nonelliptic problems, accounting for the inflow boundary conditions becomes very important because solution values in the interior, far away from the boundary, are strongly affected by the boundary conditions.

Another problem associated with nonellipticity is the fact that convergence rates in several *initial* iterations might be quite slow. Note that the modified full-space Fourier mode analysis can still be used to estimate the slowest possible (but not asymptotic) convergence rate, while the asymptotic rate can be found by an eigenvalue matrix analysis (see [8]). However, none of these (full-space Fourier and eigenvalue) analyses can predict the *number* of slow initial iterations preceding establishment of the asymptotic convergence rate.

The analysis proposed in this paper for the defect-correction method is a half-space analysis. Generally, this half-space analysis plays the same role for nonelliptic-problem solvers as the full-space Fourier mode analysis plays for elliptic-problem solvers. Applications of this analysis are not limited to the defect-correction method. (See [5] for an application of the half-space analysis to a multigrid method.) This analysis considers difference operators on the half-space including the inflow boundary. The inflow boundary conditions are represented by one Fourier component at a time. In this way, the original multidimensional problem is translated into a one-dimensional (1D) discrete problem, where the frequency of the boundary Fourier component is considered as a parameter.

The half-space analysis is able to precisely explain many phenomena observed in solving nonelliptic equations and provides a close prediction of the actual solution behavior. It predicts the convergence rate for each iteration and the asymptotic convergence rate. It can be easily adjusted to analyze a global effect of *any* discretization of the inflow boundary conditions. If necessary, it can take into account the influence of the discretized outflow boundary conditions; this is important for applications of this analysis to convection-diffusion and strongly anisotropic problems.

Although we consider only a two-dimensional (2D) problem, the half-space analysis can be extended to three dimensions. In this case, a 2D Fourier component represents the inflow boundary conditions. The resulting 1D discrete problem includes two frequency parameters.

The material in the paper is presented in 10 sections. The model problem is formulated in section 2 and is followed by a brief explanation of the defect-correction procedure in section 3. In section 4, the notion of *penetration distance* is introduced and its role in defining approximation accuracy in discretized nonelliptic problems is discussed. The applicability, complexity, and interrelation of different types of analysis are discussed in section 5. A simplified, but very useful, version of the half-space analysis is presented and applied in section 6. In this version, the difference operators involved in defect-correction iterations are replaced with their first differential approximations. Such an analysis provides a qualitative estimate of the convergence for an

arbitrary defect-correction scheme. A novel, comprehensive, *discrete* half-space analysis for the defect-correction method with the first-order driver is introduced in section 7. Section 8 refers to a standard 1D matrix analysis for estimating the asymptotic convergence rate and the convergence rate upper bound for the 2D defect-correction iterations. Results of numerical tests with the defect-correction method employing the first-order driver for the Fromm scheme are collected in section 9. Section 10 presents numerical results for the defect-correction method with a second-order driver.

2. Model problem. The model problem studied in this paper is the 2D constant-coefficient convection equation

$$LU \equiv \Big(\bar{b}\cdot\nabla\Big)U = F(x,y), \tag{2.1}$$

where $\bar{b} = (b_1, b_2)$ is a given vector. Without loss of generality one can assume $b_1^2 + b_2^2 = 1$.

The solution $U(x,y)$ is a differentiable function defined on the layer $(x,y) \in [0,1] \times (-\infty, +\infty)$. In this paper, we deal mostly with the homogeneous equation $F(x,y) \equiv 0$. Exceptions when nonzero right-hand-side (source) functions $F(x,y)$ are considered will be emphasized specifically.

Let ϕ be the nonalignment angle, i.e., the angle between the vector $\bar{b}$ and the positive direction of the x axis; $t = \tan\phi = b_2/b_1$ is the nonalignment parameter. For simplicity, we assume $b_1 \geq b_2 \geq 0$ and, therefore, $1 \geq t \geq 0$. Then, (2.1) can be rewritten as

$$\partial_\xi U = F(x,y), \tag{2.2}$$

where $\xi = b_1 x + b_2 y$ is a variable along the characteristic of (2.1). For subsequent use, we also define the cross-characteristic variable ($\perp$ to the characteristic) $\eta = -b_2 x + b_1 y$.

The equation is supplied with Dirichlet boundary conditions at the inflow boundary $x = 0$:

$$U(0,y) = g(y), \tag{2.3}$$

where $g(y)$ is a given function.

This problem is discretized on the 2D Cartesian grid with uniform mesh size h in both x and y directions. Let u_{i_1,i_2} be a discrete approximation to the solution $U(x,y)$ at the point $(x,y) = (i_1 h, i_2 h)$. Then, the second-order accurate discretization corresponding to the Fromm scheme is defined as

$$\begin{aligned}
L^h u_{i_1,i_2} &\equiv \tfrac{1}{4h}\Big(b_1\Big(u_{i_1+1,i_2} + 3u_{i_1,i_2} - 5u_{i_1-1,i_2} + u_{i_1-2,i_2}\Big) \\
&\qquad + b_2\Big(u_{i_1,i_2+1} + 3u_{i_1,i_2} - 5u_{i_1,i_2-1} + u_{i_1,i_2-2}\Big)\Big) = f_{i_1,i_2}, \\
L^h u_{N,i_2} &\equiv \tfrac{1}{2h}\Big(b_1\Big(3u_{N,i_2} - 4u_{N-1,i_2} + u_{N-2,i_2}\Big) \\
&\qquad + b_2\Big(3u_{N,i_2} - 4u_{N,i_2-1} + u_{N,i_2-2}\Big)\Big) = f_{N,i_2}, \\
& i_1 = 1,2,\ldots,N-1, \qquad N = 1/h, \\
& u_{0,i_2} = g(i_2 h), \qquad u_{-1,i_2} = g'(i_2 h).
\end{aligned} \tag{2.4}$$

The discrete scheme (2.4) in the interior is upwind biased but not a pure upstream scheme, because the operator at the point (i_1, i_2) requires the solution values at the downstream points $(i_1 + 1, i_2)$ and $(i_1, i_2 + 1)$ are required. As shown in [8], this discretization is well suited for solution by the defect-correction method. The outflow boundary conditions at $i_1 = N$ are discretized by the second-order upwind scheme. The discretization of the right-hand-side function is $f_{i_1,i_2} = F(i_1 h, i_2 h)$. Function $g'(y)$ is an additional numerical boundary condition. In model problems where the exact solution $U(x, y)$ is known, one can define $g'(y) = U(-h, y)$. This (artificial) discretization of the inflow boundary conditions was chosen due to its simplicity; it provides the same discretization stencil for operator (2.4) at all inner grid points. Any other locally defined discrete boundary conditions (e.g., given values of u_{0,i_2} and the first-order upstream discretization at $i_1 = 1$) can be considered as well.

3. Defect-correction schemes. Let the target discrete problem be

$$L^h u_{i_1,i_2} = f_{i_1,i_2}, \tag{3.1}$$

where L^h is a target discretization of the convection operator (e.g., (2.4)) and let L^h_d be an easily solvable discretization of the same convection operator. One possible candidate is the first-order upwind discretization

$$\begin{aligned} L^h_d u_{i_1,i_2} &= \frac{1}{h}\Big(b_1\Big(u_{i_1,i_2} - u_{i_1-1,i_2}\Big) + b_2\Big(u_{i_1,i_2} - u_{i_1,i_2-1}\Big)\Big), \\ u_{0,i_2} &= g(i_2 h). \end{aligned} \tag{3.2}$$

This scheme is stable in downstream marching. In the case of a layer domain, the marching of this scheme requires an implicit line-by-line rather than a simple pointwise passage.

Let $\tilde{u}_{i_1,i_2}$ be the current solution approximation. Then the improved approximation $\bar{u}_{i_1,i_2}$ is calculated in the following two steps:

1. The correction v_{i_1,i_2} is calculated by marching operator L^h_d with a right-hand side represented by the residual of (3.1) computed for the current approximation $\tilde{u}_{i_1,i_2}$. The inflow boundary conditions for v are initialized with the zero values.

$$L^h_d v_{i_1,i_2} = f_{i_1,i_2} - L^h \tilde{u}_{i_1,i_2}. \tag{3.3}$$

2. The current approximation is corrected

$$\bar{u}_{i_1,i_2} = \tilde{u}_{i_1,i_2} + v_{i_1,i_2}. \tag{3.4}$$

The operator L^h_d is called the *driver* operator. If the iteration converges, steps (3.3) and (3.4) can be repeated until the desired accuracy is reached.

4. Penetration distance and discretization accuracy. In several papers (e.g., [8, 15]), authors studying the defect-correction method for nonelliptic problems observed a slow convergence or even a divergence in some common error norms for the initial iterations and good asymptotic convergence rates afterward. This behavior is different from that observed in solving elliptic problems by the defect-correction method, where the asymptotic convergence rate is the slowest one. The analysis in this paper shows that this nonelliptic feature is explained by some properties associated with the cross-characteristic interaction (e.g., dissipation and/or dispersion)

in the operators involved in the defect-correction iterations. Specifically, this cross-characteristic interaction and the frequency of an incoming component define the *penetration distance* (also termed "survival distance" in [6]) of this component. The penetration distance is the distance from the inflow boundary in which the discrete solution of the homogeneous problem reasonably approximates the continuous one (i.e., the discretization error is substantially smaller than the solution). The ratio of penetration distances of the operators L^h and L_d^h is an important factor in determining the number of defect-correction sweeps required to reach the asymptotic convergence regime.

Nonellipticity introduces a new issue into the standard discretization analysis common for elliptic operators. The discretization error of a discrete elliptic operator defined on a grid with mesh size h regarding a given component with frequency ω is defined by (1) the operator's approximation order p and (2) the component's normalized frequency ωh. For nonelliptic operators, the main factor is (3) the penetration distance d of this component, which is a function of p, ωh, and ω, namely,

$$d = d\Big(\omega(\omega h)^p\Big). \tag{4.1}$$

In homogeneous nonelliptic problems, the discrete operator does not yield a reasonable solution approximation beyond some $O(d)$ neighborhood of the inflow boundary. A discretization shows accuracy of order p (i.e., the discretization error is reduced by factor 2^p when the mesh size h is refined to $h/2$) if and only if the penetration distance of the incoming component is comparable with (or exceeds) the characteristic size of the domain. Of course, the penetration distance of a given incoming component increases on finer grids (i.e., on the grids with smaller mesh sizes). The grid on which the penetration distance approaches the characteristic size of the domain can be considered the coarsest grid for resolving this component.

In the solution of the homogeneous differential problem (2.1)–(2.3), all the incoming components are translated along the characteristics without changing their phases and amplitudes. However, on any grid which does not align with the characteristic, the discretization unavoidably introduces some dissipation and dispersion errors by cross-characteristic interaction. A quantitative measure of this numerical cross-characteristic interaction is the coefficient of the largest cross-characteristic derivative appearing in the first differential approximation (FDA) (see [19, 20]) to the operator under consideration. Briefly, the first differential approximation (also called modified equation) to a difference operator on a grid with mesh size h is the Taylor expansion of this difference operator truncated to the first items including the least nonzero power of h. The penetration distance is limited by the cross-characteristic interaction in the discrete scheme.

5. Different types of analysis: Applicability, complexity, and interrelation. In the defect-correction method solving elliptic problems, the main mechanism of convergence is damping of error components. In nonelliptic problems, there is another very important convergence mechanism: the downstream transport of the error components. In the presence of this additional mechanism, the accuracy is first achieved near the inflow boundary and then propagated into the interior of the domain. This downstream transport mechanism also explains the observed establishment of a good asymptotic convergence rate after slowly convergent initial iterations as follows: The components with the large amplification factors (which are responsible for the initial slowness) are eventually streamed out of the domain. The error

components that approximate the eigencomponents of the discrete target operator determine the asymptotic convergence. When the target discrete operator approximates the differential operator, these latter error components become characteristic components. In general, the *characteristic components* are components that are much smoother in the characteristic direction than in other directions.

The recognition of this additional convergence mechanism urges modifications in the standard analysis developed for elliptic problems. Basically, one can distinguish four types of analysis applied to nonelliptic problems: (1) a matrix analysis, (2) a modified zero-mode-exclusion full-space Fourier mode analysis, (3) a simplified half-space analysis of the FDA, and (4) the discrete half-space analysis proposed in this paper. The quality of an analysis applied to nonelliptic problems is determined by how well the analysis handles the characteristic components.

5.1. Matrix analysis. The most general and precise analysis is the matrix analysis. (See examples in [7, 8].) This analysis considers the difference operators without assumptions about the solution and boundary conditions. It can be applied to variable-coefficient problems as well. This analysis was found very useful for analyzing 1D problems. However, the enormous computational complexity of this analysis makes it not viable for multidimensional problems.

5.2. Modified full-space Fourier mode analysis. The modified full-space Fourier mode analysis is a modification of the standard full-space Fourier mode analysis excluding from the consideration all the characteristic modes. (See [18] and the bibliography therein.) It is the simplest and most popular type of analysis (e.g., see applications in [8, 14]). This analysis deals with the full-space Fourier components. It estimates only the amplification (damping) factor. Its inherent disadvantage is the inability to take the influence of the inflow boundary into account. This explains its failure in predicting the asymptotic convergence rate and describing the downstream transport phenomenon. However, initial slow convergence rates are often caused by noncharacteristic components for which the solution of the driver operator does not approximate well the target-operator solution. Therefore, the modified full-space analysis provides a good upper bound for the convergence rate. Note that it can also be useful for analyzing the effect of source functions.

5.3. FDA half-space analysis. The FDA half-space analysis is a relatively simple and efficient tool for analyzing the effect of the inflow boundary. (See examples of applications of this analysis in [3, 6, 10, 20] and also section 6 below.) The first differential approximations are considered on the half-space including the inflow boundary. The boundary conditions are represented by one Fourier mode at a time. The FDA analysis provides a good *qualitative* description of the downstream error transport phenomenon. It can also provide an estimate of the number of iterations required to get into the asymptotic convergence regime. This analysis focuses on characteristic components and, therefore, considers homogeneous problems. Note that a combination of the FDA analysis with the modified full-space analysis can provide a good insight for nonhomogeneous problems as well. The disadvantages of this analysis are the inability to provide quantitative estimates, to analyze the effect of different boundary condition discretizations, and to address the asymptotic convergence rate.

5.4. Discrete half-space analysis. The discrete half-space analysis considers the discretizations in their exact form rather than their differential approximation, while the boundary data are represented by a Fourier component. This analysis translates the original multidimensional problem into a 1D discrete problem, where

the frequency of the boundary Fourier component is considered as a parameter. To regularize the half-space problem, the solution is not allowed to grow faster than a polynomial function. This tool is very accurate (and cumbersome at the same time). It can be used to explain in detail many phenomena observed in solving nonelliptic equations and provides a close prediction of the actual solution behavior.

The solution obtained in the discrete half-space analysis has two different representation forms: (1) away from the boundary, the solution is defined as a linear combination of a finite number of analytical components; this region is called the *analytical representation region*; (2) in the region adjacent to the inflow boundary, the solution is defined pointwise; this zone is referred to as the *pointwise representation region*. In each defect-correction iteration, the pointwise representation region penetrates by one mesh size into the interior. By using these representations, the computational complexity of the analysis becomes much less than that associated with the 1D matrix analysis. In the asymptotic regime, when the pointwise representation zone covers all the domain, this analysis becomes a discrete 1D matrix analysis of the multidimensional problem.

The discrete half-space analysis provides a *quantitative* description of the approximate solution obtained at any stage of the defect correction solver. It predicts the convergence rate for each iteration and the asymptotic convergence rate. It can be easily adjusted to analyze a global effect of *any* local discretization of the inflow boundary conditions. This adjustment can be done just by widening the initial pointwise representation region at the inflow boundary. If necessary, the analysis can take into account the influence of the discretized outflow boundary conditions; in this case, it is exact for the constant-coefficient problems on a layer. Generally, this discrete half-space analysis treats completely both mechanisms of convergence, damping and downstream transport of errors, associated with nonelliptic problem solvers. It plays the same role for nonelliptic-problem solvers as the full-space Fourier mode analysis plays for elliptic-problem solvers.

6. Half-space analysis of first differential approximations. In this section, the FDA half-space analysis is applied to provide a qualitative description of penetration distances of the operators involved in defect-correction iterations. This analysis shows that if the operators L^h and L_d^h have different approximation orders, then efficiency of the defect-correction method is actually grid dependent. Grid dependence means that the maximal number of sweeps which might be required to reach the asymptotic convergence rate (or to reduce the algebraic error to the discretization-error level) on fine grids is larger than on coarse grids or vice versa. This phenomenon relates to the fact that some *characteristic* components are poorly approximated by the driver operator. In other cases, when the operators L^h and L_d^h have the same approximation order (see section 10), efficiency of the defect-correction method is optimal and grid independent.

Let L^h be an accurate discretization with respect to a particular incoming component. To approximate the solution of the operator L^h by solving some less accurate operator L_d^h (with a correspondingly shorter penetration distance), one has to iterate L_d^h as many times as needed to attain accuracy up to the L^h penetration distance. The required number of iterations depends on N, where N is the number of grid points in the characteristic direction. To be precise, the number of iterations is proportional to $N^{\frac{p-r}{p+1}}$, where p and r are the approximation orders of operators L^h and L_d^h, respectively. Below we derive the predicted dependence for a particular case where $p = 2$ and $r = 1$.

The target operator L^h approximates the differential operator L from (2.1) with second-order accuracy; therefore

$$\mathrm{FDA}\Big(L^h\Big) = \partial_\xi - C_2 h^2 \Big(\partial_{\eta\eta\eta} + \partial_\xi B^2(\partial_\xi, \partial_\eta)\Big),$$

where η is defined in section 2, $B^2(\partial_\xi, \partial_\eta)$ is a linear combination of second-order derivatives with respect to ξ and η, and C_2 is a constant. For a characteristic component u (in terms of which $\partial_\eta u \gg \partial_\xi u$), this approximation is simplified to

$$\mathrm{FDA}\Big(L^h\Big) \approx \partial_\xi - C_2 h^2 \partial_{\eta\eta\eta}. \tag{6.1}$$

Let the driver operator L_d^h have the first-order approximation accuracy. Then its FDA taken for the characteristic components is

$$\mathrm{FDA}\Big(L_d^h\Big) \approx \partial_\xi - C_1 h \partial_{\eta\eta}, \tag{6.2}$$

where, for a stable scheme, C_1 is a *positive* constant.

As in [3] and [6], the half-space analysis for the FDAs to the operators L^h and L_d^h presented in this section is focused on approximating the characteristic components. The analysis considers the discretizations of the homogeneous equation (2.1) on the half space $\{(x, y) : x \geq 0,\ -\infty < y < \infty\}$ with boundary conditions (at $x = 0$) being represented by the Fourier mode $e^{i\omega y}$. The purpose of this analysis is to estimate the penetration distance as a function of the frequency ω of the incoming Fourier component and the mesh size h.

For the driver operator, we seek a bounded differentiable function $\phi(x, y)$ satisfying the following equation and boundary condition:

$$\partial_\xi \phi - C_1 h \partial_{\eta\eta} \phi = 0, \qquad \phi(0, y) = e^{i\omega y}. \tag{6.3}$$

The exact solution to (6.3) can be written as

$$\phi = e^{C_1 h \beta^2 \xi + \beta \eta},$$

where $\beta = a + ib$ is a complex number with a and b satisfying the system of the algebraic equations

$$\begin{cases} C_1 h(a^2 - b^2)t + a = 0, \\ 2C_1 h a b t + b = \omega\sqrt{1 + t^2}. \end{cases}$$

From the system, $a = O(h)$ and $b = \omega\sqrt{1 + t^2} + O(h^2)$. Therefore, the leading term of the bounded solution to (6.3) is

$$\phi \sim e^{-C_1 h(1+t^2)\omega^2 \xi + i\omega\eta\sqrt{1+t^2}}. \tag{6.4}$$

The factor $e^{\sqrt{1+t^2}(i\omega\eta)}$ represents the exact solution of the continuous problem, while the factor $e^{-C_1 h(1+t^2)\omega^2 \xi}$ is the influence of the numerical cross-characteristic interaction. In the case of the first-order driver, this is a dissipation which damps the amplitude. From (6.4), one can conclude that the penetration distance along the characteristic direction ξ at which this damping becomes $O(1)$ is proportional to $r^{(1)} = \frac{1}{\omega(\omega h)}$. In a similar way, one can derive the penetration distance of a second-order scheme, which is proportional to $r^{(2)} = \frac{1}{\omega(\omega h)^2}$. For even-order schemes, the

numerical cross-characteristic interaction usually affects the *phase* of the incoming component rather than the amplitude.

If the penetration distances $r^{(1)}$ and $r^{(2)}$ exceed the characteristic size of the domain, then fast initial convergence in the defect-correction method is expected because any errors including the errors at the region adjacent to the inflow boundary are swept out of the domain—to first-order accuracy on the first sweep and to the second-order accuracy on the second sweep. On the other hand, if $r^{(1)}$ does not exceed the characteristic size of the domain, errors adjacent to the inflow boundary are swept downstream only a fraction of the domain size. Multiple sweeps are required to sweep errors out of the domain, as

$$N_{\text{sweeps}} \sim \frac{R}{r^{(1)}}, \tag{6.5}$$

where R is a characteristic size of the domain ($R = \sqrt{1+t^2}$ in our problem). This consideration implies that in the case, when the second-order accuracy of the target operator is just attained, $r^{(2)} \approx R$, the number of iterations is estimated as

$$N_{\text{sweeps}} \sim \begin{cases} (R/h)^{\frac{1}{3}} & \text{for} \quad \omega = O\Big(R^{-\frac{1}{3}} h^{-\frac{2}{3}}\Big), \\ (R\omega)^{\frac{1}{2}} & \text{for} \quad h = O\Big(R^{-\frac{1}{2}} \omega^{-\frac{3}{2}}\Big). \end{cases} \tag{6.6}$$

This h dependence was first mentioned in [4]. In many practical calculations, it can hardly be noticed. Note that this h-dependence disappears in the case of close alignment ($t \approx 0$) because the coefficients C_1 (6.2) and C_2 (6.1) are proportional to t (see examples in section 9.1). However, a careful choice of data in the numerical experiments allows us to observe this behavior (section 9). This h dependence is most prominent in the problems associated with boundary information propagating to great distances—aeroacoustics or electromagnetics, for example.

7. Discrete half-space analysis for defect-correction method with first-order driver. This section presents the discrete half-space analysis for the defect-correction method solving the Fromm discretization (2.4) with the first-order driver (DC1) (3.2). The goal of this analysis is the comparison of (1) the exact solution of the differential problem, (2) the exact solution of the discrete problem, and (3) the approximate solutions at different stages of the defect-correction solver. All the results reported in sections 9.2, 9.3, and 10 below have been obtained by means of this analysis.

7.1. Exact solutions and discretization error. Let the exact solution of the problem (2.2) and (2.3) have the form $U(x,y) = e^{i(\omega_1 x + \omega_2 y)}$; then the differential problem can be rewritten as

$$\partial_\xi U(x,y) = i\beta_\xi e^{i(\omega_1 x + \omega_2 y)}, \qquad U(0,y) = e^{i\omega_2 y},$$

where $\beta_\xi = b_1\omega_1 + b_2\omega_2$ is the characteristic frequency ($\beta_\xi \approx 0$ for characteristic components).

The discrete counterpart is

$$L^h u_{i_1,i_2} = i\beta_\xi e^{i(\Omega_1 i_1 + \Omega_2 i_2)}, \qquad u_{0,i_2} = e^{i\Omega_2 i_2}, \qquad u_{-1,i_2} = e^{i(-\Omega_1 + \Omega_2 i_2)}, \tag{7.1}$$

where L^h is the target discrete operator and $\Omega_1 = \omega_1 h$ and $\Omega_2 = \omega_2 h$ are normalized frequencies.

We seek a solution of the discrete problem in the form

$$u_{i_1,i_2} = \phi_{i_1} e^{i\Omega_2 i_2}. \tag{7.2}$$

Then the problem (7.1) can be reformulated for ϕ_{i_1} as

$$a_{-2}\phi_{i_1-2} + a_{-1}\phi_{i_1-1} + a_0\phi_{i_1} + a_1\phi_{i_1+1} = \sqrt{1+t^2}hi\beta_\xi e^{i\Omega_1 i_1}, \tag{7.3}$$

$$\phi_0 = 1, \qquad \phi_{-1} = e^{-i\Omega_1}, \tag{7.4}$$

where

$$a_{-2} = \tfrac{1}{4},\ a_{-1} = -\tfrac{5}{4},\ a_0 = \tfrac{3}{4} + \tfrac{t}{4}\Big(e^{-i2\Omega_2} - 5e^{-i\Omega_2} + 3 + e^{i\Omega_2}\Big),\ a_1 = \tfrac{1}{4}. \tag{7.5}$$

The solution to (7.3) and (7.4) is given by

$$\phi_{i_1} = W_0 e^{i\Omega_1 i_1} + \Big(1 - W_0\Big)\Big(C_0 r_0^{i_1} + C_1 r_1^{i_1}\Big), \tag{7.6}$$

where r_0 and r_1 are the roots of the cubic equation

$$a_{-2} + a_{-1}r + a_0 r^2 + a_1 r^3 = 0,$$

satisfying to $|r| \le 1$,

$$W_0 = \frac{\sqrt{1+t^2}hi\beta_\xi}{a_{-2}e^{-i2\Omega_1} + a_{-1}e^{-i\Omega_1} + a_0 + a_1 e^{i\Omega_1}},$$

$$C_0 = \frac{r_0\Big(r_1 - e^{i\Omega_1}\Big)}{e^{i\Omega_1}\Big(r_1 - r_0\Big)}, \qquad C_1 = \frac{r_1\Big(r_0 - e^{i\Omega_1}\Big)}{e^{i\Omega_1}\Big(r_0 - r_1\Big)}.$$

We avoid here considering in detail the exceptional cases where either the denominator in the expression for W_0 turns out to be zero or $r_0 = r_1$. In these cases, the form of the solution (7.6) remains the same, while W_0 and/or $C_j (j = 0, 1)$ might become some linear functions of i_1.

Thus, the discretization error is calculated as

$$\begin{aligned} U\Big(i_1 h, i_2 h\Big) - u_{i_1,i_2} &= \Big[e^{i\Omega_1 i_1} - \phi_{i_1}\Big] e^{i\Omega_2 i_2} \\ &= \Big(1 - W_0\Big)\Big[e^{i\Omega_1 i_1} - C_0 r_0^{i_1} - C_1 r_1^{i_1}\Big] e^{i\Omega_2 i_2}. \end{aligned} \tag{7.7}$$

7.2. DC1 iteration. Let the boundary data be represented by a discrete Fourier mode $e^{i\Omega_2 i_2}$. Recall that the solution obtained in the discrete half-space analysis has two different representation forms: (1) Away from the boundary, the solution is defined as a collection of a finite number of analytical components; this region is called the *analytical representation region*. (2) In the region adjacent to the inflow boundary, the solution is defined pointwise; this zone is referred to as the *pointwise representation region*. The analysis is most efficient in problems where the initial analytical representation region covers most of the domain.

7.2.1. Analytical representation region. In the analytical representation region, the approximate solution to (7.1) is defined as a linear combination of analytical components; each component has the general form

$$P(i_1)q^{i_1}e^{i\Omega_2 i_2}, \tag{7.8}$$

where $P(i_1)$ is a complex-coefficient polynomial of i_1 and q ($|q| \le 1$) is the base of the given component. The polynomial part $P(i_1)$ will be referred to as the *amplitude* of the component $q^{i_1}e^{i\Omega_2 i_2}$. The initial approximation and the source function are assumed to be a finite collection of analytical components in the form (7.8). Many reasonable initial approximations satisfy this assumption: (1) zero approximation has no analytical components; (2) solution of the driver equation provides one analytical component; (3) the solution interpolated from a coarse grid in the framework of a two-grid solver is represented by several analytical components. The analytical representation at any stage of the defect-correction solver contains this finite collection plus the eigencomponent of the driver operator. To define this eigencomponent, the action of the driver operator on the component $q^{i_1}e^{i\Omega_2 i_2}$ is considered.

$$L_d^h\Big(q^{i_1}e^{i\Omega_2 i_2}\Big) \equiv \frac{d_{-1}q^{i_1-1} + d_0 q^{i_1}}{h\sqrt{1+t^2}}e^{i\Omega_2 i_2}, \tag{7.9}$$

where

$$d_{-1} = -1, \qquad d_0 = 1 + t\Big(1 - e^{-i\Omega_2}\Big). \tag{7.10}$$

Then the *driver-operator eigencomponent* which solves the equation

$$L_d^h\Big(q^{i_1}e^{i\Omega_2 i_2}\Big) = 0 \tag{7.11}$$

is the component with the base $q_d = -d_{-1}/d_0$.

7.2.2. Pointwise representation region. In most cases, the interior analytical approximation does not satisfy the discrete inflow boundary conditions, and an adjustment is needed in the neighborhood of the inflow boundary. This adjustment is given by an additional *pointwise* component

$$\begin{cases} B_{i_1}e^{i\Omega_2 i_2}, & 0 < i_1 \le N_0, \\ 0 & \text{otherwise}, \end{cases}$$

where B_{i_1} is a complex-valued vector of the length N_0. The segment $0 < i_1 \le N_0$ is the pointwise representation region and the vector B_{i_1} is called *the pointwise amplitude*. The role of the pointwise component is to correct the analytic representation in the pointwise region so that the combined approximate solution satisfies the inflow boundary conditions. In each DC1 iteration, N_0 increases by 1. (The size of the increment is determined by the length of the upwind part in the driver-operator discretization stencil.) The inflow boundary condition implementation used in the model problem (2.4) implies $N_0 = 0$. Analyzing more practical boundary conditions employing a modified stencil near the inflow boundary would be similarly done just requiring initial $N_0 > 0$.

7.2.3. Iteration. At any stage of the defect-correction iteration the solution approximation can be represented as

$$\tilde{u}_{i_1,i_2} = Q(i_1)e^{i\Omega_2 i_2}, \qquad Q(i_1) = \begin{cases} \sum\limits_j P_j(i_1)q_j^{i_1} + B_{i_1}, & 1 \le i_1 \le N_0, \\ \sum\limits_j P_j(i_1)q_j^{i_1}, & N_0 < i_1, \end{cases} \tag{7.12}$$

where $P_j(i_1)q_j^{i_1}e^{i\Omega_2 i_2}$ are the analytical components and $B_{i_1}e^{i\Omega_2 i_2}$ is the pointwise component.

The DC1 iteration consists of the two main computational blocks: (1) calculating residuals of the target operator (7.1) and (2) computing corrections by solving the driver operator with the right-hand-side function given by the target-operator residuals. The response in a DC1 iteration of each computational block to an input analytical component $v_{i_1,i_2} = P(i_1)q^{i_1}e^{i\Omega_2 i_2}$ and to the pointwise component is analyzed below. An example of the DC1 iteration analysis for the homogeneous equation (7.1) ($\beta_\xi = 0$) and the zero initial approximation can be found in [10].

Computing residual for analytical component. The residual function calculated for v_{i_1,i_2} is the same component $q^{i_1}e^{i\Omega_2 i_2}$, of course, with different amplitude $R(i_1)$. The amplitude $R(i_1)$ of the residual of v_{i_1,i_2} is calculated as

$$R(i_1) = \Lambda - \tfrac{1}{h\sqrt{1+t^2}}\Big[\, a_{-2}q^{-2}P(i_1-2) + a_{-1}q^{-1}P(i_1-1) + a_0P(i_1) + a_1qP(i_1+1)\Big], \tag{7.13}$$

where if $q^{i_1}e^{i\Omega_2 i_2}$ is a right-hand-side component of (7.1) ($q = e^{i\Omega_1}$), then $\Lambda = i\beta_\xi$; otherwise $\Lambda = 0$. Coefficients $a_j (j = -2, -1, 0, 1)$ are defined in (7.5).

Computing residual for pointwise component. Recall that the pointwise component corrects the analytical representation in the pointwise representation region. So the values B_{-1} and B_0 are given by the difference between boundary conditions (7.4) and the values calculated from analytical representation at points $i_1 = -1$ and $i_1 = 0$. Let N_0 be the current inner boundary for the pointwise representation region ($B_{i_1} = 0$ for $i_1 > N_0$) and an extended inner boundary be $N_0' = N_0 + 2$. Then the pointwise residual function is computed in the following way:

$$R_{i_1}^{\mathrm{pt}} = -\frac{1}{h\sqrt{1+t^2}} \begin{cases} a_{-2}B_{i_1-2} + a_{-1}B_{i_1-1} + a_0B_{i_1} + a_1B_{i_1+1}, & 1 \le i_1 \le N_0' \\ 0 & \text{otherwise.} \end{cases}$$

Correction to analytical component. Correction to the analytical component is a combination of the input component $C(i_1)q^{i_1}e^{i\Omega_2 i_2}$ and the driver-operator eigencomponent. The amplitude $C(i_1)$ of the correction to the analytical component v_{i_1,i_2} is calculated using the equation

$$d_{-1}q^{-1}C(i_1-1) + d_0C(i_1) = h\sqrt{1+t^2}R(i_1) \tag{7.14}$$

derived from (3.3), where the coefficients d_{-1} and d_0 are defined in (7.10). If v_{i_1,i_2} is not an eigencomponent for the driver operator ($q \ne q_d = -d_{-1}/d_0$), then the power of the polynomial $C(i_1)$ is the same as the power of the polynomial $R(i_1)$; otherwise the power of $C(i_1)$ is higher.

To satisfy the zero inflow boundary condition at $i_1 = 0$ that accompanies the correction equation, one must complete the correction with the driver-operator eigencomponent $D_0 q_d^{i_1} e^{i\Omega_2 i_2}$ with the amplitude

$$D_0 = -C(0).$$

Correction to pointwise component. The correction $C_{i_1}^{\mathrm{pt}}$ to the pointwise component is calculated from the following system of linear equations:

$$\begin{cases} C_{N_0'}^{\mathrm{pt}} = 0, \\ d_{-1} C_{i_1}^{\mathrm{pt}} + d_0 C_{i_1+1}^{\mathrm{pt}} = h\sqrt{1+t^2} R_{i_1+1}^{\mathrm{pt}}, \qquad 0 < i_1 < N_0'. \end{cases} \tag{7.15}$$

The amplitude D_1 of the accompanying driver-operator eigencomponent is computed as

$$D_1 = -\frac{h\sqrt{1+t^2} R_1^{\mathrm{pt}} - d_0 C_1^{\mathrm{pt}}}{d_{-1}}.$$

New amplitudes for analytical component, pointwise component, and driver-operator eigencomponent. The new amplitude $\tilde{P}(i_1)$ of the analytical component v_{i_1,i_2} is calculated as

$$\tilde{P}(i_1) = P(i_1) + C(i_1).$$

The corrected pointwise amplitude $\tilde{B}_{i_1}$ and the new boundary $\tilde{N}_0$ of the pointwise representation region are

$$\tilde{B}_{i_1} = B_{i_1} + C_{i_1}^{\mathrm{pt}}, \qquad \tilde{N}_0 = N_0 + 1.$$

The amplitude $D(i_1)$ of the driver-operator eigencomponent is also changed to $\tilde{D}(i_1)$ as follows:

$$\tilde{D}(i_1) = D(i_1) + D_0 + D_1.$$

7.3. Discretization of outflow boundary conditions. The discretization of the outflow boundary conditions can be taken into account as well. This is important for analyzing convection-diffusion and strongly anisotropic problems where resolution of the outflow boundary layer is of high priority. Discretized outflow boundary conditions usually imply some special discretization stencil different from that in the interior. To incorporate this feature into the analysis one can simply introduce another pointwise representation zone near the outflow boundary. In other words, an additional pointwise component makes the half-space analysis absolutely precise for all the constant-coefficient problems on a layer. This modified analysis was used in numerical experiments (sections 9.2, 9.3, and 10). The difference between the basic version (without taking outflow boundary conditions into account) and the modified version was extremely small, dropping below the computer round-off error on grids with $h \le 2^{-8}$. In the discretization accuracy tests (section 9.1), the influence of the discretized outflow boundary conditions was ignored.

7.4. Algebraic error. The *algebraic error* function, which is the difference between the exact (u_{i_1,i_2}) and approximate $(\tilde{u}_{i_1,i_2})$ solutions of the discrete target problem, is given by

$$\tilde{u}_{i_1,i_2} - u_{i_1,i_2} = \Big[Q(i_1) - \Big(W_0 e^{i\Omega_1 i_1} + (1 - W_0)(C_0 r_0^{i_1} + C_1 r_1^{i_1}) \Big) \Big] e^{i\Omega_2 i_2}, \tag{7.16}$$

where $\tilde{u}_{i_1,i_2}$ and u_{i_1,i_2} are defined in (7.12) and (7.2), respectively.

8. 1D matrix analysis. The 1D matrix analysis, applied to the problem (7.3)–(7.4), vastly simplifies analyzing the original multidimensional problem (2.4). This 1D matrix analysis predicts well both the asymptotic and the worst possible convergence rates of the multidimensional defect-correction solver. An $N\times N$ residual amplification matrix, $\mathbf{G_r}$, of the defect-correction iteration is constructed,

$$\bar{\mathbf{r}}^{\text{new}} = \mathbf{G_r}\bar{\mathbf{r}}^{\text{old}},$$

where the N-dimensional vectors $\bar{\mathbf{r}}^{\text{old}}$ and $\bar{\mathbf{r}}^{\text{new}}$ are residuals of (7.3) before and after the iteration, respectively.

The amplification of the residual can be bounded either by the spectral radius of $\mathbf{G_r}$ ($\rho(\mathbf{G_r})$) or by the L_2-norm of the matrix $\mathbf{G_r}$ ($||\mathbf{G_r}||_2 = \sqrt{\rho(\mathbf{G_r^*}\mathbf{G_r})}$). The spectral radius, $\rho(\mathbf{G_r})$, is usually associated with the asymptotic convergence rate, i.e., the rate corresponding to a large number of iterations. The L_2-norm ($||\mathbf{G_r}||_2$) indicates the "worst" possible convergence rate.

9. Numerical tests.

9.1. Verification of analytical predictions: Discretization accuracy test. A discretization is considered to have ϵ accuracy on a grid with mesh size h with respect to a given solution $U(x,y)$ of the differential problem if the following inequality holds

$$\frac{\|u_{i_1,i_2} - U(i_1h, i_2h)\|}{\|U\|} \le \epsilon, \tag{9.1}$$

where u_{i_1,i_2} is the exact discrete solution.

For characteristic components, the discretization accuracy is related with the penetration distance of this component in the discrete solution. Let the exact solution $U(x,y) = e^{i(\omega_1 x+\omega_2 y)}$ of the differential problem (2.2), (2.3) be a characteristic component, i.e., $\omega_1 + t\omega_2 \approx 0$. Below, the penetration distances of this component in the operators involved in the DC1 iterations are approximately calculated from the asymptotic solution of the half-space problem associated with each operators' FDA.

The FDA to the target second-order-accurate discrete operator (2.4) is given by

$$\text{FDA}\Big(L^h\Big) = b_1\partial_x + b_2\partial_y - \frac{h^2}{12}\Big(b_1\partial_{xxx} + b_2\partial_{yyy}\Big).$$

For characteristic components, it becomes

$$\text{FDA}\Big(L^h\Big) = \partial_\xi - \frac{h^2}{12\Big(1+t^2\Big)^2}\Big(-t^3+t\Big)\partial_{\eta\eta\eta}, \tag{9.2}$$

where the nonalignment parameter t and the characteristic variables ξ and η have been defined in section 2. Notice that when $t \approx 1$ the coefficient of the third derivative with respect to η vanishes in the FDA and the next term (the fourth derivative) becomes important.

The asymptotic solution $u(\xi,\eta)$ of the problem

$$\partial_\xi u - \frac{h^2}{12\Big(1+t^2\Big)^2}\Big(-t^3+t\Big)\partial_{\eta\eta\eta}u = 0, \qquad u\Big|_{x=0} = e^{i\omega_2 y}, \tag{9.3}$$

can be found as

$$u(\xi,\eta) = e^{\frac{\xi}{r_2}+i\omega_2\sqrt{1+t^2}\eta}, \tag{9.4}$$

where

$$r_2 = \frac{-i12\sqrt{1+t^2}}{\omega_2\Big(\omega_2 h\Big)^2\Big(-t^3+t\Big)} \tag{9.5}$$

and $|r_2|$ is the normalized penetration distance for the operator L^h in the characteristic (ξ) direction. Then the discretization error of L^h, defined as $\mathrm{DE}(L^h)$, can be estimated as

$$\mathrm{DE}(L^h) = e^{i(\omega_1 x+\omega_2 y)}\Big(1-e^{-\frac{\xi}{r_2}}\Big).$$

Thus, this discretization is estimated to have the accuracy ϵ for $U(x,y)$ on the distance δ (measured along the characteristic) from the boundary if the following inequality holds:

$$\Big|1-e^{-\frac{\xi}{r_2}}\Big| \le \epsilon \qquad \text{for} \quad \xi \le \delta. \tag{9.6}$$

Note, that ϵ defines the *relative* error; hence, $0 \le \epsilon \le 1$, and $\epsilon \approx 1$ indicates very poor accuracy. For operator (2.4), the penetration distance of the ϵ-accuracy is estimated from (9.6) as

$$\delta_2 = |r_2| \arccos(1-\epsilon^2/2). \tag{9.7}$$

The first differential approximation to the first-order-accurate driver operator (3.2) taken for the characteristic components is given by

$$\mathrm{FDA}\Big(L_d^h\Big) = \partial_\xi - \frac{h}{2\Big(1+t^2\Big)^{3/2}}\Big(t^2+t\Big)\partial_{\eta\eta}.$$

The normalized penetration distance in the ξ direction is

$$r_1 = \frac{2\sqrt{1+t^2}}{\omega_2\Big(\omega_2 h\Big)\Big(t^2+t\Big)} \tag{9.8}$$

and the ϵ-accuracy penetration distance is

$$\delta_1 = -r_1 \ln(1-\epsilon). \tag{9.9}$$

We perform the first test to validate the discrete half-space analysis without incorporating outflow boundary conditions from section 7 and analytical expressions (9.7) and (9.9) for penetration distances of characteristic components. We calculate penetration distances of $\epsilon = 0.01$ accuracy for different values of t and ω_2 for the target and driver operators (2.4) and (3.2). Our aim is to compare the following distances from the boundary (measured in mesh sizes along the x direction):

- the distances calculated by formulas (9.7) and (9.9) ($\delta_a^{(j)} = \delta_j/\big(h\sqrt{1+t^2}\big)$, $j = 1, 2$);

TABLE 9.1
Penetration distances (in mesh sizes) of the 1% accuracy.

t	ω_2	1%-Accuracy penetration distances					
		Target operator			Driver operator		
		$\delta_a^{(2)}$	$\delta_A^{(2)}$	$\delta_N^{(2)}$	$\delta_a^{(1)}$	$\delta_A^{(1)}$	$\delta_N^{(1)}$
0.2	2π	> 256	> 256	256	139.034	139	139
	4π	> 256	> 256	256	34.758	34	34
	8π	> 256	> 256	256	8.69	8	8
	10π	> 256	> 256	256	5.561	5	5
	16π	82.564	81	81	2.172	2	2
0.6	2π	> 256	> 256	256	34.758	34	34
	4π	> 256	> 256	256	8.69	8	8
	8π	> 256	> 256	256	2.172	2	2
	10π	169.092	162	162	1.39	1	1
	16π	41.282	37	37	0.543	0	0
0.8	2π	> 256	> 256	256	23.172	23	23
	4π	> 256	> 256	256	5.793	5	5
	8π	> 256	> 256	256	1.448	1	1
	10π	225.456	181	181	0.927	0	0
	16π	55.043	35	35	0.362	0	0

- the distances obtained from the L^h discretization-error formula (7.7), $\delta_A^{(2)}$, and analogously derived discretization-error formula for L_d^h, $\delta_A^{(1)}$;
- the distances computed in direct numerical simulations ($\delta_N^{(j)}$).

In analytical calculations, the exact solution of the differential problem (2.1) and (2.3) is always assumed to be $e^{i\omega_2(-tx+y)}$. In direct numerical simulations the exact solution has been chosen to be $\sin(\omega_2(-tx+y))$. The numerical distance is considered to be m if the L_∞ norm of the relative discretization error on the $(m+1)$th vertical line is greater than 0.01. The simulation grid is 257×257. If in analytical calculations the result exceeds 256 (i.e., the penetration distance covers all the domain), it has been set to > 256. Table 9.1 contains the test results. The two obvious conclusions are as follows.

1. The discrete half-space analysis is actually precise even without incorporating outflow boundary conditions. In all the tests, the results predicted by this analysis and obtained in real numerical calculations coincide.
2. The estimates (9.7) and (9.9) of the penetration distances are accurate, especially for the first-order operator or for small nonalignment angles ($t \le 0.6$). Some deterioration in predicting the second-order operator penetration distances for nearly diagonal alignment is explained by the fact that the penetration distance in the case of 45° angle of nonalignment ($t = 1$) is determined by the *third-order term*, which is not taken into account in calculating r_2. Nevertheless, the estimate (9.7) seems to be reliable for predicting the key property—whether the penetration distance is comparable with (or larger than) the characteristic size of the domain.

9.2. Convergence in different regimes. The discrete problem (2.4) approximating homogeneous differential equation (2.2) on the layer $(x, y) \in [0, 1] \times (-\infty, +\infty)$ was solved. The inflow boundary conditions $g(y) = e^{i\omega_2 y}$ and $g'(y) = e^{i(-\omega_1 h + \omega_2 y)}$ were derived from the assumption that $U(x, y) = e^{i(\omega_1 x + \omega_2 y)}$ $(\omega_1 + t\omega_2 = 0)$ is the desired solution of the differential equation (2.2). The problem is reduced to the 1D

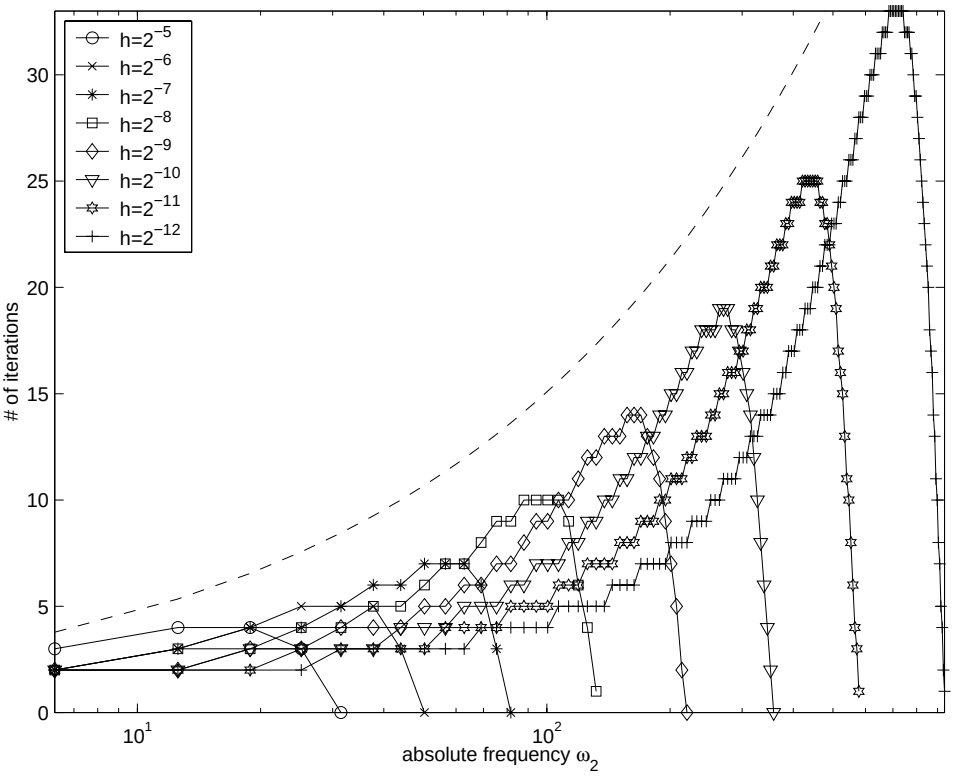


FIG. 9.1. *Algebraic-to-discretization error convergence for DC*1: *Dependence on the incoming frequency* ω_2 *and the mesh size* h.

problem (7.3)–(7.4) with outflow boundary conditions at $i_1 = N$ given by

$$b_{-2}\phi_{N-2} + b_{-1}\phi_{N-1} + b_0\phi_N = 0, \tag{9.10}$$

where

$$\begin{aligned} b_{-2} &= \tfrac{1}{2}, \\ b_{-1} &= -2, \\ b_0 &= \tfrac{3}{2} + \tfrac{t}{2}\Big(e^{-i2\Omega_2} - 4e^{-i\Omega_2} + 3\Big). \end{aligned} \tag{9.11}$$

The numerical experiments with the DC1 algorithm were performed following recipes incorporating the outflow boundary conditions, given in section 7.2. As noted earlier, this 1D analysis provides the exact simulation of the DC1 iterations performed for the 2D problem.

The tests were performed on the uniform grids with $h = 2^{-5}, 2^{-6}, \ldots, 2^{-12}$. The nonalignment angle was the same in all the tests ($t = 0.5$). The absolute (not normalized) value of the incoming frequency was varied from $\omega_2 = 2\pi$ with increment 2π. The initial approximation for the DC1 method was always obtained from the solution of the driver operator. The tests on each grid were stopped when one DC1 iteration succeeded to converge the L_2 norm of the algebraic error below the L_2 norm of the discretization error. The results of the tests are collected in Figure 9.1.

For any given inflow content, there is the worst grid on which the required number of iterations is maximal. With further grid refinement, the initial convergence rates become faster and eventually just two DC1 iterations are needed to converge the algebraic error below the discretization-error level.

One can distinguish three convergence regimes with respect to the relative accuracy of the target and driver operators. In the discussion below, the penetration distances are estimated as large or small with respect to the characteristic size of the domain.

Fast convergence regime—large penetration distance in the driver operator. For very smooth inflow boundary conditions, the solutions of both the target and the driver operators accurately approximate the true solution of the differential problem. Convergence rates are fast from the very beginning. Four DC1 iterations at most are needed to converge the algebraic errors to the discretization-accuracy level. This regime is established when the driver operator attains at least 40% accuracy ($\epsilon = 0.4$) over the entire domain. In this regime, the minimal number of required iterations is two, as expected.

Slow convergence regime—large penetration distance in the target operator, small penetration distance in the driver operator. This regime is established for the range of inflow boundary conditions, for which the target operator is accurate while the driver equation does not approximate the differential equation on a large part of the domain, i.e., the driver-operator penetration distance is substantially shorter than the characteristic size of the domain. The initial convergence rates are slow, and many iterations are required to converge the algebraic errors. The maximal number of iterations required to converge the algebraic error below the discretization-error level increases in a good accordance with FDA estimate (6.6). This regime is actually the most important one, from the consideration that it describes the convergence on grids minimally sufficient for accurate target-operator approximations of the true differential solution. Generally, the maximal number of iterations is required for the incoming components with about 50% accuracy ($\epsilon = 0.5$) provided by the target operator. This accuracy is not practical. Section 9.3 presents numerical experiments with more accurate approximations.

Fast convergence regime—small penetration distance in the target operator. The third regime where the initial convergence is fast again is established for relatively high-frequency inflow contents. In this regime, neither the target nor the driver operator is accurate.

The estimate (6.6) for the maximal number of iterations can be adjusted for the second regime by incorporating (9.7) and the assumption $\delta_2 = R$ as

$$N_{\text{sweeps}} \approx \frac{t+t^2}{2} \begin{cases} h^{-\frac{1}{3}} \left(\frac{12 \arccos(1-\frac{\epsilon^2}{2})}{t-t^3} \right)^{\frac{2}{3}}, \\ \omega_2^{\frac{1}{2}} \left(\frac{12 \arccos(1-\frac{\epsilon^2}{2})}{t-t^3} \right)^{\frac{1}{2}}. \end{cases} \tag{9.12}$$

This estimate is a counterpart to (6.6) with the target operator attaining the ϵ accuracy over the characteristic size of the domain. Substituting for the value of $\epsilon = 0.5$ observed in the numerical experiments in the second regime to (9.12) yields the dashed line in Figure 9.1; the variation of the maximal number of iteration is characterized well.

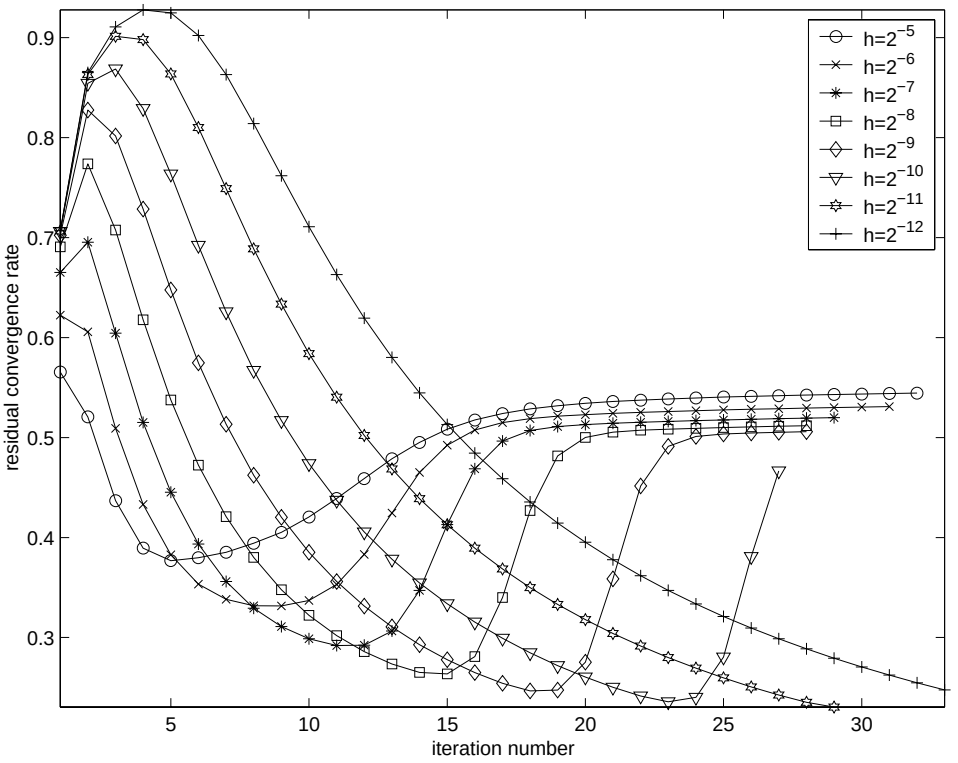


FIG. 9.2. *Residual convergence rate history for DC*1.

9.3. Grid-dependent convergence. Eight tests were performed on the same grids with the same nonalignment angle ($t = 0.5$) as in the previous section. The frequencies, ω_2, of the inflow Fourier components on different grids were chosen to provide the target-operator penetration distance to coincide with the characteristic size of the domain ($\delta_2 = \sqrt{1+t^2}$). The penetration distance was computed by the formula (9.7) for the 5% accuracy ($\epsilon = 0.05$), which is representative of the accuracy required in practice. The initial approximation for the defect-correction method was again obtained from the solution of the driver operator. Figure 9.2 depicts the residual convergence rate history in the experiments performed. The convergence rate in a DC1 iteration was defined as the ratio of the L_2 norms of the residual after and before the iteration. In each test, the last iteration shown on the Figure 9.2 corresponded to the approximate solution with the L_2 norm of the target-operator residual less than 10^{-10}. Figure 9.3 exhibits the ratios between the L_2 norms of the algebraic and discretization errors observed in the same experiments. In this figure, the solid horizontal line designates algebraic error equal to the discretization error. The first qualitative conclusion is obvious: the convergence of DC1 iterations is grid dependent, i.e., more iterations are required on finer grids to get into the asymptotic convergence regime and/or to converge the algebraic error below the discretization error level.

The quantitative characteristics of the experiments are collected in Table 9.2. The column $\omega_2 h$ shows the normalized frequency of the inflow boundary Fourier component having the unity penetration distance for 5% accuracy on the uniform grid with mesh size h. The columns $\rho(G_r)$ and $\|G_r\|_2$ exhibit the asymptotic convergence rate and the

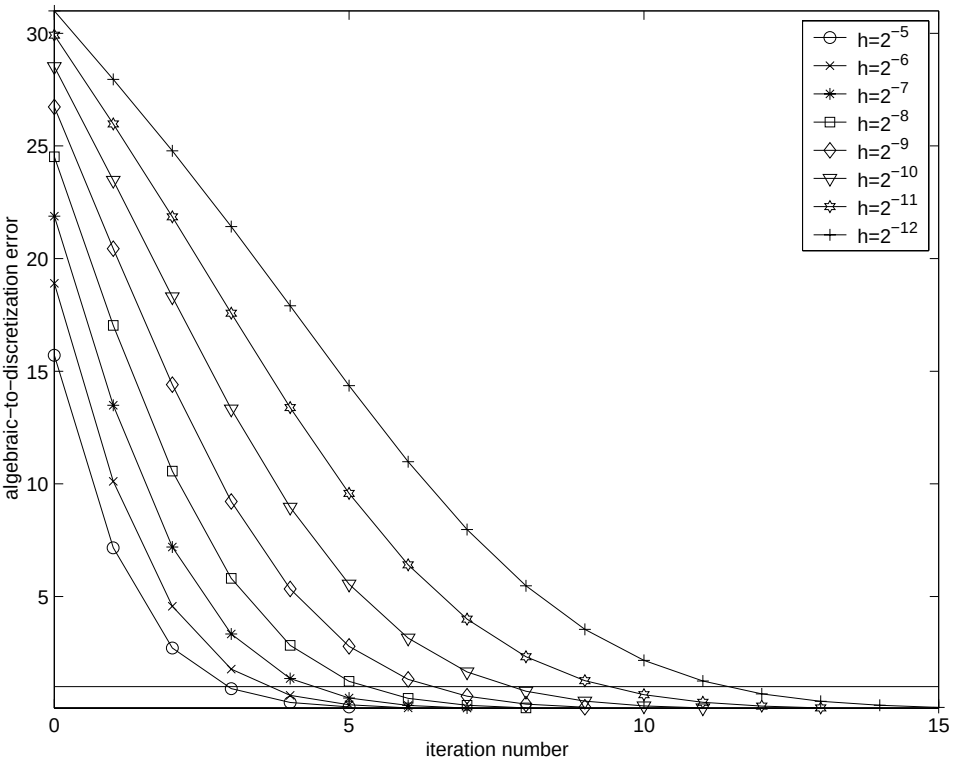


FIG. 9.3. *Algebraic-to-discretization error convergence history for DC*1.

TABLE 9.2
*Convergence tests: DC*1 *iterations.*

h	$\omega_2 h$	$\rho(G_r)$	$\|G_r\|_2$	Asympt. delay	Alg. error	Estimated N_{sweeps}
2^{-5}	0.37	0.5612	0.7894	2	3	2
2^{-6}	0.29	0.5491	0.7914	3	4	3
2^{-7}	0.23	0.5384	0.8116	4	5	3
2^{-8}	0.18	0.5299	0.8527	6	6	4
2^{-9}	0.15	0.5232	0.8923	7	7	5
2^{-10}	0.12	0.5181	0.9241	9	8	6
2^{-11}	0.09	0.5142	0.9479	12	10	7
2^{-12}	0.07	0.5111	0.9649	16	12	9

worst possible convergence rate, respectively, both calculated by 1D matrix analysis described in section 8. The DC1 iteration process was considered to have reached the asymptotic convergence regime when an iteration convergence rate became less than the asymptotic convergence rate ($\rho(G_r)$) on the given grid. The number of iterations required to reach the asymptotic convergence regime is shown in the column labeled "Asympt. delay." The column marked "Alg. error" demonstrates the number of DC1 iterations required to converge the algebraic error below the discretization error level. The last column, marked "Estimated N_{sweeps}," is the number of sweeps calculated from (9.12) for $\epsilon = 0.05$.

The fast asymptotic convergence rate predicted by the 1D matrix analysis and confirmed in the numerical tests is not a surprise. It was observed by many researchers

that the asymptotic convergence rate of the defect-correction solver for (2.4) is about 0.5 per iteration. In [8], the authors emphasized that the asymptotic convergence rate deteriorates for the central and the pure upwind target discretizations.

The bound $\|G_r\|_2$ is not very sharp in the presented tests. The full-space Fourier analysis (eliminating the characteristic components from the consideration) gives a similar estimate (see [8]). This similarity suggests that to observe this "worst" behavior one should test nonhomogeneous problems or consider special initial approximations.

The numerical tests corroborate the conclusion derived from the half-space analysis that the number of DC1 iterations required to obtain the asymptotic convergence rate or to have the algebraic error smaller than the discretization error might grow on fine grids. The quantitative prediction that the growth is proportional to $h^{-\frac{1}{3}}$, i.e., the number of the iterations is doubled when the grid becomes eight times larger (mesh size h is replaced with $h/8$), has also been confirmed with a good accuracy (see Table 9.2). The crude estimate (9.12) is a good indicator of the h-dependent delay in convergence. A smarter choice of the initial approximation (e.g., the initial approximation interpolated from the solution on a coarser grid) can shorten somewhat this delay on each given grid, but the qualitative behavior remains the same: the number of required defect-correction iterations grows as $h^{-\frac{1}{3}}$ in passing to finer grids.

There are several ways to change the algorithm in order to make its convergence grid independent:

1. The first possibility, studied in section 10, is to apply a driver of the same approximation order as the target operator.
2. The second way is to use a predictor-corrector technique for solving the target operator. This method suggests some marching along the flow direction. Preliminary numerical tests indicate that if the marching is possible (there are no recirculation zones), then the predictor-corrector method demonstrates the optimal efficiency.
3. The third method is a semicoarsening multigrid algorithm which employs the coarse-grid operators closely approximating the *characteristic-component FDA* of the target operator (see [9]).

10. Defect-correction method with second-order driver. In this section, the defect-correction algorithm with the second order driver (DC2) is presented. The driver operator which is the second-order accurate upwind discretization is accompanied with the same inflow boundary condition as the target operator (2.4).

$$
\begin{aligned}
L_d^h u_{i_1,i_2} = \frac{1}{2h}\Big(& b_1\Big(3u_{i_1,i_2} - 4u_{i_1-1,i_2} + u_{i_1-2,i_2}\Big) \\
& + b_2\Big(3u_{i_1,i_2} - 4u_{i_1,i_2-1} + u_{i_1,i_2-2}\Big)\Big), \\
i_1 = 1, 2, \ldots, N, & \\
u_{0,i_2} = g(i_2 h),\ & u_{-1,i_2} = g'(i_2 h).
\end{aligned}
\tag{10.1}
$$

The discretization scheme (10.1) is stable. This scheme may be considered as a target scheme by itself. In extension to convection-diffusion equation, however, purely upwind schemes do not possess the advantage of a one-shot solution and the accuracy considerations become dominant. The Fromm scheme (2.4) is about four times more accurate than the scheme (10.1).

We tested the DC2 iterations for the same test cases as in section 9.3. The results confirm that the DC2 solver is optimally efficient on all the grids. The convergence in DC2 iterations is grid independent. Convergence rates faster than 0.5 per iteration are obtained from the very beginning. The DC2 solver requires only one sweep to reduce the algebraic error substantially below the level of discretization accuracy.

The efficiency demonstrated by DC2 solver indicates that the second-order driver would be preferable for solving high-order operators by defect-correction method. On the other hand, implementation of second-order drivers in practical problems is not straightforward. For example, in solving discretized multidimensional hyperbolic systems of equations where downstream marching is impossible, first-order schemes are considered to be much easier to solve than second-order schemes. Nevertheless, the opportunity for employing the second-order driver should always be carefully considered.

11. Conclusions. This paper presented a novel, comprehensive, discrete, half-space analysis for a defect-correction method solving the Fromm discretization of the 2D convection equation. This analysis is precise for constant-coefficient problems on layers and can be easily generalized to the convection-diffusion and strongly anisotropic problems. The half-space analysis plays the same role for nonelliptic problem solvers as the full-space Fourier mode analysis plays for elliptic problem solvers; in nonelliptic problems, it accounts for both the damping and the downstream transport of errors. The analytical solution on the half-space is defined as a combination of a finite number of analytical components and a pointwise component. The analytical components represent the solution away from the boundary. The role of the pointwise component is to correct the analytic representation in the region adjacent to the boundary so that the combined approximate solution satisfies the inflow boundary conditions. Any locally discretized inflow boundary conditions can be considered in the framework of this analysis. In each DC1 iteration the pointwise representation region penetrates in the interior by one mesh size. By using these representations, the computational complexity of the analysis becomes much less than that associated with the 1D matrix analysis. In the asymptotic regime, when the pointwise representation zone covers all the domain, this analysis becomes a discrete 1D matrix analysis of the multidimensional problem. The discrete half-space analysis was found to be an accurate and very efficient tool for predicting actual solution behavior.

The following important findings about the analyzed defect-correction method were reported:

1. The initial convergence rate of the defect-correction method is principally a function of the relative accuracy of the driver and target operators, as reasoned by the penetration distances for characteristic components.
2. The asymptotic convergence rate of the defect-correction method is about 0.5 per iteration.
3. The efficiency of the defect-correction method with the first-order driver is grid dependent. The maximal number of iterations required to reach the discretization accuracy or to obtain the asymptotic convergence rate can be proportional to $h^{-\frac{1}{3}}$, corresponding to the characteristic error components with the target-operator penetration distances approximately equal to (and the driver-operator penetration distances substantially less than) the characteristic size of the domain.
4. Using the second-order driver in the defect-correction iterations eliminates this h-dependence and results in an optimally efficient solver.

REFERENCES

[1] S. A. Allmaras, *Multigrid for the* 2*-D Compressible Navier-Stokes Equations*, AIAA Paper 99-3336, 14th AIAA CFD Conference, Norfolk, VA, 1999.

[2] K. Böhmer, P. W. Hemker, and H. J. Stetter, *The defect correction approach*, in Defect Correction Methods, K. Böhmer and H. J. Stetter, eds., Comp. Suppl. 5, Springer-Verlag, Vienna, New York, 1984, pp. 1–32.

[3] A. Brandt, *Multigrid Solvers for Non-Elliptic and Singular-Perturbation Steady-State Problems*, manuscript, The Weizmann Institute of Science, Rehovot, Israel, 1981.

[4] A. Brandt, *The Weizmann Institute of Science research in multilevel computations: 1988 Report*, in Proceedings of the Fourth Copper Mountain Conference on Multigrid Methods, J. Mandel, S. F. McCormick, J. E. Dendy, Jr., C. Farhat, G. Lonsdale, S. V. Parker, J. W. Ruge, and K. Stuben, eds., SIAM, Philadelphia, 1989, pp. 13–53.

[5] A. Brandt and B. Diskin, *Multigrid solvers for the non-aligned sonic flow: The constant coefficient case*, Comput. & Fluids, 28 (1999), pp. 511–549.

[6] A. Brandt and I. Yavneh, *On multigrid solution of high-Reynolds incompressible entering flow*, J. Comput. Phys., 101 (1992), pp. 151–164.

[7] J. A. Désidéri and P. W. Hemker, *Analysis of Convergence of Iterative Implicit and Defect-Correction Algorithms for Hyperbolic Problems*, Report 1200, Institut National de Recherche en Informatique et en Automatique, Valbonne, France, 1990.

[8] J. A. Désidéri and P. W. Hemker, *Convergence analysis of the defect-correction iteration for hyperbolic problems*, SIAM J. Sci. Comput., 16 (1995), pp. 88–118.

[9] B. Diskin, *Solving Upwind-Biased Discretizations* II: *Multigrid Solver Using Semicoarsening*, ICASE Report 99-25, ICASE, Hampton, VA, 1999.

[10] B. Diskin and J. L. Thomas, *Solving Upwind-Biased Discretizations: Defect-Correction Iterations*, ICASE Report 99-14, ICASE, Hampton, VA, 1999.

[11] B. Koren, *Defect correction and multigrid for an efficient and accurate computation of airfoil flows*, J. Comput. Phys., 77 (1988), pp. 183–206.

[12] B. Koren, *Multigrid and defect correction for the steady Navier-Stokes equations*, J. Comput. Phys., 87 (1990), pp. 25–46.

[13] S. L. Krist, R. T. Biedron, and C. L. Rumsey, *CFL3D User's Manual (Version* 5.0*)*, NASA TM-1998-208444, NASA, Hampton, VA, 1998.

[14] C. W. Oosterlee, F. J. Gaspar, T. Washio, and R. Wienands, *Multigrid line smoothers for higher order upwind discretizations of convection-dominated problems*, J. Comput. Phys., 139 (1998), pp. 274–307.

[15] J. L. Thomas, D. L. Bonhaus, W. K. Anderson, C. L. Rumsey, and R. T. Biedron, *An* $O(n\,m^2)$ *Plane Solver for the Compressible Navier-Stokes Equations*, AIAA Paper 99-0785, 37th Aerospace Sciences Meeting and Exhibit, Reno, NV, 1999.

[16] J. L. Thomas, B. Diskin, and A. Brandt, *Distributed Relaxation Multigrid and Defect Correction Applied to the Compressible Navier-Stokes Equations*, AIAA Paper 99-3334, 14th Computational Fluid Dynamics Conference, Norfolk, VA, 1999.

[17] J. L. Thomas, B. Diskin, and A. Brandt, *Textbook Multigrid Efficiency for the Incompressible Navier-Stokes Equations: High Reynolds Number Wakes and Boundary Layers*, ICASE Report 99-51, ICASE, Hampton, VA, 1999.

[18] P. Wesseling, *An Introduction to Multigrid Methods*, Pure Appl. Math., John Wiley, Chichester, UK, 1992.

[19] N. Yanenko and Y. I. Shokin, *On the correctness of first differential approximations of difference schemes*, Dokl. Akad. Nauk SSSR, 182 (1968), pp. 776–778.

[20] N. Yanenko and Y. I. Shokin, *On the first differential approximations of difference schemes for hyperbolic systems of equations*, Siberian Math. J., 10 (1969), pp. 1174–1188.

SIAM J. SCI. COMPUT.
Vol. 22, No. 2, pp. 656–672
© 2000 Society for Industrial and Applied Mathematics

COMPACT CENTRAL WENO SCHEMES FOR MULTIDIMENSIONAL CONSERVATION LAWS*

DORON LEVY†, GABRIELLA PUPPO‡, AND GIOVANNI RUSSO§

Abstract. We present a new third-order central scheme for approximating solutions of systems of conservation laws in one and two space dimensions. In the spirit of Godunov-type schemes, our method is based on reconstructing a piecewise-polynomial interpolant from cell-averages which is then advanced exactly in time.

In the reconstruction step, we introduce a new third-order, *compact*, central weighted essentially nonoscillatory (CWENO) reconstruction, which is written as a convex combination of interpolants based on different stencils. The heart of the matter is that one of these interpolants is taken as a suitable quadratic polynomial, and the weights of the convex combination are set as to obtain third-order accuracy in smooth regions. The embedded mechanism in the WENO-like schemes guarantees that in regions with discontinuities or large gradients, there is an automatic switch to a one-sided second-order reconstruction, which prevents the creation of spurious oscillations.

In the one-dimensional case, our new third-order reconstruction is based on an extremely compact *three*-point stencil. Analogous compactness is retained in more space dimensions. The accuracy, robustness, and high-resolution properties of our scheme are demonstrated in a variety of one- and two-dimensional problems.

Key words. hyperbolic systems, central difference schemes, high-order accuracy, nonoscillatory schemes, WENO reconstruction, CWENO reconstruction

AMS subject classifications. Primary, 65M10; Secondary, 65M05

PII. S1064827599359461

1. Introduction.

We are concerned with multidimensional systems of hyperbolic conservation laws of the form

$$u_t + \nabla_x \cdot f(u) = 0, \qquad x \in \mathbb{R}^d, \quad u = (u_1, \ldots, u_n). \tag{1.1}$$

Methods for approximating solutions to (1.1) have attracted a lot of attention in recent years (see [5], [11], [25], and the references therein).

In this work we focus on Godunov-type schemes, where one first reconstructs a piecewise-polynomial interpolant which is then advanced exactly in time according to (1.1) and finally projected on its cell-averages. Generally, one can divide Godunov-type schemes into two subclasses—*upwind methods* and *central methods*.

In *upwind schemes*, one first reconstructs a polynomial in every cell, which is then used to compute a new cell-average in the same location in the next time step. This procedure requires solving Riemann problems at the discontinuous interfaces. For high-order methods, instead of analytically solving the resulting Riemann problems,

*Received by the editors July 29, 1999; accepted for publication (in revised form) November 15, 1999; published electronically August 9, 2000. The work of the first author was supported in part by the Applied Mathematical Sciences subprogram of the Office of Energy Research, U.S. Department of Energy, under contract DE–AC03–76–SF00098. The work of the second author was partly supported by the MURST-Cofin '98 program. Part of this work was done while the second and third authors were visiting the Lawrence Berkeley National Lab.

http://www.siam.org/journals/sisc/22-2/35946.html

†Department of Mathematics, University of California, Berkeley, CA 94720, and Lawrence Berkeley National Lab (dlevy@math.berkeley.edu).

‡Dipartimento di Matematica, Politecnico di Torino, Corso Duca degli Abruzzi 24, 10129 Torino, Italy (puppo@calvino.polito.it).

§Dipartimento di Matematica, Università dell'Aquila, Via Vetoio, loc. Coppito - 67100 L'Aquila, Italy (russo@univaq.it).

one typically implements approximate Riemann solvers or some form of flux splitting. For systems of conservation laws, or in the demanding context of more space dimensions, this procedure turns out to be more intricate as such Riemann solvers do not exist. The essentially nonoscillatory (ENO) methods by Harten are the prototype of high-order methods (see [6], [23], and the references therein). A recent review of ENO and weighted essentially nonoscillatory (WENO) methods can be found in [22].

Central schemes, on the other hand, are based on averaging over the Riemann fans, a procedure which is typically done by staggering between two grids. They require no Riemann solvers, no projection along characteristic directions, and no flux splitting. Therefore, all that one has to do in order to solve a problem is to supply the flux function. They are more simple when compared with upwind schemes.

The major difference between different central methods is in the reconstruction step, where one computes a piecewise-polynomial interpolant from the previously computed cell-averages.

The prototype of central schemes is the Lax–Friedrichs [4] scheme which is based on a piecewise-constant interpolant. Even though it is very robust, it is only first-order accurate and, moreover, it suffers from excessive numerical dissipation. A second-order central method was proposed by Nessyahu and Tadmor in [20]. This method is based on a MUSCL-like [10] piecewise linear interpolant and nonlinear limiters which prevent spurious oscillations (see [21] for a different approach). A variety of extensions to these methods were suggested. The one-dimensional third-order method of Liu and Tadmor [19] is based on the third-order reconstruction by Liu and Osher in [17]. For the two-dimensional method, see [1] and [8].

A first step for importing the high-order reconstructions that were derived in the upwind framework was taken in [2]. There, the ENO method was transformed into the central setup, and a new *mostly centered stencil* was shown to produce the least oscillatory results. The next step was taken in the one-dimensional case in [13], where a new *central weighted nonoscillatory* (CWENO) reconstruction was introduced. This CWENO method is based on the upwind WENO methods by [18] and [7], in which an interpolant is written as a convex combination of several reconstructions which are based on different stencils. These methods include an internal switch which is designed such as to provide the maximum possible accuracy in smooth regions, while automatically switching to a more robust one-sided stencil in the presence of discontinuities and large gradients. The one-dimensional CWENO method was extended to two space dimensions in [15] and [16]. For a numerical study of the behavior of the total variation for the CWENO scheme, see [14].

In this paper we present a new, compact CWENO reconstruction. This new reconstruction is based on defining a suitable quadratic function which is added to linear interpolants in such a way as to obtain third-order accuracy in smooth regions (in one and two space dimensions). In regions with discontinuities or large gradients, the weights are automatically changed so that they switch to a one-sided second-order linear reconstruction. This reconstruction turns out to be extremely compact; in the one-dimensional case, e.g., the reconstruction is based on a three-point stencil.

The structure of this paper is as follows: We start in section 2 with a brief overview of central schemes for conservation laws in one and two space dimensions.

We then proceed to present our new, compact, third-order CWENO reconstruction in section 3. The idea of introducing a quadratic reconstruction is first presented in the one-dimensional framework and then extended to two space dimensions.

We end in section 4 where we demonstrate our new method in several test cases.

First, the accuracy tests (both in one-dimensional and two-dimensional cases) show the third-order accuracy of the method. We then solve the one-dimensional system of the Euler equations of gas dynamics for a few test problems and we illustrate the behavior of the scheme in scalar two-dimensional cases. In particular, we would like to stress that in our numerical results we observe a very robust and nonoscillatory behavior of the weights, which can be related to the overall robustness and accuracy properties of our new method.

2. Central schemes for conservation laws. In this section we give a brief overview of central schemes for approximating solutions to hyperbolic conservation laws in one and two space dimensions. For further details we refer the reader to [25], [8], [13], and the references therein.

Starting in the one-dimensional case, we seek numerical solutions of the Cauchy problem

$$\begin{cases} u_t + f(u)_x = 0, \\ \\ u(x, t{=}0) = u_0(x). \end{cases} \tag{2.1}$$

For simplicity, we introduce a uniformly spaced grid in the (x,t) space, where the mesh spacings are denoted by $h := \Delta x$ and $k := \Delta t$, respectively. We denote by $\bar{u}_j^n$ the numerical approximation of the cell average in the cell $I_j := [x_{j-1/2}, x_{j+1/2}]$ at time $t^n = nk$, where $x_j = jh$. The finite-difference method will approximate the cell-averages at time t^{n+1} based on their values at time t^n.

We start by reconstructing at time t^n a piecewise-polynomial conservative interpolant from the known cell-averages, $\bar{u}_j^n$, i.e.,

$$P_u(x, t^n) := \sum_j R_j(x)\chi_j, \tag{2.2}$$

where χ_j is the characteristic function of the interval I_j and $R_j(x)$ is a polynomial defined in I_j. It is here, in the reconstruction step, where the accuracy and nonoscillatory requirements enter. Different central methods will be typically based on different reconstructions.

The reconstruction, $P_u(x, t^n)$, is then evolved exactly in time (integrating (2.1)), and then projected on staggered cells, in order to compute the cell average at $I_{j+1/2}$. With this procedure we obtain

$$\bar{u}_{j+1/2}^{n+1} = \bar{u}_{j+1/2}^n + \frac{1}{h}\int_{t^n}^{t^{n+1}} [f(P_u(x_j, \tau)) - f(P_u(x_{j+1}, \tau))]\, d\tau. \tag{2.3}$$

Based on the reconstruction, (2.2), the first term on the right-hand side (RHS) of (2.3) can be explicitly computed,

$$\bar{u}_{j+1/2}^n = \frac{1}{h}\int_{x_j}^{x_{j+1}} P_u(x, t^n)\, dx.$$

In order to compute the second integral on the RHS of (2.3), namely the integral in time over the fluxes, one should observe that due to the staggering, up to a suitable Courant–Friedrichs–Lewy (CFL) condition, these integrals involve only smooth functions, and hence can be approximated by a sufficiently smooth quadrature. A second-order method can be obtained, e.g., using the midpoint rule in time (see [20]).

Simpson's rule for the quadrature in time will provide fourth-order accuracy (which will naturally be sufficient also for a third-order method).

The quadratures for the time integrals of the fluxes require the *prediction* of point-values of the function in several points in the interval $[t^n, t^{n+1}]$. One possible approach is to use a Taylor expansion based on (2.1). Such an approach was used, e.g., in [20] and [19]. In order to avoid the technical complications involved in the Taylor series (in particular with high-order methods, and when dealing with systems of equations), one can alternatively use a Runge–Kutta (RK) method directly on (2.1) to predict the required values in later times. By using the *natural continuous extension* (NCE) of RK schemes [26], the value of the flux at the nodes of the quadrature formula can be computed with a single RK step. Such a method is simpler compared with the Taylor expansion method, but it does require another reconstruction: the reconstruction of the point values of the derivatives of the fluxes at time t^n which are then used as the input to the RK solver (see [2] and [13] for more details).

The simplicity of central schemes manifests itself when turning to deal with systems of equations. Basically, the algorithm that was described in the scalar case repeats itself componentwise. Based on the type of the reconstruction, when solving systems of equations, there can even be simplifications over a purely componentwise extension of the scalar scheme. These can be found, e.g., in section 3.3 below.

The extension to two space dimensions is straightforward: We consider

$$u_t + f(u)_x + g(u)_y = 0, \tag{2.4}$$

subject to the initial condition, $u(x, y, t{=}0) = u_0(x, y)$. Here, Δx and Δy will denote the spatial mesh spacings, while Δt denotes the spacing in time. The two-dimensional cells are now $I_{i,j} = [x_{i-1/2}, x_{i+1/2}] \times [y_{j-1/2}, y_{j+1/2}]$.

Following the general methodology, we wish to construct the cell-averages, $\bar{u}^{n+1}_{j,k}$, based on the cell-averages at time t^n. First, we construct a two-dimensional interpolant which reads

$$P_u(x, y, t^n) = \sum_{i,j} R_{i,j}(x, y)\chi_{i,j}, \tag{2.5}$$

with $\chi_{i,j}$ being the characteristic function of the cell $I_{i,j}$ and $R_{i,j}(x, y)$ a polynomial of a suitable degree. An exact evolution in time of the interpolant (2.5) which is projected on its cell-averages now reads (compare with (2.3))

(2.6)

$$\begin{aligned}\bar{u}^{n+1}_{i+1/2,j+1/2} = \bar{u}^{n}_{i+1/2,j+1/2} &+ \frac{1}{\Delta x \Delta y}\int_{t^n}^{t^{n+1}}\int_{y=y_j}^{y_{j+1}} [f(P_u(x_i, y, \tau)) - f(P_u(x_{i+1}, y, \tau)]\, dy d\tau \\ &+ \frac{1}{\Delta x \Delta y}\int_{t^n}^{t^{n+1}}\int_{x=x_i}^{x_{i+1}} [f(P_u(x, y_j, \tau)) - f(P_u(x, y_{j+1}, \tau)]\, dx d\tau.\end{aligned}$$

The staggered cell-average at time t^n can be directly computed by

$$\bar{u}^n_{i+1/2,j+1/2} = \frac{1}{\Delta x \Delta y}\int_{x_i}^{x_{i+1}}\int_{y_j}^{y_{j+1}} P_u(x, y, t^n)\, dy dx.$$

Analogously to the one-dimensional case, the flux integrals on the RHS of (2.6) can be approximated using a quadrature rule in time. This can be coupled with a quadrature

rule for the line integrals in space. Here it is necessary to avoid quadrature points at which the fluxes may not be smooth. The details can be found, e.g., in [15] and [16] (see also [8]).

3. A compact third-order CWENO reconstruction.

3.1. The one-dimensional framework. In this section we derive our new CWENO reconstruction in one space dimension. The two-dimensional extension will follow in section 3.2.

We first note that in the absence of large gradients, we obtain third-order accuracy if we choose for the reconstruction the *optimal* polynomial

$$P_j(x) = P_{\text{OPT},j}(x),$$

where $P_{\text{OPT},j}(x)$ is the parabola that interpolates the data $\bar{u}^n_{j-1}, \bar{u}^n_j, \bar{u}^n_{j+1}$ in the sense of cell-averages to enforce conservation:

$$\int_{x_{j+l-1/2}}^{x_{j+l+1/2}} P_{\text{OPT},j}(x)\,dx = \bar{u}^n_{j+l}, \qquad l = -1, 0, 1.$$

These conditions determine $P_{\text{OPT},j}(x)$ completely, namely,

$$P_{\text{OPT},j}(x) = u^n_j + u'_j(x - x_j) + \frac{1}{2}u''_j(x - x_j)^2, \tag{3.1}$$

with

$$u^n_j = \bar{u}^n_j - \frac{1}{24}(\bar{u}^n_{j+1} - 2\bar{u}^n_j + \bar{u}^n_{j-1}),$$
$$u'_j = \frac{\bar{u}^n_{j+1} - \bar{u}^n_{j-1}}{2\Delta x}, \qquad u''_j = \frac{\bar{u}^n_{j-1} - 2\bar{u}^n_j + \bar{u}^n_{j+1}}{\Delta x^2}.$$

However, when discontinuities or large gradients occur, this reconstruction would be oscillatory. Therefore, following the WENO methodology [18], [7], [13], we construct an ENO interpolant as a convex combination of polynomials which are based on different stencils. Specifically, in the cell I_j we write

$$P_j(x) = \sum_i w^j_i P^j_i(x), \quad \sum_i w^j_i = 1, \quad w_i \geq 0, \quad i \in \{\text{L}, \text{C}, \text{R}\}, \tag{3.2}$$

where P_{L} and P_{R} are linear functions based on a left stencil and a right stencil, respectively, and P_{C} is a quadratic polynomial. In order to simplify the notations, we will omit the upper index j, remembering that the weights and the three polynomials change from cell to cell.

Conservation requires that $P_{\text{R}}(x)$ will interpolate the cell-averages

$$\int_{x_{j-1/2}}^{x_{j+1/2}} P_{\text{R}}(x)dx = \Delta x\,\bar{u}^n_j, \qquad \int_{x_{j+1/2}}^{x_{j+3/2}} P_{\text{R}}(x)dx = \Delta x\,\bar{u}^n_{j+1},$$

which, in turn, results with

$$P_{\text{R}}(x) = \bar{u}^n_j + \frac{\bar{u}^n_{j+1} - \bar{u}^n_j}{\Delta x}(x - x_j). \tag{3.3}$$

Similarly, for the left interpolant we have

$$P_{\mathrm{L}}(x) = \bar{u}_j^n + \frac{\bar{u}_j^n - \bar{u}_{j-1}^n}{\Delta x}(x - x_j). \tag{3.4}$$

All that is left is to reconstruct a centered polynomial, P_{C}, such that the convex combination, (3.2), will be third-order accurate in smooth regions. It must, therefore, satisfy

$$P_{\mathrm{OPT}}(x) = C_{\mathrm{L}}P_{\mathrm{L}}(x) + C_{\mathrm{R}}P_{\mathrm{R}}(x) + C_{\mathrm{C}}P_{\mathrm{C}}(x), \quad \sum_i C_i = 1, \quad i \in \{\mathrm{L}, \mathrm{C}, \mathrm{R}\}, \tag{3.5}$$

where $C_{\mathrm{L}}, C_{\mathrm{C}}$, and C_{R} are constants. Due to the staggering between every two consecutive steps of the central method, our reconstruction should provide half-cell averages which are third-order accurate. A straightforward calculation shows that *any* symmetric choice of constants C_i in (3.5) provides the desired accuracy. In particular, for the specific choice of $C_{\mathrm{L}} = C_{\mathrm{R}} = 1/4$, (3.3)–(3.5) yield

$$\begin{aligned} P_{\mathrm{C}}(x) &= 2P_{\mathrm{OPT}}(x) - \frac{1}{2}(P_{\mathrm{R}}(x) + P_{\mathrm{L}}(x)) = \bar{u}_j^n - \frac{1}{12}(\bar{u}_{j+1}^n - 2\bar{u}_j^n + \bar{u}_{j-1}^n) \\ &\quad + \frac{\bar{u}_{j+1}^n - \bar{u}_{j-1}^n}{2\Delta x}(x - x_j) + \frac{\bar{u}_{j+1}^n - 2\bar{u}_j^n + \bar{u}_{j-1}^n}{\Delta x^2}(x - x_j)^2. \end{aligned} \tag{3.6}$$

In order to complete the reconstruction of $P_j(x)$ in (3.2), it is left to compute the weights w_i. Following [7] and [13], we write

$$w_i = \frac{\alpha_i}{\sum_k \alpha_k}, \qquad \alpha_i = \frac{C_i}{(\varepsilon + IS_i)^p}, \qquad i, k \in \{\mathrm{L}, \mathrm{C}, \mathrm{R}\}. \tag{3.7}$$

The constants C_is in (3.7) are chosen to be the same as in (3.5), i.e., $C_{\mathrm{L}} = C_{\mathrm{R}} = 1/4$, while $C_{\mathrm{C}} = 1/2$.

The *smoothness indicators*, IS_i, are responsible for detecting large gradients or discontinuities and to automatically switch to the stencil that generates the least oscillatory reconstruction in such cases. Once again, we follow [7] and [13] and define in each cell I_j the three smoothness indicators, IS_i, as

$$IS_i = \sum_{l=1}^{2} \int_{x_{j-1/2}}^{x_{j+1/2}} h^{2l-1}(P_i^{(l)}(x))^2 dx, \qquad i \in \{\mathrm{L}, \mathrm{C}, \mathrm{R}\}. \tag{3.8}$$

A direct computation, based on (3.3), (3.4), and (3.6), yields

$$\begin{aligned} IS_{\mathrm{L}} &= (\bar{u}_j^n - \bar{u}_{j-1}^n)^2, \qquad IS_{\mathrm{R}} = (\bar{u}_{j+1}^n - \bar{u}_j^n)^2, \\ IS_{\mathrm{C}} &= \frac{13}{3}(\bar{u}_{j+1}^n - 2\bar{u}_j^n + \bar{u}_{j-1}^n)^2 + \frac{1}{4}(\bar{u}_{j+1}^n - \bar{u}_{j-1}^n)^2. \end{aligned} \tag{3.9}$$

The constant ε is taken to prevent the denominator from vanishing. Furthermore, its value has to satisfy two requirements, namely,

(i) $\varepsilon + h^2 u_x^2 \gg |IS - h^2 u_x^2|$ in smooth regions,

(ii) $\varepsilon \ll IS$ near discontinuities.

The first condition guarantees that, on smooth regions, the weights are basically equal to the constants that provide high accuracy. The second condition guarantees that in the presence of a discontinuity, the weights of the parabola and of one of the one-sided

linear reconstructions will be practically zero, and we will be left with the other one-sided linear reconstruction. Hence, our third-order method automatically switches to a second-order method in the presence of large gradients, which is exactly what makes it so robust as will be evident in our numerical computations presented below.

The constant p weights the departure from smoothness. We used the value $p = 2$. For a discussion about its choice, see [7] and [13].

A second, nonoscillatory reconstruction is required for the flux derivative. It is natural to adapt the reconstruction that we used for the half-cell averages also to the reconstruction of the point-values of the flux derivative. Here, however, the interpolation requirements will be in the sense of point-values instead of cell-averages. Once again, a direct computation shows that any symmetric choice of constants will provide the desired accuracy, and it will only be natural to use the same constants, C_is, that were used in (3.5). This is simpler than the case of [13] where we had to use different sets of constants for the two different reconstructions (the reconstruction of the half-cell averages and the reconstruction of the point-values of the flux derivative at the center of the cells).

Remarks.

1. We would like to emphasize that our new method is based on adding the parabola P_{C} into the convex combination which is the heart of our nonoscillatory reconstruction. Our numerical simulations showed that the freedom we have in selecting the constants C_i has no influence on the properties of the method. It is easy to prove that we obtain third-order accuracy regardless of the smoothness of the weights, as long as they are symmetric. Of course this is mainly of theoretical interest, since it is very unlikely that only a smooth departure from symmetry is maintained if the weights are not smooth.
2. By extending our new ideas, one can modify our previous CWENO method presented in [13] in order to obtain a fifth-order, central, nonoscillatory scheme. This can be done by simply adding a fourth-order polynomial for the computation of the point-values.
3. Our new reconstruction is equivalent to limiting with the minimum slope instead of slope zero in the presence of a discontinuity. Hence, it is based on a second-order reconstruction close to shocks, unlike the scheme of [19] which can be only first-order accurate in such regions.
4. In the one-dimensional case, the additional parabola is needed only for the accurate recovery of the point-values. In the two-dimensional framework, however, the equivalent additional parabola will be required also for the accurate reconstruction of the fractional cell-averages.

3.2. A two-dimensional extension. We extend the ideas of section 3.1 to the two-dimensional framework. This extension is straightforward and is based on reconstructing an interpolant as a convex combination of four one-sided, piecewise-linear interpolants, and a centered, quadratic interpolant, in order to get the desired third-order accuracy in smooth regions.

Following these ideas, the reconstruction in the cell I_{ij}, can be written as

$$P_{i,j}(x,y) = \sum_k w_k^{i,j} P_k^{i,j}(x,y), \qquad k \in \{\mathrm{NE}, \mathrm{NW}, \mathrm{SE}, \mathrm{SW}, \mathrm{C}\}, \tag{3.10}$$

with $\sum_k w_k^{i,j} = 1$, and $w_k^{i,j} \geq 0$. Here, P_{NE}, P_{NW}, P_{SE}, and P_{SW} are the one-sided linear reconstructions, and P_{C} is a centered quadratic reconstruction (see Figure 3.1). As in

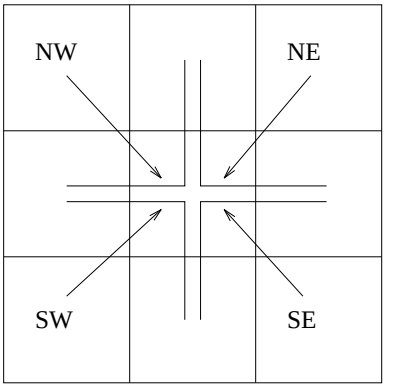


FIG. 3.1. *The two-dimensional stencil.*

the one-dimensional case, we will simplify our notations by omitting the superscripts, remembering that both the weights and the polynomials change from cell to cell.

Clearly, in analogy with the one-dimensional case, (3.3)–(3.4), the four linear reconstructions are given by

$$
\begin{aligned}
P_{\text{NE}}(x,y) &= \bar{u}^n_{i,j} + \frac{\bar{u}^n_{i+1,j} - \bar{u}^n_{i,j}}{\Delta x}(x - x_i) + \frac{\bar{u}^n_{i,j+1} - \bar{u}^n_{i,j}}{\Delta y}(y - y_j),\\
P_{\text{NW}}(x,y) &= \bar{u}^n_{i,j} + \frac{\bar{u}^n_{i,j} - \bar{u}^n_{i-1,j}}{\Delta x}(x - x_i) + \frac{\bar{u}^n_{i,j+1} - \bar{u}^n_{i,j}}{\Delta y}(y - y_j),\\
P_{\text{SW}}(x,y) &= \bar{u}^n_{i,j} + \frac{\bar{u}^n_{i,j} - \bar{u}^n_{i-1,j}}{\Delta x}(x - x_i) + \frac{\bar{u}^n_{i,j} - \bar{u}^n_{i,j-1}}{\Delta y}(y - y_j),\\
P_{\text{SE}}(x,y) &= \bar{u}^n_{i,j} + \frac{\bar{u}^n_{i+1,j} - \bar{u}^n_{i,j}}{\Delta x}(x - x_i) + \frac{\bar{u}^n_{i,j} - \bar{u}^n_{i,j-1}}{\Delta y}(y - y_j).
\end{aligned}
\tag{3.11}
$$

The centered polynomial, $P_{\text{C}}(x,y)$, is taken such as to satisfy

$$
P_{\text{OPT}}(x,y) = \sum_k C_k P_k(x,y), \quad \sum_k C_k = 1, \qquad k \in \{\text{NE}, \text{NW}, \text{SE}, \text{SW}, \text{C}\}.
\tag{3.12}
$$

Here, P_{OPT} is the quadratic polynomial based on a nine-point stencil, centered around $I_{i,j}$, which is given by (see [12])

$$
\begin{aligned}
P_{\text{OPT}}(x,y) = {} & u^n_{i,j} + u'_{i,j}(x - x_i) + u^{\backslash}_{i,j}(y - y_j) + u^{\wedge}_{i,j}(x - x_i)(y - y_j)\\
& + \frac{1}{2}u''_{i,j}(x - x_i)^2 + \frac{1}{2}u^{\prime\prime}_{i,j}(y - y_j)^2,
\end{aligned}
\tag{3.13}
$$

where

$$
\begin{aligned}
u^n_{i,j} &= \bar{u}_{i,j} - \frac{1}{24}\left((\Delta x)^2 u''_{i,j} + (\Delta y)^2 u^{\prime\prime}_{i,j}\right),\\
u'_{i,j} &= \frac{\bar{u}^n_{i+1,j} - \bar{u}^n_{i-1,j}}{2\Delta x}, \qquad u^{\backslash}_{i,j} = \frac{\bar{u}^n_{i,j+1} - \bar{u}^n_{i,j-1}}{2\Delta y},\\
u''_{i,j} &= \frac{\bar{u}^n_{i+1,j} - 2\bar{u}^n_{i,j} + \bar{u}^n_{i-1,j}}{\Delta x^2}, \qquad u^{\prime\prime}_{i,j} = \frac{\bar{u}^n_{i,j+1} - 2\bar{u}^n_{i,j} + \bar{u}^n_{i,j-1}}{\Delta y^2},\\
u^{\wedge}_{i,j} &= \frac{\bar{u}^n_{i+1,j+1} + \bar{u}^n_{i-1,j-1} - \bar{u}^n_{i+1,j-1} - \bar{u}^n_{i-1,j+1}}{4\Delta x \Delta y}.
\end{aligned}
\tag{3.14}
$$

Unlike the one-dimensional case, not every symmetric selection of the constants C_ks will provide a third-order reconstruction for the quarter cell-averages. Here, a straightforward computation shows that in order to satisfy the accuracy requirements, we must take $C_{\rm NE} = C_{\rm NW} = C_{\rm SW} = C_{\rm SE} = 1/8$. Hence, $C_{\rm C} = 1/2$, and (3.12) implies

$$\begin{aligned} P_{\rm C}(x,y) &= 2P_{\rm OPT}(x,y) - \frac{1}{4}\Big[P_{\rm NE}(x,y) + P_{\rm NW}(x,y) + P_{\rm SW}(x,y) + P_{\rm SE}(x,y)\Big] \\ &= u^n_{i,j} + u'_{i,j}(x-x_i) + u^{\backprime}_{i,j}(y-y_j) + 2u^{\wedge}_{i,j}(x-x_i)(y-y_j) \\ &\quad + u''_{i,j}(x-x_i)^2 + u^{\backprime\backprime}_{i,j}(y-y_j)^2, \end{aligned} \tag{3.15}$$

where for this formula

$$u^n_{i,j} = \bar{u}^n_{i,j} - \frac{1}{12}\Big[(\Delta x)^2 u''_{i,j} + (\Delta y)^2 u^{\backprime\backprime}_{i,j}\Big].$$

All that remains is to determine the weights $w^{i,j}_k$ in (3.10). Once again, we write

$$w^{i,j}_k = \frac{\alpha^{i,j}_k}{\sum_l \alpha^{i,j}_l}, \qquad \alpha^{i,j}_k = \frac{C^{i,j}_k}{(\varepsilon + IS^{i,j}_k)^p}, \qquad k,l \in \{{\rm NE}, {\rm NW}, {\rm SE}, {\rm SW}, {\rm C}\}.$$

The constants C_k are the same constants that were used to reconstruct the centered parabola in (3.12). The constants ε and p play the same role as in the one-dimensional case. At this point, to simplify the notations we assume that the mesh spacings are equal in the x and y directions, i.e., $\Delta x = \Delta y = h$. We can then follow [15] and define the smoothness indicators, $IS^{i,j}_k$, as

(3.16)

$$IS^{i,j}_k = \sum_{|\alpha|=1,2} \int_{x_{i-h/2}}^{x_{i+h/2}} \int_{y_{j-h/2}}^{y_{j+h/2}} h^{2(|\alpha|-1)} (D^\alpha P_k)^2, \qquad k \in \{{\rm NE}, {\rm NW}, {\rm SE}, {\rm SW}, {\rm C}\}.$$

If $\Delta x \neq \Delta y$, then only a trivial enhancement to (3.16) is required. For the four one-sided linear reconstructions, which can all be written as

$$P_k = \hat{u}_k + \hat{u}'_k(x-x_i) + \hat{u}^{\backprime}_k(y-y_j), \qquad k \in \{{\rm NE}, {\rm NW}, {\rm SE}, {\rm SW}\},$$

with suitable reconstructed point-values and first derivatives, a direct computation of (3.16) results with

$$IS_k = h^2[(\hat{u}')^2 + (\hat{u}^{\backprime})^2]. \tag{3.17}$$

The centered smoothness indicator, $IS_{\rm C}$, which corresponds to the centered quadratic reconstruction, $P_c(x,y)$, (3.15), is given by

$$IS_{\rm C} = h^2\left[(u')^2 + (u^{\backprime})^2\right] + \frac{h^4}{3}\left[13(u'')^2 + 14(u^{\wedge})^2 + 13(u^{\backprime\backprime})^2\right]$$

(the discrete derivatives are given by (3.14)).

3.3. A note on systems. Almost nothing changes when turning to deal with systems. The reconstruction that was described in the previous sections directly transforms to systems and is performed componentwise. The only relatively subtle issue is the computation of the smoothness indicators.

In our previous work [13], we have suggested several approaches for the computation of the smoothness indicators. Three different options were suggested: the first is to allow every component to have a strictly individual behavior, namely, to allow a different stencil with different smoothness indicators to each component. In the second approach, we designed *global smoothness indicators* such as to force every component to adjust even when the discontinuity is in a different component. The last approach was to use external information about the system. For example, in the Euler equations of gas dynamics, one expects both shocks and contact discontinuities to occur in the density. Hence, all stencils can be tuned according to this component.

Our results in [13] showed that the best approach is the one which was based on the *global smoothness indicators*. It requires no additional information on the system, it produced less oscillatory results compared with the individual smoothness indicators for each component, and it was the simplest to implement.

In the one-dimensional case, e.g., the global smoothness indicators are given by (compare with (3.8))

$$IS_k = \frac{1}{d}\sum_{r=1}^{d}\frac{1}{\|\bar{u}_r\|_2}\left(\sum_{l=1}^{2}\int_{x_{j-1/2}}^{x_{j+1/2}} h^{2l-1}\left(P_{k,r}^{(l)}\right)^2\,dx\right), \quad k\in\{\text{L},\text{C},\text{R}\}. \tag{3.18}$$

Here the kth polynomial in the rth component is denoted by $P_{k,r}$, and d is the number of equations. The scaling factor $\|\bar{u}_r\|_2$ is defined as the L^2 norm of the cell-averages of the rth component of u,

$$\|\bar{u}_r\|_2 = \left(\sum_{\text{all } j}|\bar{u}_{j,r}|^2 h\right)^{1/2}.$$

The numerical examples performed for systems, appearing in section 4, were carried out with the global smoothness indicators.

4. Examples. We present numerical tests in one and two space dimensions, in which we demonstrate the accuracy, robustness, and high-resolution properties of our new method.

Following our previous works [2], [13], [15], and [16], in all our numerical examples we integrate in time using a two-step RK method with natural continuous extension, which was presented in [26], with a fixed time step.

The time step is determined by imposing that the Courant number is a given fraction of the maximum Courant number determined by linear stability analysis. The Courant number is defined by

$$C = \frac{\lambda}{\max_j \rho_j},$$

where ρ_j denotes the spectral radius of the matrix $f'(u_j)$ computed on the initial condition, and $\lambda = \Delta t/\Delta x$ is the mesh ratio.

The linear stability analysis carried out in [3] yields a Courant number $C = 3/7$ for the one-dimensional case. We remark that the stencil used in [3] was different than

the one that we use, and therefore linear stability should be repeated for the compact scheme in order to obtain a sharp estimate on the maximum Courant number.

Example 1: *Accuracy tests.* Our first example checks the accuracy of our new method in several one- and two-dimensional test cases. In all of the one-dimensional tables, the norms of the errors are given by

$$\begin{aligned} L^1 - \text{error}: &\quad ||\text{Error}||_1 = \textstyle\sum_{j=1}^{N} |u(x_j, t^n) - u_j^n| h, \\ L^\infty - \text{error}: &\quad ||\text{Error}||_\infty = \max_{1 \le j \le N} |u(x_j, t^n) - u_j^n|. \end{aligned}$$

Analogous expressions hold for the two-dimensional norms.

1. *Linear advection:* This test estimates the convergence rate at large times. We solve $u_t + u_x = 0$, subject to the initial data $u(x, t=0) = \sin(\pi x)$ and to periodic boundary conditions on $[-1, 1]$. The integration time was taken as $T = 10$.
   In Table 4.1, we show the results obtained for this test problem with $\epsilon = 10^{-2}$. We clearly see a third-order convergence rate in both L^1 and L^∞ norms. We also show in Table 4.2 the results obtained for the same example with $\epsilon = 10^{-4}$. Note that in order to observe high-order accuracy, a higher value of ϵ is required compared to the one used by the purely parabolic reconstruction (see [7] and [13]).
   Compared with Table 4.1, the errors are larger, and the convergence rate is not as regular as before. This is mainly due to the fact that for a very small value of ε, condition (i) is not satisfied until the grid spacing Δx becomes very small. Note that, at variance with CWENO based on piecewise parabolic reconstruction, here a small deviation of the weights from their optimal constant value results in a degradation of the order of accuracy of the reconstruction.
2. *Linear advection with oscillatory data:* This test is used to detect deteriorations of accuracy due to oscillations in the parameters that control the selection of the stencil (for details see [2] and the references therein). Once again, the equation is $u_t + u_x = 0$, subject to the oscillatory initial data $u(x, t=0) = \sin^4(\pi x)$ and to periodic boundary conditions on $[-1, 1]$. Here, the integration time is taken as $T = 1$ and $\epsilon = 10^{-4}$.
   The results of this test are displayed in Table 4.3 and confirm the third-order accuracy of the method with no deteriorations in its accuracy.
3. *Burgers equation:* We solve the Burgers equation $u_t + (0.5u^2)_x = 0$, subject to the initial data $u(x, t=0) = 1 + 0.5\sin(\pi x)$ and to periodic boundary conditions on $[-1, 1]$. The integration time is $T = .33$, and $\epsilon = 10^{-4}$. Here, a shock develops at $T = 2/\pi$. Note that here the maximum speed of propagation is $f'(u) = 3/2$. Thus we use $\lambda = .66 * 3/7 \simeq 2/3\lambda_{\max}$.
   Table 4.4 shows the results we obtained which verify the third-order accuracy of the method also for nonlinear problems.
   The result of the computation for time $T = 1.5$ is shown in Figure 4.1.
4. *Two-dimensional linear advection:* Finally, we implemented our method for the two-dimensional linear advection problem, $u_t + u_x + u_y = 0$. The initial condition is taken as $u(x, t=0) = \sin^2(\pi x)\sin^2(\pi y)$, and we impose periodic boundary conditions on $[0, 1]^2$. The errors and the estimated convergence rate are computed at time $T = 1$.
   In Table 4.5 we present the results obtained when the weights are taken as constants (3.12). Table 4.6 shows the results obtained with the fully nonlinear

TABLE 4.1
Linear advection, $T = 10$, $u_0(x) = \sin(\pi x)$, $\epsilon = 10^{-2}$; *Courant number* $C = 0.9C_{max}$, $C_{max} = 3/7$.

N	L^1 error	L^1 order	L^∞ error	L^∞ order
20	0.1423	-	0.1484	-
40	0.1308E-01	3.44	0.1708E-01	3.12
80	0.7054E-03	4.21	0.1071E-02	4.00
160	0.7517E-04	3.23	0.7823E-04	3.78
320	0.9391E-05	3.00	0.7977E-05	3.29
640	0.1174E-05	3.00	0.9406E-06	3.08
1280	0.1467E-06	3.00	0.1158E-06	3.02

TABLE 4.2
Linear advection, $T = 10$, $u_0(x) = \sin(\pi x)$, $\epsilon = 10^{-4}$; *Courant number* $C = 0.9C_{max}$, $C_{max} = 3/7$.

N	L^1 error	L^1 order	L^∞ error	L^∞ order
20	0.2275	-	0.2499	-
40	0.7798E-01	1.54	0.8705E-01	1.52
80	0.8786E-02	3.15	0.1756E-01	2.31
160	0.6024E-03	3.87	0.1603E-02	3.45
320	0.3031E-04	4.31	0.6726E-04	4.58
640	0.1635E-05	4.21	0.2803E-05	4.58
1280	0.1467E-06	3.48	0.1740E-06	4.01

TABLE 4.3
Linear advection, $T = 1$, $u_0(x) = \sin^4(\pi x)$, $\epsilon = 10^{-4}$; *Courant number* $C = 0.9C_{max}$, $C_{max} = 3/7$.

N	L^1 error	L^1 order	L^∞ error	L^∞ order
20	0.1368	-	0.2026	-
40	0.4685E-01	1.55	0.8759E-01	1.21
80	0.1247E-01	1.91	0.3320E-01	1.40
160	0.1326E-02	3.23	0.6239E-02	2.41
320	0.7851E-04	4.08	0.3951E-03	3.98
640	0.5383E-05	3.87	0.1585E-04	4.64
1280	0.5092E-06	3.40	0.8398E-06	4.24

TABLE 4.4
Burgers equation, $T = .33$, $u_0(x) = 1 + 0.5\sin(\pi x)$, $\epsilon = 10^{-4}$, $\lambda = 0.66 * 3/7$, *corresponding to a Courant number* $C = 0.99C_{max}$.

N	L^1 error	L^1 order	L^∞ error	L^∞ order
20	0.1874E-01	-	0.3065E-01	-
40	0.3513E-02	2.42	0.1042E-01	1.56
80	0.2675E-03	3.72	0.9609E-03	3.44
160	0.1567E-04	4.09	0.4008E-04	4.58
320	0.1175E-05	3.74	0.3172E-05	3.66
640	0.1309E-06	3.17	0.4482E-06	2.82
1280	0.1660E-07	2.98	0.7042E-07	2.67

scheme, where the weights of the reconstruction include also the oscillatory indicators. With constant weights our method is third-order as expected, while with nonconstant weights, there seems to be a better convergence rate. A careful study of the tables shows, however, that this better convergence rate is mainly due to larger errors on the coarser grids.

Example 2*: One-dimensional systems—Euler equations of gas dynamics.* Next,

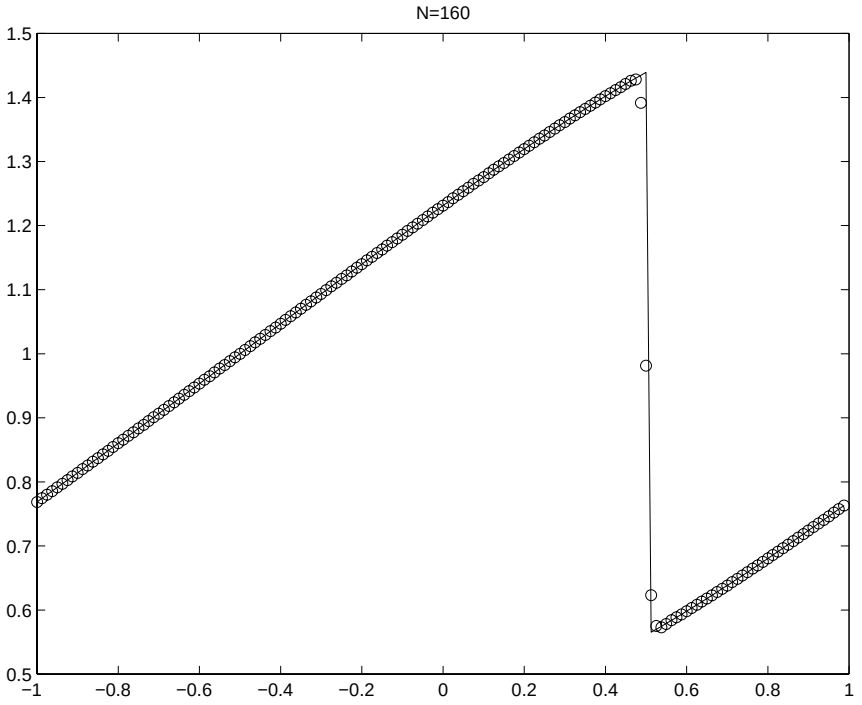


FIG. 4.1. *Burgers equation. Solution at time* $T = 1.5$. $\lambda = 0.66 * 3/7$, $N = 160$, $\epsilon = 10^{-4}$.

TABLE 4.5
Two-dimensional linear advection, constant weights; $T = 1$, $\lambda = 0.425$, $u_0(x) = \sin^2(\pi x)\sin^2(\pi y)$.

N	L^1 error	L^1 order	L^∞ error	L^∞ order
10	3.696E-02	-	1.252E-01	-
20	4.964E-03	2.90	1.767E-02	2.83
40	6.304E-04	2.98	2.264E-03	2.96
80	7.902E-05	3.00	2.842E-04	2.99
160	9.880E-06	3.00	3.555E-05	3.00

TABLE 4.6
Two-dimensional linear advection, non-constant weights; $T = 1$, $\lambda = 0.425$, $u_0(x) = \sin^2(\pi x)\sin^2(\pi y)$, $\epsilon = 10^{-4}$.

N	L^1 error	L^1 order	L^∞ error	L^∞ order
10	1.154E-01	-	4.024E-01	-
20	3.273E-02	1.82	1.675E-01	1.26
40	7.933E-03	2.04	5.413E-02	1.63
80	6.107E-04	3.70	9.097E-03	2.57
160	2.662E-05	4.52	4.744E-04	4.26

we solve the Euler equations of gas dynamics,

$$\frac{\partial}{\partial t}\begin{pmatrix}\rho\\ m\\ E\end{pmatrix} + \frac{\partial}{\partial x}\begin{pmatrix} m\\ \rho u^2 + p\\ u(E+p)\end{pmatrix} = 0, \qquad p = (\gamma - 1)\cdot\left(E - \frac{\rho}{2}u^2\right),$$

where the variables ρ, u, $m = \rho u$, p, and E denote the density, velocity, momentum,

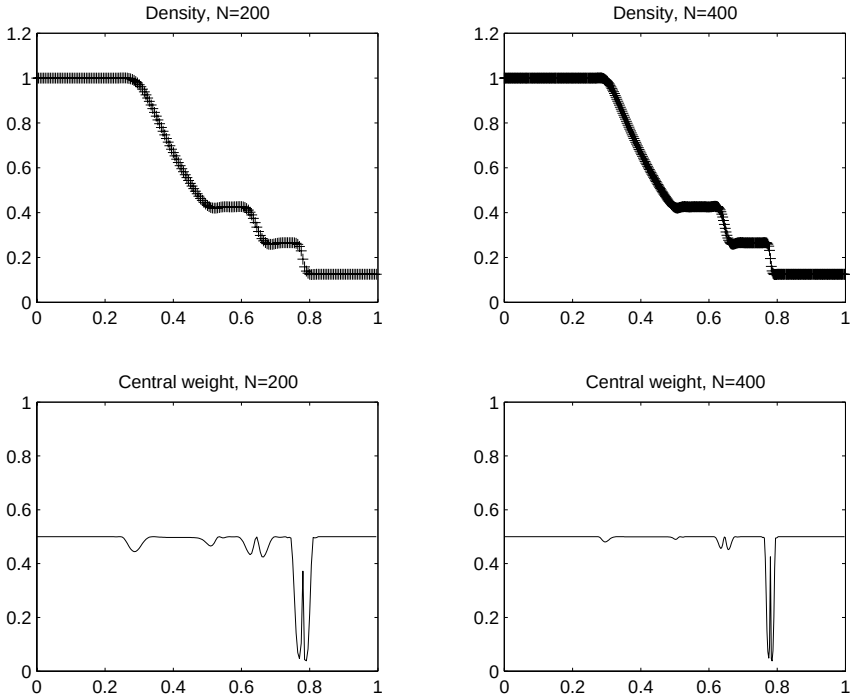


FIG. 4.2. *Euler equations of gas dynamics—Sod's initial data,* $\lambda = 0.1$, $T = 0.16$.

pressure, and total energy, respectively. We use two sets of initial data:

1. Shock tube problem with Sod's initial data [24],

$$\begin{cases} (\rho_l, m_l, E_l) = (1, 0, 2.5), & x < 0.5, \\ (\rho_r, m_r, E_r) = (0.125, 0, 0.25), & x > 0.5. \end{cases}$$

2. Shock tube problem with Lax's initial data [9],

$$\begin{cases} (\rho_l, m_l, E_l) = (0.445, 0.311, 8.928), & x < 0.5, \\ (\rho_r, m_r, E_r) = (0.5, 0, 1.4275), & x > 0.5. \end{cases}$$

We integrate the equations up to time $T = 0.16$, using $\epsilon = 10^{-4}$. In Figure 4.2 we show the density components for Sod's initial data, and in Figure 4.3 we show the equivalent plot for Lax's initial data. In both figures we also present the weight of the central stencil at the final time. All the results are given for 200 and 400 grid points. Following [20], we pick $\lambda = .1$. Note that the maximum characteristic speed for Sod's problem is roughly 2.5, while for the Lax problem the maximum propagation speed is $\simeq 5$. Since our scheme has a Courant number no larger than .5, we see that $\lambda = .1$ is actually the maximum value compatible with stability.

It is interesting to compare the behavior of the central weight of the new method to the behavior of the central weight of the original CWENO method [13]. Here, the weights are much smoother compared with the weights in [13]. The accuracy and stability properties of the method can be related to the smoothness of the nonlinear weights involved (see [2]).

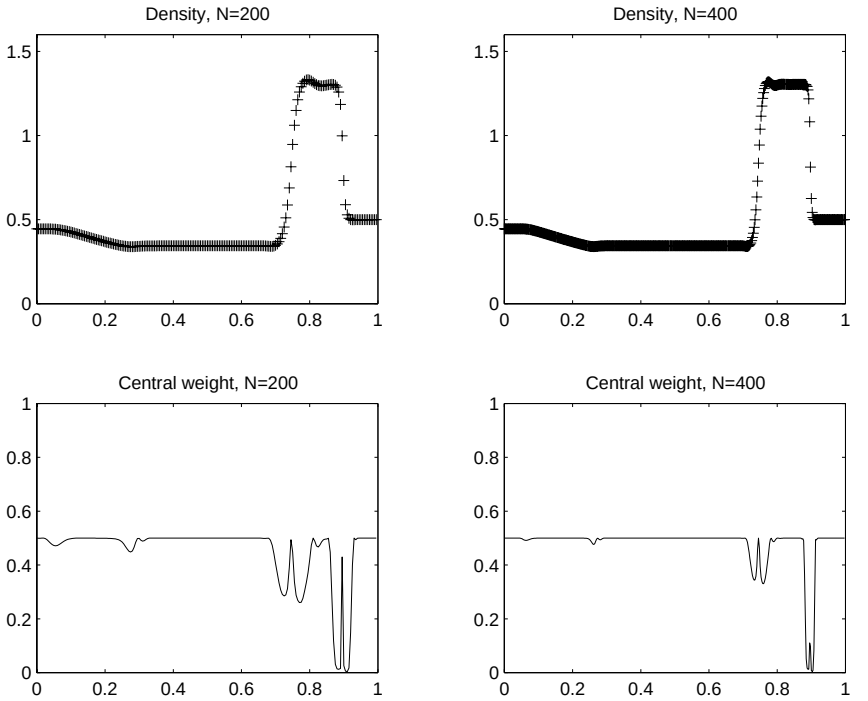


FIG. 4.3. *Euler equations of gas dynamics—Lax's initial data,* $\lambda = 0.1$, $T = 0.16$.

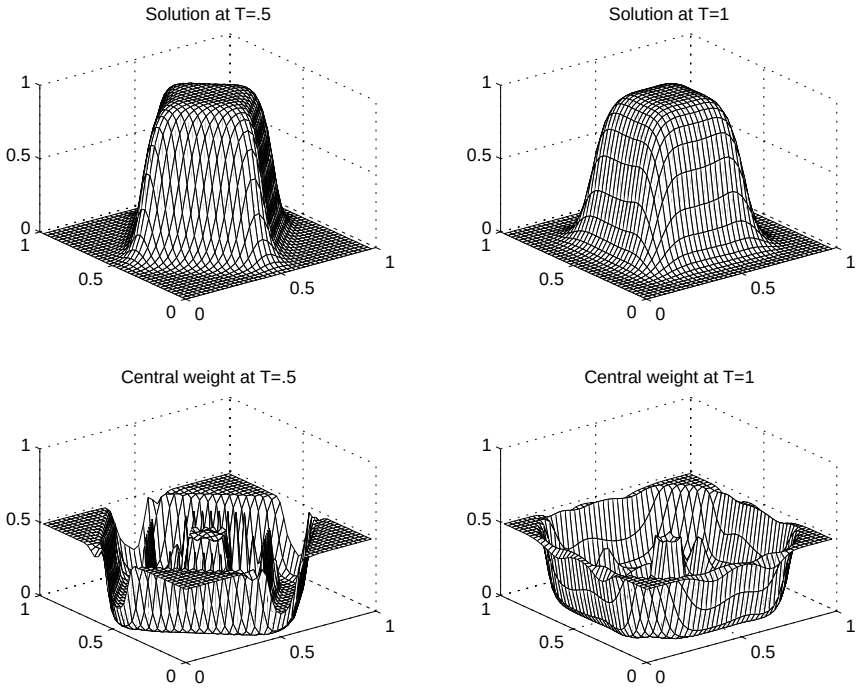


FIG. 4.4. *Linear rotation,* $\lambda = 0.425$, $N = 40$.

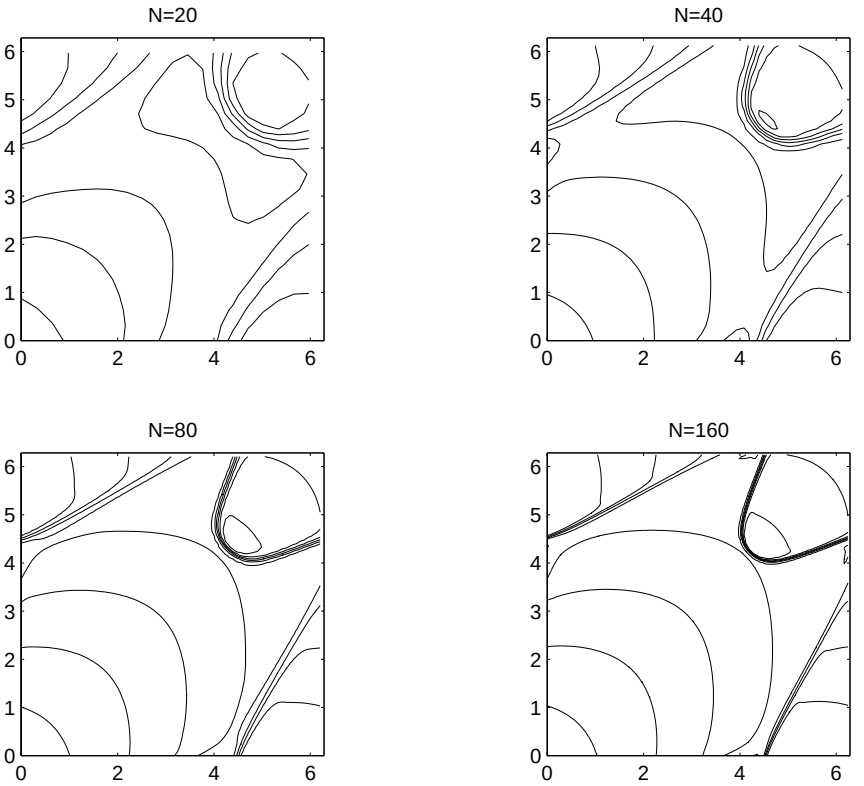


FIG. 4.5. *Two-dimensional Burgers equation,* $\lambda = 0.425$, $T = 1.5$.

Example 3*: Two-dimensional problems.*

1. *Linear rotation:* Following [15], we consider a linear rotation of a square patch on $[0,1]^2$, with initial condition $u_0(x,y) = 1$ for $\{|x - 1/2| \leq 1/2\} \times \{|y - 1/2| \leq 1/2\}$ and zero elsewhere. In Figure 4.4 we display the solution after a rotation of $\pi/4$ and of $\pi/2$. As before, the value of ϵ is taken as 10^{-4}. There are no spurious oscillations. We also show the corresponding central weight. As expected, this weight is zero in regions where the solution has steep gradients, and that is exactly the property that prevents spurious oscillations from developing. Even though we are dealing with linear waves, the resulting resolution is relatively good. Due to the compactness of the stencil, when the slopes are not sharp, they are not identified as discontinuities. This can be observed in the plots of the central weight, which returns to its constant value (1/2) on the slope.
2. *Two-dimensional—Burgers equation:* We end by solving the two-dimensional Burgers equation, $u_t + (u^2/2)_x + (u^2/2)_y = 0$, subject to the initial data $u(x, t = 0) = \sin^2(\pi x)\sin^2(\pi y)$ and periodic boundary conditions on $[0,1]^2$. Here also $\epsilon = 10^{-4}$. In Figure 4.5 we present the solution obtained at time $T = 1.5$ with different mesh spacings. One can easily notice that the shocks are well resolved and there are no spurious oscillations.

REFERENCES

[1] P. Arminjon, D. Stanescu, and M.-C. Viallon, *A two-dimensional finite volume extension of the Lax-Friedrichs and Nessyahu-Tadmor schemes for compressible flow*, in Proceedings of the 6th International Symposium on Computational Fluid Dynamics, Vol. IV, M. Hafez and K. Oshima, eds., Lake Tahoe, NV, 1995, pp. 7–14.

[2] F. Bianco, G. Puppo, and G. Russo, *High order central schemes for hyperbolic systems of conservation laws*, SIAM J. Sci. Comput., 21 (1999), pp. 294–322.

[3] F. Bianco, G. Puppo, and G. Russo, *High order central schemes for hyperbolic systems of conservation laws*, Internat. Ser. Numer. Math. 129, Birkhäuser-Verlag, Basel, Switzerland, 1999, pp. 55–64.

[4] K. O. Friedrichs and P. D. Lax, *Systems of conservation equations with a convex extension*, Proc. Natl. Acad. Sci., 68 (1971), pp. 1686–1688.

[5] E. Godlewski and P.-A. Raviart, *Numerical Approximation of Hyperbolic Systems of Conservation Laws*, Springer, New York, 1996.

[6] A. Harten, B. Engquist, S. Osher, and S. Chakravarthy, *Uniformly high order accurate essentially non-oscillatory schemes* III, J. Comput. Phys., 71 (1987), pp. 231–303.

[7] G.-S. Jiang and C.-W. Shu, *Efficient implementation of weighted ENO schemes*, J. Comput. Phys., 126 (1996), pp. 202–228.

[8] G.-S. Jiang and E. Tadmor, *Nonoscillatory central schemes for multidimensional hyperbolic conservation laws*, SIAM J. Sci. Comput., 19 (1998), pp. 1892–1917.

[9] P. D. Lax, *Weak solutions of non-linear hyperbolic equations and their numerical computation*, Comm. Pure. Appl. Math., 7 (1954), pp. 159–193.

[10] B. van Leer, *Towards the ultimate conservative difference scheme,* V. *A second-order sequel to Godunov's method*, J. Comput. Phys., 32 (1979), pp. 101–136.

[11] R. J. LeVeque, *Numerical Methods for Conservation Laws*, 2nd ed., Lectures in Mathematics ETH Zürich, Birkhauser-Verlag, Basel, Switzerland, 1992.

[12] D. Levy, *A third-order* 2*D central scheme for conservation laws*, in Systèmes Hyperboliques: Nouveaux Schémas et Nouvelles Applications, Vol. I, INRIA, Roquencourt, France, 1998, pp. 489–504.

[13] D. Levy, G. Puppo, and G. Russo, *Central WENO schemes for hyperbolic systems of conservation laws*, Math. Model. Numer. Anal., 33 (1999), pp. 547–571.

[14] D. Levy, G. Puppo, and G. Russo, *On the behavior of the total variation in CWENO methods for conservation laws*, Appl. Numer. Math., 33 (2000), pp. 407–414.

[15] D. Levy, G. Puppo, and G. Russo, *A third order central WENO scheme for* 2*D conservation laws*, Appl. Numer. Math., 33 (2000), pp. 415–421.

[16] D. Levy, G. Puppo, and G. Russo, *Central Weno Schemes for Multi-Dimensional Hyperbolic Systems of Conservation Laws*, in preparation.

[17] X.-D. Liu and S. Osher, *Nonoscillatory high order accurate self-similar maximum principle satisfying shock capturing schemes* I, SIAM J. Numer. Anal., 33 (1996), pp. 760–779.

[18] X.-D. Liu, S. Osher, and T. Chan, *Weighted essentially non-oscillatory schemes*, J. Comput. Phys., 115 (1994), pp. 200–212.

[19] X.-D. Liu and E. Tadmor, *Third order nonoscillatory central scheme for hyperbolic conservation laws*, Numer. Math., 79 (1998), pp. 397–425.

[20] H. Nessyahu and E. Tadmor, *Non-oscillatory central differencing for hyperbolic conservation laws*, J. Comput. Phys., 87 (1990), pp. 408–463.

[21] R. Sanders and A. Weiser, *A high resolution staggered mesh approach for nonlinear hyperbolic systems of conservation laws*, J. Comput. Phys., 101 (1992), pp. 314–329.

[22] C.-W. Shu, *Essentially non-oscillatory and weighted essentially non-oscillatory schemes for hyperbolic conservation laws*, in Advanced Numerical Approximation of Nonlinear Hyperbolic Equations, A. Quarteroni, ed., Lecture Notes in Math. 1697, Springer, Berlin, 1998.

[23] C.-W. Shu and S. Osher, *Efficient implementation of essentially non-oscillatory shock-capturing schemes,* II, J. Comput. Phys., 83 (1989), pp. 32–78.

[24] G. Sod, *A survey of several finite difference methods for systems of nonlinear hyperbolic conservation laws*, J. Comput. Phys., 22 (1978), pp. 1–31.

[25] E. Tadmor, *Approximate solutions of nonlinear conservation laws*, in Advanced Numerical Approximation of Nonlinear Hyperbolic Equations, A. Quarteroni, ed., Lecture Notes in Math. 1697, Springer, Berlin, 1998.

[26] M. Zennaro, *Natural continuous extensions of Runge–Kutta methods*, Math. Comp., 46 (1986), pp. 119–133.

SIAM J. SCI. COMPUT.
Vol. 22, No. 2, pp. 673–703
© 2000 Society for Industrial and Applied Mathematics

CONSTRUCTION AND ANALYSIS OF OPTIMAL HIERARCHIC MODELS OF BOUNDARY VALUE PROBLEMS ON THIN CIRCULAR AND SPHERICAL GEOMETRIES*

MARK AINSWORTH† AND MARK ARNOLD‡

Abstract. A sequence of optimal hierarchic models is derived for stationary heat conduction on the following laminated domains: a flat plate, a circular arch, and a spherical shell. An a priori bound is presented, which is evaluated explicitly in the homogeneous case on each domain. Computational examples are given, which illustrate the theoretical results. Furthermore, in the case of a spherical shell, the hierarchic models are compared numerically to models obtained using polynomial approximation in the radial direction, and it is found that the hierarchic models are both more accurate than the polynomial ones and provide more robust convergence with respect to the shell thickness.

Key words. hierarchic models, thin domains, dimensional reduction

AMS subject classifications. Primary 65N35; Secondary 41A60, 73V05

PII. S1064827599356274

1. Introduction.

Dimensional reduction of boundary value problems posed on thin domains has a long history in the mathematical modelling of physical problems, as illustrated, for instance, in [12, 15]. Traditionally, the derivation of the models is based on physical intuition and practical experience, with little attention paid to bounding the accuracy of the resulting models. Hierarchic modelling [27, 28] provides an effective and systematic framework for modelling problems on thin domains by reducing the dimension of the resulting boundary value problem. Original work was carried out in this area by Vogelius and Babuska [27, 28] for stationary heat conduction on flat, laminated plates. They derived a sequence of models, each capturing more modes of variation in the thickness of the plate of the true solution, designed to be optimally accurate when the thickness of the plate was small. In [27], a priori error bounds were derived giving the order of accuracy as the thickness of the plate tends to zero. This work has been subsequently applied to several areas; in particular, hierarchic models of thin domains have been derived for nonlinear problems [13, 14], elasticity [8, 22], Stokes flow [6], and Helmholtz equations [5, 17, 18]. For an overview of hierarchic modelling of thin domains, see [22] and the references therein.

The treatment of flat plates is well understood thanks to the efforts of [27, 28]. One notable feature of this work is the realization that the selection of the functions to be used in constructing the dimensionally reduced models has a significant impact on the accuracy of the reduced model. However, for curved geometries such as circular arches and spherical shells, most practitioners have based the models on a family of polynomial functions in the radial direction. In particular, we mention the work of Cho and Oden [7, 8] in the context of linear elastic shells, Shen [23, 24] for Helmholtz's equation on polar and cylindrical geometries, and Surana and Abusaleh [25] and

*Received by the editors May 18, 1999; accepted for publication (in revised form) January 27, 2000; published electronically August 9, 2000.
http://www.siam.org/journals/sisc/22-2/35627.html
†Mathematics Department, Strathclyde University, Livingstone Tower, 26 Richmond St., Glasgow, G1 1XH, Scotland (M.Ainsworth@strath.ac.uk).
‡Epidemiology Department, Veterinary Laboratory Agency, New Haw, Addlestone, Surrey, UK (M.Arnold@vla.maff.gsi.gov.uk).This author was supported through an EPSRC research studentship.

Surana and Bose [26]. The results presented in these sources show the efficiency of basing models on the use of polynomials. Here, we adopt a different approach, more akin to the original investigations of Vogelius and Babuska [27, 28] and attempt to obtain the optimal family of dimensionally reduced models. It is found that the optimal family of functions depends on the thickness of the domain in a nontrivial manner. This renders the derivation of a priori error bounds more difficult since the dependence of the optimal family on the thickness means that certain constants now depend on the thickness. An approach is presented that allows the constants to be computed *explicitly*, and furthermore, the analysis makes no assumptions on the thickness being small. A by-product of the analysis is sharper bounds for the flat plates already analyzed in [27, 28], along with sharp new results for circular arches and spherical shells that have not been studied hitherto.

Numerical examples are presented, supporting and verifying the theory. It is also shown that the computational cost associated with using the optimal models is essentially identical to the cost of simply using polynomials. However, numerical evidence is presented showing that the accuracy of the optimal models can be orders of magnitude superior to the simple polynomial based models. The numerical results also indicate that the optimal family is significantly more robust in the case when the domains are relatively thick.

1.1. Dimensional reduction on a spherical shell. Consider the steady state temperature distribution $u(r,\boldsymbol{\theta})$, where $\boldsymbol{\theta}=(\theta,\phi)$, in a thin spherical cap of average radius $R>0$:

$$\Omega=\{(r,\boldsymbol{\theta}):(1-\epsilon)R<r<(1+\epsilon)R,\boldsymbol{\theta}\in\omega\}\,,$$

where ω is a simply-connected portion of the surface of a sphere, with Lipschitz boundary $\partial\omega$. The physical principles determining the temperature distribution are as follows.

1. Conservation of energy:

$$\boldsymbol{\nabla}\cdot\boldsymbol{q}=0 \text{ in } \Omega, \tag{1}$$

where $\boldsymbol{q}=\boldsymbol{q}(r,\boldsymbol{\theta})$ is the heat flux.

2. Constitutive equation (Fourier's law):

$$\boldsymbol{q}=-\mathcal{K}\boldsymbol{\nabla}u \text{ in } \Omega, \tag{2}$$

where $\mathcal{K}$ is the thermal conductivity tensor.

3. Boundary conditions.

(a) Prescribed heat fluxes on top and bottom surfaces:

$$\pm\hat{\boldsymbol{e}}_r\cdot\boldsymbol{q}=-f^{\pm}(\boldsymbol{\theta}) \text{ on } \Omega_{\pm}, \tag{3}$$

where

$$\Omega_{\pm}=\{(r,\boldsymbol{\theta})\in\Omega:r=(1\pm\epsilon)R\}$$

and $\hat{\boldsymbol{e}}_r$ is the unit vector in the radial direction.

(b) Prescribed temperature:

$$u=0 \text{ on } \Gamma, \tag{4}$$

where Γ is the lateral boundary

$$\Gamma=\{(r,\boldsymbol{\theta}):(1-\epsilon)R<r<(1+\epsilon)R,\boldsymbol{\theta}\in\partial\omega\}\,.$$

For the geometry of the sphere, the following relations assume the particular form.

1. Conservation of energy:

$$\frac{1}{r^2}\frac{\partial}{\partial r}(r^2 q_r) + \frac{1}{r}\bar{\boldsymbol{\nabla}}' \cdot \boldsymbol{q}' = 0, \tag{5}$$

where the heat flux is written as

$$\boldsymbol{q} = \boldsymbol{q}' + q_r \hat{\boldsymbol{e}}_r; \boldsymbol{q}' = q_\theta \hat{\boldsymbol{e}}_\theta + q_\phi \hat{\boldsymbol{e}}_\phi$$

and the divergence operator on the surface of the unit sphere is given by

$$\bar{\boldsymbol{\nabla}}' \cdot \boldsymbol{q}' = \frac{1}{\sin\theta}\frac{\partial}{\partial\theta}\sin\theta q_\theta + \frac{1}{\sin\theta}\frac{\partial}{\partial\phi}q_\phi. \tag{6}$$

2. Constitutive relation: The conductivity tensor is assumed to have the form

$$\boldsymbol{q}' = -k_1\left(\frac{r-R}{\epsilon R}\right)\mathcal{K}'(\boldsymbol{\theta})\frac{1}{r}\bar{\boldsymbol{\nabla}}' u, \tag{7}$$

$$q_r = -k_2\left(\frac{r-R}{\epsilon R}\right)\frac{\partial u}{\partial r}, \tag{8}$$

where

$$\bar{\boldsymbol{\nabla}}' u = \frac{\partial u}{\partial\theta}\hat{\boldsymbol{e}}_\theta + \frac{1}{\sin\theta}\frac{\partial u}{\partial\phi}\hat{\boldsymbol{e}}_\phi. \tag{9}$$

The determination of the true solution of the boundary value problem represented by (1)–(4) generally entails a considerable degree of complexity, especially if the domain is of laminated construction, meaning that the coefficients k_1, k_2 are only piecewise continuous. Typically, one must resort to an approximate numerical method to obtain a sufficiently accurate solution. However, if the domain is very thin ($\epsilon \ll 1$), then one experiences difficulties even with a numerical approach owing to the anisotropy of the domain.

A standard approach to such problems posed on thin domains consists of replacing the full (three-)dimensional problem by a dimensionally reduced model posed in fewer (two) dimensions over the mid-surface ω. The dimensionally reduced model is generally an elliptic system (rather than a scalar problem) with the unknowns corresponding to weighted averages of the true solution u through the thickness. The equations satisfied by the averages derive from the physical model (1)–(4) as follows.

Let $\psi \in L^\infty(-1,1)$ be a given function to be selected later and consider a weighted average of the solution u_ψ defined by

$$u_\psi(\boldsymbol{\theta}) = \int_{(1-\epsilon)R}^{(1+\epsilon)R} \psi\left(\frac{r-R}{\epsilon R}\right) k_1\left(\frac{r-R}{\epsilon R}\right) u(r,\boldsymbol{\theta})\, dr.$$

It then follows that

$$\mathcal{K}'(\boldsymbol{\theta})\bar{\boldsymbol{\nabla}}' u_\psi(\boldsymbol{\theta}) = \mathcal{K}'(\boldsymbol{\theta})\int_{(1-\epsilon)R}^{(1+\epsilon)R} \psi\left(\frac{r-R}{\epsilon R}\right) k_1\left(\frac{r-R}{\epsilon R}\right)\bar{\boldsymbol{\nabla}}' u\, dr$$

and applying the constitutive relation (7) leads to

$$\mathcal{K}'(\boldsymbol{\theta})\bar{\boldsymbol{\nabla}}' u_\psi(\boldsymbol{\theta}) = -\int_{(1-\epsilon)R}^{(1+\epsilon)R} \psi\left(\frac{r-R}{\epsilon R}\right)\boldsymbol{q}'(r,\boldsymbol{\theta}) r dr.$$

Consequently,

$$-\bar{\boldsymbol{\nabla}}' \cdot \left(\mathcal{K}'(\boldsymbol{\theta}) \bar{\boldsymbol{\nabla}}' u_\psi(\boldsymbol{\theta}) \right) = \int_{(1-\epsilon)R}^{(1+\epsilon)R} \psi\left(\frac{r-R}{\epsilon R} \right) \bar{\boldsymbol{\nabla}}' \cdot \boldsymbol{q}'(r, \boldsymbol{\theta}) r dr$$

and by conservation of energy (5), one obtains

$$\bar{\boldsymbol{\nabla}}' \cdot \left(\mathcal{K}'(\boldsymbol{\theta}) \bar{\boldsymbol{\nabla}}' u_\psi(\boldsymbol{\theta}) \right) = \int_{(1-\epsilon)R}^{(1+\epsilon)R} \psi\left(\frac{r-R}{\epsilon R} \right) \frac{\partial}{\partial r}(r^2 q_r) dr.$$

Integrating by parts and inserting the boundary conditions (3) reveals that

$$\begin{aligned} -\bar{\boldsymbol{\nabla}}' \cdot \left(\mathcal{K}'(\boldsymbol{\theta}) \bar{\boldsymbol{\nabla}}' u_\psi(\boldsymbol{\theta}) \right) &= (1+\epsilon)^2 R^2 \psi(1) f^+(\boldsymbol{\theta}) + (1-\epsilon)^2 R^2 \psi(-1) f^-(\boldsymbol{\theta}) \\ &\quad + \frac{1}{\epsilon R} \int_{(1-\epsilon)R}^{(1+\epsilon)R} \psi'\left(\frac{r-R}{\epsilon R} \right) r^2 q_r dr, \end{aligned}$$

where $\psi' = d\psi(s)/ds$. The constitutive relation (8) allows the final term to be rewritten as

$$-\frac{1}{\epsilon R} \int_{(1-\epsilon)R}^{(1+\epsilon)R} \psi'\left(\frac{r-R}{\epsilon R} \right) k_2\left(\frac{r-R}{\epsilon R} \right) \frac{\partial u}{\partial r} r^2 dr.$$

In conclusion, we have shown that the average u_ψ satisfies

$$\begin{aligned} -\bar{\boldsymbol{\nabla}}' \cdot \left(\mathcal{K}' \bar{\boldsymbol{\nabla}}' u_\psi(\boldsymbol{\theta}) \right) + \frac{1}{\epsilon R} \int_{(1-\epsilon)R}^{(1+\epsilon)R} \psi'\left(\frac{r-R}{\epsilon R} \right) k_2\left(\frac{r-R}{\epsilon R} \right) \frac{\partial u}{\partial r} r^2 dr \\ = R \left[\psi(1) f_\epsilon^+(\boldsymbol{\theta}) + \psi(-1) f_\epsilon^-(\boldsymbol{\theta}) \right], \end{aligned} \tag{10}$$

where

$$f_\epsilon^\pm = R(1 \pm \epsilon)^2 f^\pm.$$

Obviously, the boundary condition (4) implies that

$$u_\psi = 0 \text{ on } \partial\omega. \tag{11}$$

1.2. Simplest reduced model. The simplest dimensionally reduced model consists of selecting ψ to be a constant function, e.g., $\psi \equiv 1$. In this case, (10) reduces to the single scalar equation for $\alpha_0(\boldsymbol{\theta})$:

$$-\bar{\boldsymbol{\nabla}}' \cdot \left(\mathcal{K}' \bar{\boldsymbol{\nabla}}' \alpha_0 \right) = R(f_\epsilon^+(\boldsymbol{\theta}) + f_\epsilon^-(\boldsymbol{\theta})) \text{ in } \omega \tag{12}$$

subject to $\alpha_0 = 0$ on $\partial\omega$, where α_0 denotes the average

$$\alpha_0(\boldsymbol{\theta}) = u_\psi(\boldsymbol{\theta}) = \int_{(1-\epsilon)R}^{(1+\epsilon)R} k_1\left(\frac{r-R}{\epsilon R} \right) u(r, \boldsymbol{\theta}) dr.$$

Obviously, one is particularly interested in the accuracy of this dimensionally reduced model approximating the original full three-dimensional model. In [3] it was shown that, for $\epsilon \ll 1$, the error $e = u - \alpha_0$ is bounded by

$$\begin{aligned} |||e|||^2 \le \frac{2}{3}\epsilon \int_\omega R \sin\theta (f^e(\boldsymbol{\theta}))^2 d\theta d\phi \\ + 2\epsilon \int_\omega R \sin\theta (f^o(\boldsymbol{\theta}))^2 d\theta d\phi, \end{aligned} \tag{13}$$

where $D_\epsilon = \sqrt{\frac{1-\epsilon}{1+\epsilon}}$,

$$f^e = \frac{1}{2}\left(f_\epsilon^+ + \sqrt{\frac{1+\epsilon}{1-\epsilon}} f_\epsilon^-\right), \quad f^o = \frac{1}{2}\left(f_\epsilon^+ - \sqrt{\frac{1+\epsilon}{1-\epsilon}} f_\epsilon^-\right), \tag{14}$$

and $|||\cdot|||$ denotes the energy norm (defined later). This bound justifies the choice $\psi \equiv 1$ since as the thickness ϵ of the shell approaches 0, the accuracy of the model is improved. However, the question arises of what can be done if the accuracy for a finite ϵ is not acceptable.

1.3. Higher order models. The natural route to obtaining a more sophisticated model consists of including further weighted averages of the true solution in the reduced approximation. Let $\{\psi_l\}_{l=0}^\infty \in L_\infty(-1,1)$ be given functions to be selected later and consider a model based on the weighted averages

$$u_l(\boldsymbol{\theta}) = \int_{(1-\epsilon)R}^{(1+\epsilon)R} \psi_l\left(\frac{r-R}{\epsilon R}\right) k_1\left(\frac{r-R}{\epsilon R}\right) u(r,\boldsymbol{\theta}) dr \ , \ l = 0, \ldots, N.$$

The weighted averages satisfy (10). However, it is worthwhile to consider first how the averages are used to reconstruct an approximation u_N of the solution u of the original three-dimensional problem. The approximation u_N is sought in the form

$$u_N(r,\boldsymbol{\theta}) = \sum_{l=0}^N \alpha_l(\boldsymbol{\theta}) \psi_l\left(\frac{r-R}{\epsilon R}\right), \tag{15}$$

where the coefficients $\{\alpha_l\}_{l=0}^N$ are determined by the condition that u_N satisfy

$$\int_{(1-\epsilon)R}^{(1+\epsilon)R} (u(r,\boldsymbol{\theta}) - u_N(r,\boldsymbol{\theta})) \psi_l\left(\frac{r-R}{\epsilon R}\right) k_1\left(\frac{r-R}{\epsilon R}\right) dr = 0$$

for $l = 0, 1, \ldots, N$. Rearranging these equations leads to the conclusion that

$$\begin{aligned}
&\sum_{l=0}^N \alpha_l(\boldsymbol{\theta}) \int_{(1-\epsilon)R}^{(1+\epsilon)R} \psi_l\left(\frac{r-R}{\epsilon R}\right) \psi_m\left(\frac{r-R}{\epsilon R}\right) k_1\left(\frac{r-R}{\epsilon R}\right) dr \\
&= \int_{(1-\epsilon)R}^{(1+\epsilon)R} \psi_m\left(\frac{r-R}{\epsilon R}\right) k_1\left(\frac{r-R}{\epsilon R}\right) u(r,\boldsymbol{\theta}) dr \\
&= u_{\psi_m}(\boldsymbol{\theta})
\end{aligned}$$

for $m = 0, 1, \ldots, N$. Furthermore, by performing a change of variable, it follows that

$$u_{\psi_m}(\boldsymbol{\theta}) = \epsilon R \sum_{l=0}^N A_{ml} \alpha_l(\boldsymbol{\theta}), \tag{16}$$

where

$$\begin{aligned}
A_{ml} &= \frac{1}{\epsilon R} \int_{(1-\epsilon)R}^{(1+\epsilon)R} k_1\left(\frac{r-R}{\epsilon R}\right) \psi_l\left(\frac{r-R}{\epsilon R}\right) \psi_m\left(\frac{r-R}{\epsilon R}\right) dr \\
&= \int_{-1}^1 k_1(z) \psi_m(z) \psi_l(z) dz.
\end{aligned}$$

If the functions $\{\psi_l\}_{l=0}^N$ are linearly independent, then the matrix $\boldsymbol{A}$ is symmetric and positive definite since the coefficient k_1 is nonnegative.

Thanks to (15)–(16), it is possible to dispense with the weighted averages $\{u_{\psi_m}\}_{m=0}^N$ entirely and instead work with the coefficients $\{\alpha_l\}_{l=0}^N$ directly. In fact, inserting (15)–(16) into (10) leads to an elliptic system for the unknown functions $\{\alpha_l\}_{l=0}^N$:

$$-\epsilon R\bar{\boldsymbol{\nabla}}'\cdot\left(\mathcal{K}'\boldsymbol{A}\bar{\boldsymbol{\nabla}}'\boldsymbol{\alpha}\right)+\frac{1}{\epsilon R}\boldsymbol{B}\boldsymbol{\alpha}=\boldsymbol{f}_\epsilon, \tag{17}$$

where $\boldsymbol{B}$ is the matrix with entries

$$\begin{aligned}B_{ml}&=\int_{(1-\epsilon)R}^{(1+\epsilon)R}k_2\left(\frac{r-R}{\epsilon R}\right)\psi_m'\left(\frac{r-R}{\epsilon R}\right)\psi_l'\left(\frac{r-R}{\epsilon R}\right)r^2dr\\&=\int_{-1}^{1}k_2(z)\psi_m'(z)\psi_l'(z)(1+\epsilon z)^2dz\end{aligned}$$

and $\boldsymbol{f}_\epsilon$ is the load vector with entries

$$[\boldsymbol{f}_\epsilon]_m=R\left[\psi_m(1)f_\epsilon^+(\boldsymbol{\theta})+\psi_m(-1)f_\epsilon^-(\boldsymbol{\theta})\right].$$

The boundary condition $\boldsymbol{\alpha}=\boldsymbol{0}$ on $\partial\omega$ follows in a similar fashion from (11).

The coefficients are uniquely determined as the solution of the elliptic system (17) once the functions $\{\psi_l\}_{l=0}^N$ to be used in the weighted averages have been chosen. The purpose of the present work is as follows:

1. to determine a principle for the selection of the functions $\{\psi_l\}_{l=0}^N$ that are optimal in a certain sense,
2. to develop explicit a priori estimates for the accuracy of the resulting dimensionally reduced model.

2. Dimensional reduction. In the introduction we considered the problem given by (1)–(4) posed on a spherical cap. In our analysis we will consider more generally the physical problem given by (1)–(4) posed on a flat plate and a circular arch. The next section is devoted to presenting a general framework incorporating these cases.

2.1. General framework. The variational form of the problem (1)–(4) for each of the domains under consideration is

$$\text{find } u\in V \text{ such that } B_\epsilon(u,v)=F_\epsilon(v)\ \forall v\in V, \tag{18}$$

where V is the space

$$V=\{v\in H^1(\Omega):v=0 \text{ on } \partial\omega\},$$

and the bilinear form $B_\epsilon(u,v):V\times V\to\mathbb{R}$, and load functional $F_\epsilon(v):V\to\mathbb{R}$ are given for each domain as follows.

1. Flat plate:

$$\begin{aligned}\Omega&=\{(\boldsymbol{x}',z):\boldsymbol{x}'\in\omega,-\epsilon<z<\epsilon\},\\\Omega_\pm&=\{(\boldsymbol{x}',z):z=\pm\epsilon\},\\\Gamma&=\{(\boldsymbol{x}',z):-\epsilon<z<\epsilon,\boldsymbol{x}'\in\omega\},\\B_\epsilon(u,v)&=\int_\omega\int_{-1}^1\left\{\frac{k_2(z)}{\epsilon}\frac{\partial u}{\partial z}\frac{\partial v}{\partial z}+\epsilon k_1(z)\left(\boldsymbol{\nabla}'u\right)^T\mathcal{K}'\boldsymbol{\nabla}'v\right\}dz\,d\boldsymbol{x}',\\F_\epsilon(v)&=\int_\omega d\boldsymbol{x}'(f^+(\boldsymbol{x}')v(\boldsymbol{x}',1)+f^-(\boldsymbol{x}')v(\boldsymbol{x}',-1)),\end{aligned}$$

where

$$\bar{\boldsymbol{\nabla}}' = \frac{\partial}{\partial x}\hat{\boldsymbol{e}}_x + \frac{\partial}{\partial y}\hat{\boldsymbol{e}}_y.$$

2. Circular arch:

$$\Omega = \{(\hat{r}, \theta) : \theta \in \omega, R(1-\epsilon) < \hat{r} < R(1+\epsilon)\};$$

$\Gamma, \Omega_\pm$ are as they are for the spherical shell except that ω is now a section of a circle with end points $\partial\omega$.

$$B_\epsilon(u,v) = \int_\omega \int_{-1}^{1} \left\{ \frac{k_2(r)(1+\epsilon r)}{\epsilon} \frac{\partial u}{\partial r}\frac{\partial v}{\partial r} + \frac{k_1(r)\epsilon}{1+\epsilon r} (\bar{\boldsymbol{\nabla}}' u)^T \mathcal{K}' \bar{\boldsymbol{\nabla}}' v \right\} dr\, d\theta,$$

$$F_\epsilon(v) = \int_\omega R d\theta (f^+(\theta)(1+\epsilon)v(\theta,1) + f^-(\theta)(1-\epsilon)v(\theta,-1)),$$

$$\bar{\boldsymbol{\nabla}}' = \frac{\partial}{\partial\theta}\hat{\boldsymbol{e}}_{\boldsymbol{\theta}}.$$

3. Spherical shell:

$$\Omega = \{(\hat{r}, \theta, \phi) : (\theta, \phi) \in \omega, R(1-\epsilon) < \hat{r} < R(1+\epsilon)\},$$

$$\begin{aligned} B_\epsilon(u,v) = \int_\omega & R\sin(\theta) d\theta d\phi \int_{-1}^{1} dr \\ & \times \left\{ \frac{k_2(r)(1+\epsilon r)^2}{\epsilon} \frac{\partial u}{\partial r}\frac{\partial v}{\partial r} + \epsilon k_1(r) \left(\bar{\boldsymbol{\nabla}}' u\right)^T \mathcal{K}' \left(\bar{\boldsymbol{\nabla}}' v\right) \right\}, \\ F_\epsilon(v) = \int_\omega & R^2 \sin(\theta) d\theta d\phi \\ & \left\{ f^+(\theta,\phi)(1+\epsilon)^2 v(\theta,\phi,1) + f^-(\theta,\phi)(1-\epsilon)^2 v(\theta,\phi,-1) \right\}, \\ \bar{\boldsymbol{\nabla}}' = & \frac{\partial}{\partial\theta}\hat{\boldsymbol{e}}_{\boldsymbol{\theta}} + \frac{1}{\sin\theta}\frac{\partial}{\partial\phi}\hat{\boldsymbol{e}}_{\boldsymbol{\phi}}. \end{aligned}$$

In general, the bilinear form B_ϵ and linear functional F_ϵ are of the form

$$B_\epsilon(u,v) = \int_\omega d\boldsymbol{\theta} \int_{-1}^{1} dr \left\{ \frac{a(1+\epsilon r)}{\epsilon} \frac{\partial u}{\partial r}\frac{\partial v}{\partial r} + \epsilon b(1+\epsilon r) \left(\bar{\boldsymbol{\nabla}}' u\right)^T \mathcal{K}' \bar{\boldsymbol{\nabla}}' v \right\} \tag{19}$$

and

$$F_\epsilon(v) = \int_\omega (f_\epsilon^+(\boldsymbol{\theta}) v(\boldsymbol{\theta}, 1) + f_\epsilon^-(\boldsymbol{\theta}) v(\boldsymbol{\theta}, -1)) d\boldsymbol{\theta}, \tag{20}$$

where $\boldsymbol{\theta}$ represents the mid-surface coordinates for each of the three cases [10]: (x, y) for flat plates, (θ) for circular arches, and (θ, ϕ) for spherical shells; where $f_\epsilon^\pm$ represents $f^\pm$ for plates, $f^\pm R(1 \pm \epsilon)$ for arches, and $f^\pm R(1 \pm \epsilon)^2$ for shells; and where $d\boldsymbol{\theta}$ represents $d\boldsymbol{x}'$ for plates, $d\theta$ arches, and $R\sin(\theta) d\theta d\phi$ for spherical shells.

The energy norm is defined by

$$|||v|||^2 = B_\epsilon(v, v),$$

and a weighted space is defined by

$$L^2_M(\omega) = \{u : \|u\|_{L^2_M(\omega)} < \infty\}$$

and equipped with the norm

$$\|u\|^2_{L^2_M(\omega)} = \int_\omega M(\boldsymbol{\theta})|u(\boldsymbol{\theta})|^2 d\boldsymbol{\theta},$$

where, in the cases of the circular arch and flat plate, the weight $M(\boldsymbol{\theta}) \equiv 1$, while for spherical shells $M(\boldsymbol{\theta}) = R\sin\theta$.

2.2. Simple dimensionally reduced models. In [3], the accuracy of the simplest dimensionally reduced model was studied in detail. The dimensionally reduced problem for the unknown coefficient function $\alpha_0(\boldsymbol{\theta})$, obtained for spherical shells in (12), is similar in the case of circular arches and flat plates [3]:

$$\begin{aligned} -2\epsilon\bar{b}\bar{\boldsymbol{\nabla}}' \cdot \left(\mathcal{K}'\bar{\boldsymbol{\nabla}}'\alpha_0\right) &= f^+_\epsilon + f^-_\epsilon \text{ in } \omega, \\ \alpha_0 &= 0 \text{ on } \partial\omega, \end{aligned}$$

where

$$\bar{b} = \frac{1}{2}\int_{-1}^{1} k_1(z)dz.$$

The error in the simple dimensionally reduced models is bounded in each case by the following.

1. Homogeneous flat plate [2, 4]:

$$|||e|||^2 \le \frac{\epsilon}{6}\|f^+ + f^-\|^2_{L^2_M(\omega)} + \frac{\epsilon}{2}\|f^+ - f^-\|^2_{L^2_M(\omega)}.$$

2. Homogeneous circular arch (Theorem 3.2 in [3]):

$$|||e|||^2 \le \frac{1}{12}\ln\left(\frac{1+\epsilon}{1-\epsilon}\right)\|f^+_\epsilon + f^-_\epsilon\|^2_{L^2_M(\omega)} + \frac{1}{4}\ln\left(\frac{1+\epsilon}{1-\epsilon}\right)\|f^+_\epsilon - f^-_\epsilon\|^2_{L^2_M(\omega)}.$$

3. Homogeneous spherical shell (Theorem 3.4 in [3]):

$$\begin{aligned} |||e|||^2 \le \epsilon &\left(-\frac{1}{\epsilon^3}\ln\left(\frac{1+\epsilon}{1-\epsilon}\right)(D_\epsilon - \epsilon + 1) + \frac{1}{\epsilon^2}(D_\epsilon + 1)^2\right)\|f^e\|^2_{L^2_M(\omega)} \\ +\epsilon &\left(\frac{1}{\epsilon^3}\ln\left(\frac{1+\epsilon}{1-\epsilon}\right)(D_\epsilon + \epsilon - 1) + \frac{1}{\epsilon^2}(D_\epsilon - 1)^2\right)\|f^o\|^2_{L^2_M(\omega)}, \end{aligned}$$

which leads to the result already quoted in (13) since the terms in parentheses tend to 2/3 and 2 as $\epsilon \to 0$.

These bounds show that, in general, as the thickness of the domain becomes small ($\epsilon \to 0$), the accuracy of the simple models derived under the assumption $u \approx u_0$ is improved. However, it is worth bearing in mind that the bounds are valid for all ϵ, not necessarily small.

A key step [3] in developing these bounds consists of a representation for the error $B(u - u_0, v)$ in terms of Peano-type kernels: K^e_0, K^o_0,

$$B(u - u_0, v) = \int_\omega d\boldsymbol{\theta}\int_{-1}^{1} dr\,(f^e(\boldsymbol{\theta})K^e_0(r) + f^o(\boldsymbol{\theta})K^o_0(r))\,\frac{\partial v}{\partial r}, \tag{21}$$

where K_0^e, K_0^o satisfy the orthogonality condition

$$\int_{-1}^{1} \frac{1}{a} K_0^e K_0^o dr = 0.$$

Here, and in what follows, we work exclusively with the scaled bilinear form and omit the subscript ϵ on $B(u,v)$ and $F(v)$. The particular form of the kernels for homogeneous materials, $k_1 = k_2 = 1$, along with the corresponding data f^e, f^o, are given in each case by the following.

1. Flat plates:

$$K_0^e = z; K_0^o = 1,$$
$$f^e = \frac{1}{2}(f^+ + f^-); f^o = \frac{1}{2}(f^+ - f^-). \tag{22}$$

2. Circular arches:

$$K_0^e = \frac{1}{\ln\left(\frac{1+\epsilon}{1-\epsilon}\right)} \left(2\ln\left(\frac{1+\epsilon r}{1-\epsilon}\right) - \ln\left(\frac{1+\epsilon}{1-\epsilon}\right)\right); K_0^o = 1,$$
$$f^e = \frac{1}{2}(f_\epsilon^+ + f_\epsilon^-); f^o = \frac{1}{2}(f_\epsilon^+ - f_\epsilon^-). \tag{23}$$

3. Spherical shells:

$$K_0^e = \frac{1}{2}\left(1 - \sqrt{\frac{1-\epsilon}{1+\epsilon}} + r\left(1 + \sqrt{\frac{1-\epsilon}{1+\epsilon}}\right)\right),$$
$$K_0^o = \frac{1}{2}\left(1 + \sqrt{\frac{1-\epsilon}{1+\epsilon}} + r\left(1 - \sqrt{\frac{1-\epsilon}{1+\epsilon}}\right)\right); \tag{24}$$

f^e, f^0 are defined in (14).

A similar technique will be employed in the forthcoming analysis, where the error term of the higher order models, $B(u-u_N, v)$, is expressed in terms of higher order kernels K_N^e, K_N^o orthogonal with respect to the weighted inner product $\int_{-1}^{1} \frac{1}{a} K_N^e K_N^o dr$.

3. Determination of the hierarchy of dimensionally reduced models. This section is concerned with selecting the functions $\{\psi_l\}$ to be used in developing more sophisticated dimensionally reduced models of the original three-dimensional boundary value problems. First, we define the space V_N from which our dimensionally reduced approximation will be sought to be

$$V_N = \left\{ v \in V : v = \sum_{i=0}^{N} \beta_i^e(\boldsymbol{\theta})\psi_i^e(r) + \sum_{i=1}^{N} \beta_i^o(\boldsymbol{\theta})\psi_i^o(r),\ \beta_i^{e,o} \in H_0^1(\omega) \right\}. \tag{25}$$

Further, we also introduce the space $\mathcal{H}^{2N}$ with which we shall express the regularity required of the data f^e, f^o, defined by

$$\mathcal{H}^{2N} = \{\beta : A^N \beta \in L_M^2(\omega) : A^k \beta \in H_0^1(\omega), k = 0, \ldots, N-1\},$$

where $H_0^1(\omega)$ is the usual Sobolev space [1], and A is the operator

$$A[v] = -\bar{\boldsymbol{\nabla}}' \cdot \left(\mathcal{K}' \bar{\boldsymbol{\nabla}}' v\right), \tag{26}$$

with domain and range satisfying

$$L^2(\omega) \supseteq \mathcal{D}(A) \to L^2_M(\omega).$$

Remark. It can be seen that the condition $A^k\beta \in H^1_0(\omega)$ in the definition of $\mathcal{H}^{2N}$ imposes a severe restriction on the data. If the data does not satisfy these compatibility requirements, then the convergence rates given later in this section will not be attained and boundary layers [9, 29] will be formed. In practice, this means that the solution will be hard to approximate using dimensional reduction in the neighborhood of the boundary. In this eventuality one would employ a hybrid model whereby the dimensionally reduced models of the current work would be employed in the interior of the domain and coupled to a three-dimensional model near the boundary. See [19, 20, 21] for further details on boundary layers in plate models and their resolution.

We now state and prove a general result concerning how the basis functions $\{\psi_i\}_{i=0}^{\infty}$ should be selected and give the corresponding bound in the energy norm.

THEOREM 3.1. *Let V_N be the subspace*

$$V_N = \left\{ v \in V : v = \sum_{i=0}^{N} \alpha_i^e \psi_i^e + \alpha_i^o \psi_i^o,\ \alpha_i^{e,o} \in H^1_0(\omega) \right\},$$

where the functions $\{\psi_N^e, \psi_N^o\}_{N=0}^{\infty}$ are defined recursively as follows:

$$\psi_0^e(r) = \left(\int_{-1}^{1} b dr \right)^{-1}, \psi_0^o(r) = 0$$

and

$$a \frac{d\psi_N}{dr}^{\{e,o\}} = K_{N-1}^{\{e,o\}}, \int_{-1}^{1} b\psi_N^{\{e,o\}} = 0, \quad N = 1, 2, \ldots.$$

If the data satisfy $f^e, f^o \in \mathcal{H}^{2N}$, then there exists $u_N \in V_N$ such that for all $v \in V$

$$B(u - u_N, v) = \epsilon^{2N} \int_{\omega} d\boldsymbol{\theta} \int_{-1}^{1} dr \left(A^N f^e(\boldsymbol{\theta}) K_N^e(r) + A^N f^o(\boldsymbol{\theta}) K_N^o(r) \right) \frac{\partial v}{\partial r}, \tag{27}$$

where the kernels $\{K_N^e, K_N^o\}_{N=0}^{\infty}$ are defined by (22)–(24) if $N = 0$ and

$$Q(K_N^{\{e,o\}}) = K_{N-1}^{\{e,o\}}, \tag{28}$$

$$K_N^{\{e,o\}}(1) = 0 = K_N^{\{e,o\}}(-1) \tag{29}$$

for $N = 1, 2, \ldots$ with

$$Q = a \frac{d}{dr} \frac{1}{b} \frac{d}{dr}.$$

Proof. Let $u_N \in V_N$ be given by

$$u_N = \sum_{i=0}^{N} \alpha_i^e \psi_i^e + \alpha_i^o \psi_i^o,$$

where each $\alpha_i^{e,o} \in H_0^1(\omega)$ satisfies

$$\alpha_i^{\{e,o\}} = \epsilon^{2i-1} A^{i-1} f^{\{e,o\}}, \qquad i = 0, \ldots, N.$$

The condition $\alpha_i^{e,o} \in H_0^1(\omega)$ is required in order for the homogeneous boundary conditions on $\partial\omega$ to be satisfied, and it is ensured by the compatibility conditions imposed on the data to be in the space $\mathcal{H}^{2N}$.

First, we prove the result for the case when either f^e or f^o vanishes. The result in the case $N = 0$ is stated in (21) and we proceed to prove the general case by induction. Thus, assume (27) holds for $N = k$. Then, from (19),

$$\begin{aligned} B(u - u_{k+1}, v) &= B(u - u_k, v) - B(\alpha_{k+1}\psi_{k+1}, v) \\ &= \epsilon^{2k} \int_\omega d\boldsymbol{\theta} A^k f(\boldsymbol{\theta}) \int_{-1}^1 dr K_k \frac{\partial v}{\partial r} \\ &\quad - \int_\omega d\boldsymbol{\theta} \int_{-1}^1 dr \left(\alpha_{k+1} \frac{a}{\epsilon} \frac{d\psi_{k+1}}{dr} \frac{\partial v}{\partial r} + \epsilon b \psi_{k+1} (\bar{\boldsymbol{\nabla}}' \alpha_{k+1})^T \mathcal{K}' \bar{\boldsymbol{\nabla}}' v \right). \end{aligned} \tag{30}$$

Inserting $a d\psi_{k+1}/dr = K_k(r)$ and $\alpha_{k+1} = \epsilon^{2k+1} A^k f$ results in the lowest order terms in (30) cancelling, and we are left with

$$\begin{aligned} B(u - u_{k+1}, v) &= -\epsilon^{2k+2} \int_\omega d\boldsymbol{\theta} \int_{-1}^1 dr b \psi_{k+1} (\bar{\boldsymbol{\nabla}}' A^k f)^T \mathcal{K}' \bar{\boldsymbol{\nabla}}' v \\ &= \epsilon^{2k+2} \int_\omega d\boldsymbol{\theta} \bar{\boldsymbol{\nabla}}' \cdot \left(\mathcal{K}' \bar{\boldsymbol{\nabla}}' A^k f \right) \int_{-1}^1 dr b \psi_{k+1} v \\ &= -\epsilon^{2k+2} \int_\omega d\boldsymbol{\theta} A^{k+1} f \int_{-1}^1 dr b \psi_{k+1} v \\ &= -\epsilon^{2k+2} \int_\omega d\boldsymbol{\theta} A^{k+1} f \int_{-1}^1 dr \frac{dK_{k+1}}{dr} v, \end{aligned} \tag{31}$$

where the last line follows, since

$$a \frac{d\psi_i}{dr} = K_{i-1} = a \frac{d}{dr} \frac{1}{b} \frac{dK_i}{dr}, \qquad i = 1, 2, \ldots,$$

so that, due to (29), and the fact from Theorem 3.1

$$\int_{-1}^1 b \psi_N^{\{e,o\}} = 0,$$

we obtain

$$b\psi_i = \frac{dK_i}{dr}, \qquad i = 1, 2, \ldots.$$

The result then follows for $N = k+1$ at once by integrating (31) by parts with respect to r using the fact that K_{k+1} vanishes at ± 1. The general case when both f^o and f^e are present follows from linearity. □

Now as in (21), it is useful to express the error in terms of kernels K_N^o, K_N^e orthogonal in the weighted inner product $\int_{-1}^1 \frac{1}{a} K_N^e K_N^o dr$. It will now be proved that if the kernels K_N^e, K_N^o are selected as in Theorem 3.1, then they are indeed orthogonal in the weighted inner product.

LEMMA 3.1. *Let K_N^e, K_N^o be selected as in Theorem* 3.1*; then*

$$\int_{-1}^{1} \frac{K_N^e K_N^o}{a} dr = 0 \quad \forall\ N \in \mathbb{N}.$$

Proof. In [3] the kernels K_0^e, K_0^o in (22)–(24) were selected so that, if expanded in terms of the eigenfunctions of Q, that is,

$$K_0^e = \sum_{m=1}^{\infty} \alpha_m^e \phi_m, K_0^o = \sum_{m=1}^{\infty} \alpha_m^o \phi_m,$$

where ϕ_m are the eigenfunctions of Q, then

$$\alpha_m^e \alpha_m^o = 0.$$

Hence by direct calculation using (28)–(29),

$$\begin{aligned}\int_{-1}^{1} \frac{K_N^e K_N^o}{a} dr &= \int_{-1}^{1} \frac{1}{a} Q^{-N} K_0^e Q^{-N} K_0^o dr \\ &= \int_{-1}^{1} \frac{1}{a} \left(\sum_{m=1}^{\infty} \lambda_m^{-N} \alpha_m^e \phi_m \right) \left(\sum_{n=1}^{\infty} \lambda_n^{-N} \alpha_n^o \phi_n \right) dr \\ &= \sum_{m=1}^{\infty} \frac{\alpha_m^e \alpha_m^o}{\lambda_m^{2N}} \\ &= 0\end{aligned}$$

since $\alpha_m^e \alpha_m^o = 0$. ∎

The main result concerning the accuracy of the dimensionally reduced models can now be given.

THEOREM 3.2. *Suppose $f^e, f^o \in \mathcal{H}^{2N}$, and V_N be defined as in Theorem* 3.1*; let A be given as in* (26)*; and let a be given in* (19)*. Then*

$$\begin{aligned}\inf_{u_N \in V_N} |||u - u_N|||^2 \le \epsilon^{4N+1} \Bigg(&\|A^N f^e\|^2_{L^2_M(\omega)} \int_{-1}^{1} dr \frac{(K_N^e)^2}{a} \\ &+ \|A^N f^o\|^2_{L^2_M(\omega)} \int_{-1}^{1} dr \frac{(K_N^o)^2}{a} \Bigg),\end{aligned} \tag{32}$$

where K_N^e, K_N^o are given in Theorem 3.1.

Proof. Let u_N be the function constructed in Theorem 3.1. Then from (27), for $v \in V$,

$$B(u - u_N, v) = \epsilon^{2N+\frac{1}{2}} \int_{\omega} d\boldsymbol{\theta} \int_{-1}^{1} \frac{dr}{a^{\frac{1}{2}}} \left(A^N f^e(\boldsymbol{\theta}) K_N^e(r) + A^N f^o(\boldsymbol{\theta}) K_N^o(r) \right) \frac{a^{\frac{1}{2}}}{\epsilon^{\frac{1}{2}}} \frac{\partial v}{\partial r}.$$

Applying the Cauchy–Schwarz inequality and squaring, one arrives at

$$\begin{aligned}&|B(u - u_N, v)|^2 \\ &\le \epsilon^{4N+1} \int_{\omega} d\boldsymbol{\theta} \int_{-1}^{1} \frac{dr}{a} \left(A^N f^e(\boldsymbol{\theta}) K_N^e(r) + A^N f^o(\boldsymbol{\theta}) K_N^o(r) \right)^2 |||v|||^2.\end{aligned}$$

Choosing $v = u - u_N$, dividing both sides by $|||u - u_N|||^2$, and applying Lemma 3.1 gives the result claimed. ∎

In the above analysis, the model order N was the same for both odd and even components u_N^o, u_N^e of the approximation u_N. Occasionally, due to symmetries in the data, it is worthwhile using differing model orders N_o, N_e for the odd and even modes. The corresponding error bound is then as follows.

COROLLARY 3.3. *Let V_{N_e,N_o} denote the subspace*

$$V_{N_e,N_o} = \left\{ v \in V : v = \sum_{i=0}^{N_e} \alpha_i^e \psi_i^e + \sum_{i=0}^{N_o} \alpha_i^o \psi_i^o,\ \alpha_i^{e,o} \in H_0^1(\omega) \right\}$$

and suppose $f^e, f^o \in \mathcal{H}^{2N}$. Then there exists $u_{N_{e,o}} \in V_{N_e,N_o}$ such that

$$\inf_{u_{N_{e,o}} \in V_{N_e,N_o}} |||u - u_{N_{e,o}}|||^2 \le \epsilon^{4N_e+1} \left(\|A^{N_e} f^e\|^2_{L^2_M(\omega)} \int_{-1}^{1} dr \frac{(K^e_{N_e})^2}{a} \right) + \epsilon^{4N_o+1} \left(\|A^{N_o} f^o\|^2_{L^2_M(\omega)} \int_{-1}^{1} dr \frac{(K^o_{N_o})^2}{a} \right).$$

Proof. Due to linearity, if differing orders are used, then

$$B(u - u_{N_{e,o}}, v) = \int_\omega d\boldsymbol{\theta} \int_{-1}^{1} \frac{dr}{a^{\frac{1}{2}}} \left(\epsilon^{2N_e+\frac{1}{2}} A^{N_e} f^e(\boldsymbol{\theta}) K^e_{N_e}(r) + \epsilon^{2N_o+\frac{1}{2}} A^{N_o} f^o(\boldsymbol{\theta}) K^o_{N_o}(r) \right) \frac{a^{\frac{1}{2}}}{\epsilon^{\frac{1}{2}}} \frac{\partial v}{\partial r}.$$

The result then follows from the Cauchy–Schwarz inequality, and the fact that Lemma 3.1 clearly holds when differing model orders are used, so that $K^o_{N_o}, K^e_{N_e}$ are orthogonal with respect to the $1/a$ weighted inner product. ∎

4. Evaluating the constant of approximation. The error bound in Theorem 3.2 is of the form

$$\inf_{u_N \in V_N} |||u - u_N|||^2 \le \epsilon^{4N+1} C(\epsilon, N, f^e, f^o),$$

where $C(\epsilon, N, f^e, f^o)$ is a constant which depends on ϵ, the model order N, and the data f^e, f^o. It is essential to determine the dependence of the constant C on the thickness ϵ. For flat plates, as considered by Vogelius and Babuska [27], it was shown that C is independent of ϵ. However, in general C depends on ϵ in a nontrivial way. The purpose of this section is to present a technique whereby the constant can actually be evaluated completely. This strengthens the analysis of [27], but more significantly, exhibits the dependence on ϵ explicitly for the more general geometries under consideration here. Furthermore, it is not assumed that ϵ is necessarily small, and the bounds are therefore valid for thick domains.

First, an abstract result is presented, which gives an expression for the constant of approximation in terms of an infinite series involving the eigenvalues of the operator Q defined in Theorem 3.1.

THEOREM 4.1. *Let $f^e, f^o \in \mathcal{H}^{2N}$, and V_N be defined as in Theorem* 3.1. *Then*

$$\inf_{u_N \in V_N} |||u - u_N|||^2 \le \epsilon^{4N+1} \left\{ \left(\sum_{m=1}^{\infty} \frac{(\alpha_m^e)^2}{\lambda_m^{2N}} \right) \|A^N f^e\|^2_{L^2_M(\omega)} + \left(\sum_{m=1}^{\infty} \frac{(\alpha_m^o)^2}{\lambda_m^{2N}} \right) \|A^N f^o\|^2_{L^2_M(\omega)} \right\}, \tag{33}$$

where α_m^o, α_m^o *are given by*

$$\alpha_m^{e,o} = \int_{-1}^{1} \frac{K_0^{e,o}\phi_m}{a} ds$$

and where $\{\lambda_m\phi_m\}_{m=0}^{\infty}$ *are the eigensolutions of* Q.

Proof. Let u_N be constructed as in Theorem 3.1 and define

$$C(\epsilon, N) = \int_{-1}^{1} \frac{(K_N)^2}{a} dr$$

where we omit the subscripts e, o, since the analysis is identical for both the odd and even cases. In order to evaluate C explicitly, we recall the recurrence relation from Theorem 3.1:

$$QK_N = K_{N-1}; K_N(\pm 1) = 0 \ , \ N = 1, \ldots, \tag{34}$$

where

$$Q = a\frac{d}{dr}\frac{1}{b}\frac{d}{dr},$$

and K_0 is defined in (22)–(24). The solution of this differential equation is given by

$$K_N(r) = \int_{-1}^{1} \frac{G(r,s)K_{N-1}(s)}{a(s)} ds,$$

where G is the Green function written in the form [16]

$$G(r,s) = \sum_{m=1}^{\infty} \frac{\phi_m(r)\phi_m(s)}{\lambda_m}$$

and ϕ_m, λ_m are the eigensolutions of the Sturm–Liouville problem

$$\frac{d}{dr}\frac{1}{b}\frac{d\phi_m}{dr} + \frac{\lambda_m}{a}\phi_m = 0, \qquad \phi_m(\pm 1) = 0,$$

normalized so that

$$\int_{-1}^{1} \frac{\phi_m\phi_n}{a} dr = \delta_{mn}. \tag{35}$$

Consequently,

$$K_N(r) = \int_{-1}^{1} ds \frac{K_{N-1}(s)}{a(s)} \left(\sum_{m=1}^{\infty} \frac{\phi_m(r)\phi_m(s)}{\lambda_m} \right). \tag{36}$$

Expanding $K_0 = \sum_{n=1}^{\infty} \alpha_n\phi_n$ and using the recurrence relation (34), we obtain

$$\begin{aligned} K_1 &= \int_{-1}^{1} ds \frac{K_0(s)}{a(s)} \left(\sum_{m=1}^{\infty} \frac{\phi_m(r)\phi_m(s)}{\lambda_m} \right) \\ &= \int_{-1}^{1} ds \sum_{n=1}^{\infty} \frac{\alpha_n\phi_n(s)}{a(s)} \left(\sum_{m=1}^{\infty} \frac{\phi_m(r)\phi_m(s)}{\lambda_m} \right) \\ &= \sum_{n=1}^{\infty} \frac{\alpha_n\phi_n(r)}{\lambda_n} \end{aligned}$$

on recalling (35). Similarly, by repeated application of (36), we conclude

$$K_N = \sum_{m=1}^{\infty} \frac{\phi_m}{\lambda_m} \int_{-1}^{1} ds \frac{\phi_m(s)}{a(s)} \left(\sum_{n=1}^{\infty} \frac{\alpha_n \phi_n(s)}{\lambda_n^{N-1}} \right) = \sum_{m=1}^{\infty} \frac{\alpha_m \phi_m(r)}{\lambda_m^N}. \tag{37}$$

It follows from (35) that the constant C in the error bound is given by

$$\int_{-1}^{1} ds \frac{K_N^2}{a} = \int_{-1}^{1} \frac{1}{a} \left(\sum_{m=1}^{\infty} \frac{\alpha_m \phi_m(r)}{\lambda_m^N} \right)^2 = \sum_{m=1}^{\infty} \frac{\alpha_m^2}{\lambda_m^{2N}}. \tag{38}$$

The result now follows directly from Theorem 3.2. □

The general error bound obtained in Theorem 4.1 will now be evaluated for the three cases of a homogeneous flat plate, a homogeneous circular arch, and a homogeneous spherical shell.

4.1. Homogeneous flat plate.

THEOREM 4.2. *Let $f^e, f^o \in \mathcal{H}^{2N}$, and V_N be defined as in Theorem* 3.1. *Then*

$$\inf_{u_N \in V_N} |||u - u_N|||^2 \leq \left(\frac{\epsilon}{\pi}\right)^{4N+1} \frac{4}{\pi} \left\{ \zeta(4N+2) \|A^N f^e\|^2_{L^2(\omega)} + \zeta\left(4N+2, -\tfrac{1}{2}\right) \|A^N f^o\|^2_{L^2(\omega)} \right\},$$

where $\zeta(n)$ represents the Riemann–Zeta function [11]

$$\zeta(n) = \sum_{i=1}^{\infty} \frac{1}{i^n}$$

and where $\zeta(n, w)$ represents the generalized Riemann–Zeta function [11]

$$\zeta(n, w) = \sum_{i=1}^{\infty} \frac{1}{(i+w)^n}.$$

Remark. Note that the above result and the ones that follow in this section show that if ϵ is sufficiently small, then exponential rates of convergence are obtained for compatible data. In fact, the method converges, as $N \to \infty$, for all values of ϵ, and for a proof of this for flat plates, see Vogelius and Babuska [28].

The proof of Theorem 4.2 follows at once from the two special cases presented in the following lemmas.

LEMMA 4.1. *Let $f^e, f^o \in \mathcal{H}^{2N}$, and V_N be defined as in Theorem* 3.1. *If $f^+ = f^-$ (so that $f^o = 0$), then*

$$\inf_{u_N \in V_N} |||u - u_N|||^2 \leq \left(\frac{\epsilon}{\pi}\right)^{4N+1} \frac{4}{\pi} \zeta(4N+2) \|A^N f^e\|^2_{L^2(\omega)}. \tag{39}$$

Proof. It is merely necessary to calculate the quantities appearing in the bounds of Theorem 4.1. Expanding $K_0^e = s$ in terms of the eigenfunctions ϕ_m, where $\phi_m = \sin m\pi$, and $\lambda_m = (m\pi)^2$, we obtain

$$\alpha_m^e = \frac{2}{\pi m} (-1)^{m+1}.$$

Hence (33) becomes

$$\begin{aligned}\sum_{m-1}^{\infty}\frac{(\alpha_m^e)^2}{\lambda_m^{2N}} &= \frac{4}{\pi^2}\sum_{m=1}^{\infty}\frac{1}{m^2\lambda_m^{2N}}\\ &= \frac{4}{\pi^2}\sum_{m=1}^{\infty}\frac{1}{m^2}\frac{1}{(m\pi)^{4N}}\\ &= \frac{4}{\pi^{4N+2}}\sum_{m=1}^{\infty}\frac{1}{m^{4N+2}}\\ &= \frac{4}{\pi^{4N+2}}\zeta(4N+2).\end{aligned}$$

Hence, after substituting this result into Theorem 3.1, the lemma follows. ∎

The procedure which has just been used to obtain a bound in the case of symmetric data may be carried out similarly for the case of antisymmetric data to obtain the following lemma.

LEMMA 4.2. *Let $f^e, f^o \in \mathcal{H}^{2N}$, and V_N be defined as in Theorem* 3.1. *Then if $f^+ = -f^-$ (so that $f^e = 0$),*

$$\inf_{u_N\in V_N}|||u-u_N|||^2 \le \frac{4}{\pi}\left(\frac{\epsilon}{\pi}\right)^{4N+1}\zeta\left(4N+2,-\frac{1}{2}\right)\|A^N f^o\|^2_{L^2(\omega)}. \tag{40}$$

Proof. As before we need to expand K_0^o in terms of $\{\phi_m\}$, and from our earlier analysis we know that if we transform K_0^o onto $(0,1)$ and expand in a half-range Fourier sine series, then

$$\alpha_m^o = \frac{2}{\pi m}(1+(-1)^{m+1})$$

and therefore

$$\begin{aligned}K_0^o(\hat{r}) &= \frac{2}{\pi}\sum_{m=1}^{\infty}\frac{(1+(-1)^{n+1})\phi_m(\hat{r})}{m}\\ &= \frac{4}{\pi}\sum_{m=1}^{\infty}\frac{\phi_{2m-1}(\hat{r})}{2m-1}.\end{aligned}$$

Hence, from (37),

$$K_N^o(\hat{r}) = \frac{4}{\pi}\sum_{m=1}^{\infty}\frac{\phi_{2m-1}(\hat{r})}{\lambda_{2m-1}^N(2m-1)}$$

so that

$$\begin{aligned}\|K_N^o(r)\|^2_{L^2(-1,1)} &= 2\|K_N^o(\hat{r})\|^2_{L^2(0,1)}\\ &= 2\frac{16}{\pi^2}\int_0^1\left(\phi^2_{2m-1}ds\right)\sum_{m=1}^{\infty}\frac{1}{\lambda^{2N}_{2m-1}(2m-1)^2}\\ &= \frac{16}{\pi^2}\sum_{m=1}^{\infty}\frac{1}{\lambda^{2N}_{2m-1}(2m-1)^2}\end{aligned}$$

since $\int_0^1 \phi_m \phi_n ds = \frac{1}{2}\delta_{mn}$. But $\lambda_m = (\frac{m\pi}{2})^2$ for the rescaled problem and hence we arrive at

$$\begin{aligned}\|K_N^o\|_{L^2}^2 &= \frac{16}{\pi^2}\sum_{m=1}^{\infty}\frac{1}{(\frac{2m-1}{2})^{4N}\pi^{4N}(2m-1)^2}\\ &= \frac{4}{\pi^{4N+2}}\sum_{m=1}^{\infty}\frac{1}{(m-\frac{1}{2})^{4N+2}}\\ &= \frac{4}{\pi^{4N+2}}\zeta\left(4N+2,-\frac{1}{2}\right). \qquad \square\end{aligned}$$

The proof of Theorem 4.2 follows directly from Lemmas 4.1 and 4.2.

4.2. Homogeneous circular arch.

THEOREM 4.3. *Let $f^e, f^o \in \mathcal{H}^{2N}$, and V_N be defined as in Theorem* 3.1. *Then for the arch problem we have*

$$\inf_{u_N\in V_N}|||u-u_N|||^2 \le \left(\frac{1}{2\pi}\ln\left(\frac{1+\epsilon}{1-\epsilon}\right)\right)^{4N+1}\frac{4}{\pi}\left\{\zeta(4N+2)\|A^N f^e\|_{L^2(\omega)}^2 + \zeta\left(4N+2,-\frac{1}{2}\right)\|A^N f^o\|_{L^2(\omega)}^2\right\}.$$

Proof. The result can be obtained by applying Theorem 4.1. However, by a change of variable, the result for circular arches can be derived from that for flat plates. To show this, recall that the bilinear form for arches is given by

$$B(u,v) = \int_\omega\int_{-1}^1\left\{\frac{(1+\epsilon r)}{\epsilon}\frac{\partial u}{\partial r}\frac{\partial v}{\partial r} + \frac{\epsilon}{1+\epsilon r}(\bar{\boldsymbol{\nabla}}' u)^T\mathcal{K}'\bar{\boldsymbol{\nabla}}' v\right\}dr\,d\theta.$$

Then, making the substitution

$$t = \frac{2}{\ln\left(\frac{1+\epsilon}{1-\epsilon}\right)}\left(\ln(1+\epsilon r) - \ln(1-\epsilon) - \frac{1}{2}(\ln(1+\epsilon) - \ln(1-\epsilon))\right)$$

we arrive at

$$B(u,v) = \int_\omega\int_{-1}^1\left\{\frac{2}{\ln\left(\frac{1+\epsilon}{1-\epsilon}\right)}\frac{\partial u}{\partial t}\frac{\partial v}{\partial t} + \frac{\ln\left(\frac{1+\epsilon}{1-\epsilon}\right)}{2}(\bar{\boldsymbol{\nabla}}' u)^T\mathcal{K}'\bar{\boldsymbol{\nabla}}' v\right\}dt\,d\theta.$$

It can now be seen that the $B(u,v)$ in this case is equivalent to $B(u,v)$ in the flat plate case, except that now we have $\frac{1}{2}\ln(\frac{1+\epsilon}{1-\epsilon})$ instead of ϵ. Therefore the functions which optimally capture the variation in the thickness of the true solution in this case will be polynomials in t, that is, in $\ln(1+\epsilon r)$. As a consequence of this equivalence to the flat plate case, one can easily obtain formulae for arches by referring to the formulae for plates and replacing ϵ by $\frac{1}{2}\ln\left(\frac{1+\epsilon}{1-\epsilon}\right)$. $\square$

Remark. It can be seen from the above formula that exponential convergence is obtained for $\frac{1}{2\pi}\ln\left(\frac{1+\epsilon}{1-\epsilon}\right) < 1$, that is, $\epsilon < \tanh(2\pi) \approx 0.996$. So for arches, the bound promises exponential convergence for practically the whole range of values of the thickness (if the data is sufficiently compatible).

4.3. Homogeneous spherical shells. In order to derive the bound in the energy norm we estimate the weighted L_2 norms of K_N^o,K_N^e separately and then combine them as in Theorem 4.1 to obtain an overall bound. From earlier, we know that the eigensolutions $\lambda_m\phi_m$ of the equation

$$-(\epsilon r+1)^2\frac{d^2\phi_m}{dr^2}=\lambda\phi_m,\qquad \phi_m(\pm1)=0,\qquad m=1,2,\ldots,$$

are given by

$$\phi_m(r)=\left(\frac{2\epsilon(1+\epsilon r)}{\ln\left(\frac{1+\epsilon}{1-\epsilon}\right)}\right)^{\frac12}\sin\left(\frac{m\pi\ln\left(\frac{1+\epsilon r}{1-\epsilon}\right)}{\ln\left(\frac{1+\epsilon}{1-\epsilon}\right)}\right),$$

$$\lambda_m=\epsilon^2\left(\left(\frac{m\pi}{\ln\left(\frac{1+\epsilon}{1-\epsilon}\right)}\right)^2+\frac14\right).$$

One may proceed as in the case of flat plates and circular arches by using the eigenfunctions and eigenvalues to evaluate the weighted L_2 norm of K_N^e, K_N^o. Using the expressions for the weighted L_2 norm of the kernels K_N^o, K_N^e we can now state and prove a result concerning the convergence rate for dimensional reduction on a spherical shell.

THEOREM 4.4. *Let $f^e, f^o \in \mathcal{H}^{2N}$, and V_N be defined as in Theorem* 3.1. *Then*

$$\inf_{u_N\in V_N}|||u-u_N|||^2\le\frac{4}{\pi(1+\epsilon)}\left(\frac{1}{2\pi}\ln\left(\frac{1+\epsilon}{1-\epsilon}\right)\right)^{4N+1}\times\left\{\zeta\left(4N+2,-\frac12\right)\|A^Nf^o\|^2_{L^2_M(\omega)}+\zeta(4N+2)\|A^Nf^e\|^2_{L^2_M(\omega)}\right\}.$$

Before commencing with the proof, two lemmas are stated and proved, from which the proof of the theorem will immediately follow.

LEMMA 4.3. *Let K_N^o be as in Theorem* 3.1 *and a be as in* (19). *Then*

$$\left\|\frac{K_N^o}{a^{\frac12}}\right\|^2_{L^2(-1,1)}\le\frac{4}{\pi(1+\epsilon)}\left(\frac{1}{2\epsilon\pi}\ln\left(\frac{1+\epsilon}{1-\epsilon}\right)\right)^{4N+1}\zeta\left(4N+2,-\frac12\right).$$

Proof. Now it can be shown that (see [3]) if K_0^o is expanded in terms of the eigenfunctions of Q in the form

$$K_0^o=\sum_{m=1}^{\infty}\alpha_m^o\phi_m,$$

then

$$\alpha_m^o=4\pi m\left(\frac{2\ln\left(\frac{1+\epsilon}{1-\epsilon}\right)}{\epsilon(1-\epsilon)}\right)^{\frac12}D_\epsilon\left(\frac{1+(-1)^{m+1}}{\ln\left(\frac{1+\epsilon}{1-\epsilon}\right)^2+4\pi^2m^2}\right)$$

$$=\frac{m}{\pi}\left(\frac{2\ln\left(\frac{1+\epsilon}{1-\epsilon}\right)}{\epsilon(1+\epsilon)}\right)^{\frac12}\left(\left\{\frac{1+(-1)^{m+1}}{(m^2+\sigma^2)}\right\}\right),$$

where $\sigma = \frac{1}{2\pi}\ln\left(\frac{1+\epsilon}{1-\epsilon}\right)$ and $D_\epsilon = \sqrt{\frac{1-\epsilon}{1+\epsilon}}$.

Hence, substituting in the value of α_m^o into (38) one obtains

$$\begin{aligned}\left\|\frac{K_N^o}{a^{\frac{1}{2}}}\right\|^2_{L^2(-1,1)} &= \sum_{m=1}^{\infty}\frac{(\alpha_m^o)^2}{\lambda_m^{2N}}\\ &= \frac{2}{\pi^2\epsilon(1+\epsilon)}\ln\left(\frac{1+\epsilon}{1-\epsilon}\right)\sum_{m=1}^{\infty}\frac{((-1)^{m+1}+1)^2m^2}{(m^2+\sigma^2)^2\lambda_m^{2N}}. \qquad (41)\end{aligned}$$

Recalling that

$$\lambda_m = \frac{\epsilon^2\pi^2}{\ln\left(\frac{1+\epsilon}{1-\epsilon}\right)^2}(m^2+\sigma^2), \qquad (42)$$

and substituting into (41), one obtains

$$\begin{aligned}&\left\|\frac{K_N^o}{a^{\frac{1}{2}}}\right\|^2_{L^2(-1,1)} \qquad (43)\\ &= \frac{2}{\pi^2\epsilon(1+\epsilon)}\ln\left(\frac{1+\epsilon}{1-\epsilon}\right)\left(\frac{1}{\epsilon\pi}\ln\left(\frac{1+\epsilon}{1-\epsilon}\right)\right)^{4N}\sum_{m=1}^{\infty}\frac{((-1)^{m+1}+1)^2m^2}{(m^2+\sigma^2)^{2N+2}}\\ &= \frac{2}{\pi(1+\epsilon)}\left(\frac{1}{\epsilon\pi}\ln\left(\frac{1+\epsilon}{1-\epsilon}\right)\right)^{4N+1}\sum_{m=1}^{\infty}\frac{4(2m-1)^2}{((2m-1)^2+\sigma^2)^{2N+2}}\\ &= \frac{4}{\pi(1+\epsilon)}\left(\frac{1}{2\epsilon\pi}\ln\left(\frac{1+\epsilon}{1-\epsilon}\right)\right)^{4N+1}\sum_{m=1}^{\infty}\frac{(m-\frac{1}{2})^2}{\left((m-\frac{1}{2})^2+\left(\frac{\sigma}{2}\right)^2\right)^{2N+2}}\\ &\le \frac{4}{\pi(1+\epsilon)}\left(\frac{1}{2\epsilon\pi}\ln\left(\frac{1+\epsilon}{1-\epsilon}\right)\right)^{4N+1}\sum_{m=1}^{\infty}\frac{1}{(m-\frac{1}{2})^{4N+2}}. \qquad (44)\end{aligned}$$

Hence, one arrives at the estimate

$$\left\|\frac{K_N^o}{a^{\frac{1}{2}}}\right\|^2_{L^2(-1,1)} \le \frac{4}{\pi(1+\epsilon)}\left(\frac{1}{2\epsilon\pi}\ln\left(\frac{1+\epsilon}{1-\epsilon}\right)\right)^{4N+1}\zeta\left(4N+2,-\frac{1}{2}\right),$$

which is the result stated in the lemma. □

LEMMA 4.4. *Let K_N^e be as in Theorem* 3.1 *and a be as in* (19). *Then*

$$\left\|\frac{K_N^e}{a^{\frac{1}{2}}}\right\|^2_{L^2(-1,1)} \le \frac{4}{\pi(1+\epsilon)}\left(\frac{1}{2\epsilon\pi}\ln\left(\frac{1+\epsilon}{1-\epsilon}\right)\right)^{4N+1}\zeta(4N+2).$$

Proof. It can be shown that if $K_0^e = \sum_{m=1}^{\infty}\alpha_m^e\phi_m$, then

$$\alpha_m^e = \frac{m}{\pi}\left(\frac{2\ln\left(\frac{1+\epsilon}{1-\epsilon}\right)}{\epsilon(1+\epsilon)}\right)^{\frac{1}{2}}\left\{\frac{(-1)^m+1}{(m^2+\sigma^2)}\right\}. \qquad (45)$$

Using this expression for α_m^e and reasoning as before, one obtains

$$
\begin{aligned}
(46)\qquad & \left\| \frac{K_N^e}{a^{\frac{1}{2}}} \right\|^2_{L^2(-1,1)} \\
&= \sum_{m=1}^{\infty} \frac{(\alpha_m^e)^2}{\lambda_m^{2N}} \\
&= \frac{2}{\pi^2 \epsilon(1+\epsilon)} \ln\left(\frac{1+\epsilon}{1-\epsilon}\right) \sum_{m=1}^{\infty} \frac{((-1)^m+1)^2 m^2}{\left(m^2+\sigma^2\right)^2 \lambda_m^{2N}} \\
&= \frac{2}{\pi^2 \epsilon(1+\epsilon)} \ln\left(\frac{1+\epsilon}{1-\epsilon}\right) \left(\frac{1}{\epsilon\pi} \ln\left(\frac{1+\epsilon}{1-\epsilon}\right)\right)^{4N} \sum_{m=1}^{\infty} \frac{4(2m)^2}{\left((2m)^2+\sigma^2\right)^{2N+2}} \\
&= \frac{2}{\pi(1+\epsilon)} \left(\frac{1}{\epsilon\pi} \ln\left(\frac{1+\epsilon}{1-\epsilon}\right)\right)^{4N+1} \frac{1}{2^{4N}} \sum_{m=1}^{\infty} \frac{m^2}{\left(m^2+\left(\frac{\sigma}{2}\right)^2\right)^{2N+2}} \\
&= \frac{4}{\pi(1+\epsilon)} \left(\frac{1}{2\epsilon\pi} \ln\left(\frac{1+\epsilon}{1-\epsilon}\right)\right)^{4N+1} \sum_{m=1}^{\infty} \frac{m^2}{\left(m^2+\left(\frac{\sigma}{2}\right)^2\right)^{2N+2}} \\
&\le \frac{4}{\pi(1+\epsilon)} \left(\frac{1}{2\epsilon\pi} \ln\left(\frac{1+\epsilon}{1-\epsilon}\right)\right)^{4N+1} \sum_{m=1}^{\infty} \frac{1}{m^{4N+2}} \\
(47)\qquad &= \frac{4}{\pi(1+\epsilon)} \left(\frac{1}{2\epsilon\pi} \ln\left(\frac{1+\epsilon}{1-\epsilon}\right)\right)^{4N+1} \zeta(4N+2),
\end{aligned}
$$

which is the result stated in the lemma. ∎

The proof of Theorem 4.4 follows directly from Lemma 4.3, Lemma 4.4, and Theorem 3.2.

Remark. Note that the error bound in Theorem 4.4 is identical to that of arches in Theorem 4.3 except for a factor of $(1+\epsilon)^{-1}$.

5. Numerical examples. In this section, the optimal functions to be used to construct the dimensionally reduced models in the case of circular arches and spherical shells will be given explicitly. The functions are then used to construct the subspace $V_N \subset V$ defined in (25). The dimensionally reduced approximation of the problem

$$u \in V : B(u,v) = F(v) \quad \forall v \in V$$

consists of seeking the solution

$$u_N^* \in V_N : B(u_N^*, v_N) = F(v_N) \quad \forall v_N \in V_N.$$

This results in the following elliptic system, in one dimension less than the original problem

$$(48)\qquad -\epsilon \bar{\boldsymbol{B}} \nabla' \cdot \mathcal{K} \nabla' \boldsymbol{\alpha} + \frac{1}{\epsilon} \bar{\boldsymbol{A}} \boldsymbol{\alpha} = \boldsymbol{c}^+ f_\epsilon^+ + \boldsymbol{c}^- f_\epsilon^- \quad \text{in } \omega$$

subject to $\boldsymbol{\alpha} = \boldsymbol{0}$ on $\partial\omega$, where

$$(49)\qquad \bar{B}_{jk} = \int_{-1}^{1} b \psi_j \psi_k dr, \ \bar{A}_{jk} = \int_{-1}^{1} a \frac{d\psi_j}{dr} \frac{d\psi_k}{dr} dr,$$

$$(50)\qquad \boldsymbol{\alpha} = (\alpha_0, \ldots, \alpha_N)^T; \boldsymbol{c}_j^{\pm} = \psi_j(\pm 1),$$

with the approximation given by

$$u_N = \boldsymbol{\alpha}^T \Psi,$$

where $\Psi = (\psi_0, \ldots, \psi_N)^T$.

In general, the solution u_N^* will differ from the function u_N constructed in Theorem 3.1. However, the standard property of the Galerkin method means that

$$|||u - u_N^*||| \leq |||u - v_N||| \quad \forall v_N \in V_N$$

and as a consequence

$$|||u - u_N^*||| \leq \inf_{u_N \in V_N} |||u - u_N|||.$$

The approximation theoretic results of the foregoing sections find immediate application in obtaining bounds for the accuracy of the dimensionally reduced approximation u_N^*.

The optimal functions will now be used in the resolution of some simple illustrative problems in order to verify the theoretical results concerning the rates of convergence. Occasionally, the odd and even model orders N_o and N_e will differ, as in Corollary 3.3, and the model order is then written as a pair (N_e, N_o).

5.1. Flat plates.

5.1.1. Selection of optimal functions. It follows from Theorem 3.1 that $\psi_0^e = 1/2$, and that the functions $\psi_N^{e,o}$ for $N = 1, 2, \ldots$, for a flat plate are given by

$$\frac{d\psi_N^{e,o}}{dr} = K_{N-1}^{e,o}, \int_{-1}^{1} \psi_N^{e,o} dr = 0,$$

where $K_0^e = r$, $K_0^o = 1$, and $K_N^{e,o}$ for $N = 1, 2, \ldots$ are given by

$$\frac{d^2 K_N^{e,o}}{dr^2} = K_{N-1}^{e,o}, K_N^{e,o}(\pm 1) = 0, \quad N = 1, 2, \ldots.$$

From these relations, it is easy to show that K_N^e and K_N^o are odd and even polynomials, respectively, and consequently ψ_N^e and ψ_N^o are even and odd polynomials, respectively. Since it is not the exact basis functions themselves but the space spanned by them that is crucial in order to obtain the approximation properties given earlier in the paper, we have freedom to choose the polynomial basis used. In the numerical results that follow, the basis used was $\psi_0^e = 1, \psi_1^o = r$, and

$$\psi_i^e = \int_{-1}^{r} P_{2i-1}(s) ds, \quad i = 1, 2, \ldots, \; \psi_i^o = \int_{-1}^{r} P_{2i-2}(s) ds, \quad i = 2, 3, \ldots,$$

where P_i is the Legendre polynomial of degree i. The matrices $\bar{\boldsymbol{A}}, \bar{\boldsymbol{B}}$ can be calculated explicitly with this choice of basis, with entries given by

$$\bar{A}_{ij}^{plate} = \frac{2}{2i-1} \delta_{ij},$$

and

$$\bar{B}_{ii}^{plate} = \frac{1}{(2i-1)^2} \left(\frac{2}{2i+1} + \frac{2}{2i-3} \right),$$

$$\bar{B}_{i,i+2}^{plate} = \frac{-2}{(2i-1)(2i+1)(2i+3)},$$

$$\bar{B}_{i,i-2}^{plate} = \frac{-2}{(2i-5)(2i-3)(2i-1)},$$

with the remaining entries vanishing.

5.1.2. Numerical example for a flat plate. We approximate the flat plate problem with domain and flux given by

$$\begin{aligned}\Omega &= (-1,1)\times(-\epsilon,\epsilon),\\ f^+ &= -\pi e^{-\pi\epsilon}\sin(\pi x),\\ f^- &= \pi e^{\pi\epsilon}\sin(\pi x).\end{aligned}$$

For this problem the true solution is given by $e^{-\pi y}\sin(\pi x)$. Note that

$$\begin{aligned}f^e &= \pi(e^{\pi\epsilon}-e^{-\pi\epsilon})\sin(\pi x),\\ f^o &= -\pi(e^{\pi\epsilon}+e^{-\pi\epsilon})\sin(\pi x).\end{aligned}$$

With the aid of Corollary 3.3 the error is bounded by

$$\begin{aligned}|||u-u_N|||^2 &\le \epsilon^{4N_e+1}C(N_e)\|A^{N_e}f^e\|^2_{L^2(\omega)}\\ &\quad +\epsilon^{4N_o+1}C(N_o)\|A^{N_o}f^o\|^2_{L^2(\omega)}.\end{aligned}\tag{51}$$

Now

$$\begin{aligned}\|A^{N_e}f^e\|^2_{L^2(\omega)} &= \|\pi^{2N_e+1}2\sinh(\pi\epsilon)\sin(\pi x)\|^2_{L^2(\omega)} = 4\pi^{4N_e+2}\sinh^2(\pi\epsilon),\\ \|A^{N_o}f^o\|^2_{L^2(\omega)} &= \|\pi^{2N_o+1}2\cosh(\pi\epsilon)\sin(\pi x)\|^2_{L^2(\omega)} = 4\pi^{4N_o+2}\cosh^2(\pi\epsilon).\end{aligned}$$

Substituting these into (51), this bound can then be rewritten as

$$\begin{aligned}|||u-u_N|||^2 &\le 4\pi^{4N_e+2}\epsilon^{4N_e+1}C(N_e)\sinh^2(\pi\epsilon)\\ &\quad +4\pi^{4N_o+2}\epsilon^{4N_o+1}C(N_o)\cosh^2(\pi\epsilon).\end{aligned}\tag{52}$$

The overall convergence rate will be governed by the lowest order term in (52). Note that $\sinh^2(\pi\epsilon)=O(\epsilon^2)$, and this will affect the order of convergence when the lowest order term is the even one (that is, where $N_o > N_e$). So when $N_o \le N_e$, (52) predicts ϵ^{4N_o+1} in the bound, but when $N_o > N_e$, (52) predicts ϵ^{4N_e+3}. As can be seen from Figure 1, these predicted results are attained. Vogelius and Babuska [27] found bounds in the flat plate case of the form

$$|||u-u_N|||^2 \le C(u,N)\epsilon^{4N+1}.$$

This bound takes no account of symmetries in the data that lead to higher rates of convergence than $O(\epsilon^{4n+1})$ suggested here. Schwab (e.g., [19, 20, 22]) has split the problem into odd and even components and shows that in the case of even data, a rate of ϵ^{4N_e+1} is obtained, which agrees with the bound stated in Theorem 4.2.

5.2. Circular arches.

5.2.1. Selection of optimal functions. It follows from Theorem 3.1 that

$$\psi_0^e = \frac{1}{2\epsilon}\ln\left(\frac{1+\epsilon}{1-\epsilon}\right)$$

and that the functions $\psi_N^{e,o}$ for $N=1,2,\ldots$, for a circular arch are given by

$$(1+\epsilon r)\frac{d\psi_N^{e,o}}{dr} = K_{N-1}^{e,o}, \int_{-1}^{1}\frac{1}{1+\epsilon r}\psi_N^{e,o}\,dr = 0,$$

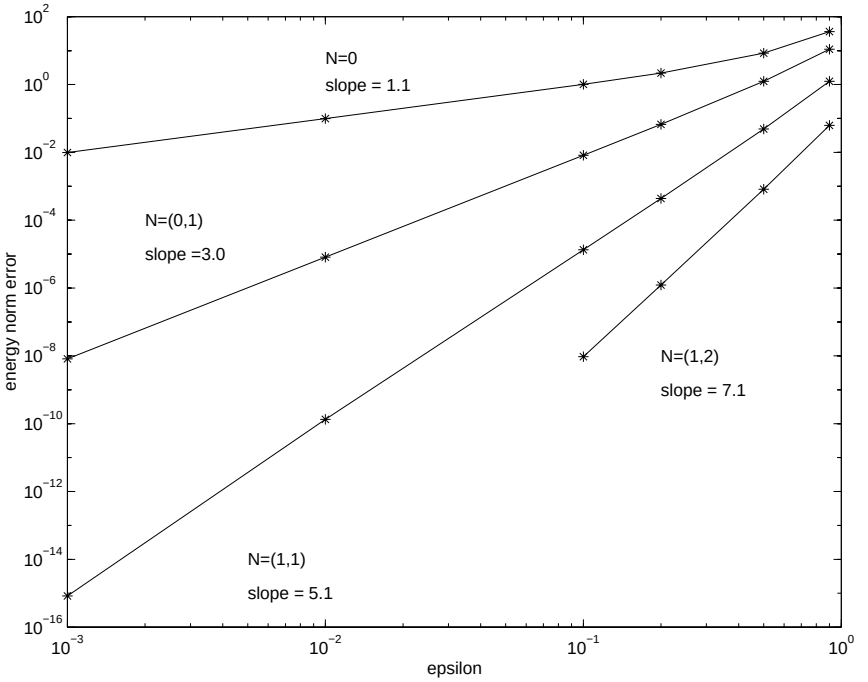


FIG. 1. *Convergence rates for the flat plate problem.*

where K_0^e, K_0^o are as in (23), and $K_N^{e,o}$ for $N = 1, 2, \ldots$ are given by

$$(1+\epsilon r)\frac{d}{dr}(1+\epsilon r)\frac{dK_N^{e,o}}{dr} = K_{N-1}^{e,o}, K_N^{e,o}(\pm 1) = 0, \quad N = 1, 2, \ldots.$$

From these relations, it is easy to show that a basis spanning the same space as the solutions to the above relations is given by the following logarithmic polynomial basis:

$$\psi_0^e = 1; \psi_1^o = \frac{2}{\ln\left(\frac{1+\epsilon}{1-\epsilon}\right)}\left(\ln(1+\epsilon r) - \frac{1}{2}\left(\ln(1+\epsilon) + \ln(1-\epsilon)\right)\right),$$

and

$$\psi_i^e = \int_{-1}^{t} P_{2i-1}(s)ds, \ i = 1, 2, \ldots,$$
$$\psi_i^o = \int_{-1}^{t} P_{2i-2}(s)ds, \ i = 2, 3, \ldots,$$

where

$$t = \frac{2}{\ln\left(\frac{1+\epsilon}{1-\epsilon}\right)}\left(\ln(1+\epsilon r) - \frac{1}{2}\left(\ln(1+\epsilon) + \ln(1-\epsilon)\right)\right) \tag{53}$$

and P_i is the Legendre polynomial of degree i. Now

$$\bar{A}_{ij} = \int_{-1}^{1}(1+\epsilon r)\frac{d\psi_i}{dr}\frac{d\psi_j}{dr}dr,$$

and so with t as in (53), one obtains

$$\bar{A}_{ij} = \frac{2\epsilon}{\ln\left(\frac{1+\epsilon}{1-\epsilon}\right)} \int_{-1}^{1} \frac{d\psi_i}{dt} \frac{d\psi_j}{dt} dt.$$

Hence, with the above choice of basis

$$\begin{aligned} \bar{A}^{arch}_{ij} &= \frac{2\epsilon}{\ln\left(\frac{1+\epsilon}{1-\epsilon}\right)} \frac{2}{2i-1} \delta_{ij} \\ &= \frac{2\epsilon}{\ln\left(\frac{1+\epsilon}{1-\epsilon}\right)} \bar{A}^{plate}_{ij}, \end{aligned}$$

which is the same as for flat plates but with an ϵ dependent scaling. Similarly,

$$\bar{B}^{arch} = \frac{1}{2\epsilon} \ln\left(\frac{1+\epsilon}{1-\epsilon}\right) \bar{B}^{plate}.$$

5.2.2. Numerical examples for a circular arch. We illustrate in this section that the convergence rates predicted in Theorem 4.3 are attained for arches using the optimal basis functions. We use the following annular domain with which to verify our theory:

$$\Omega = (0, \pi) \times (1 - \epsilon, 1 + \epsilon).$$

The results in Theorem 4.3 are verified using data with various symmetries to illustrate each of the possibilities.

1. Symmetric data: $f^+_\epsilon = f^-_\epsilon = x(\pi - x)$. Theorem 4.3 predicts that

$$|||u - u_N|||^2 \leq \left(\frac{1}{2\pi} \ln\left(\frac{1+\epsilon}{1-\epsilon}\right)\right)^{4N+1} \zeta(4N+2) \|\triangle^N f^e\|^2_{L^2(\omega)},$$

   where N is the even model order. In particular, the odd basis functions have no effect on the accuracy of the approximation since $f^o = 0$ and there is no odd component of the error term. Since $f_e \in \mathcal{H}^2$ we would not expect to obtain the rate $\ln\left(\frac{1+\epsilon}{1-\epsilon}\right)^{4N+1}$ for values of N larger than 1. The results are given in Figure 2. As can be seen the convergence rate for even model orders $N = 0, 1$ are $\ln\left(\frac{1+\epsilon}{1-\epsilon}\right), \ln\left(\frac{1+\epsilon}{1-\epsilon}\right)^5$, respectively, as predicted, while the full rate is not attained for $N = 2$.
2. Antisymmetric data: $f^+_\epsilon = -f^-_\epsilon = x\pi(x^2 - \pi^2) + x^3(\pi - x)$. Theorem 4.3 predicts that

$$|||u - u_N\|^2 \leq \left(\frac{1}{2\pi} \ln\left(\frac{1+\epsilon}{1-\epsilon}\right)\right)^{4N+1} \frac{4}{\pi} \zeta\left(4N+2, -\frac{1}{2}\right) \|\triangle^N f^o\|^2_{L^2(\omega)},$$

   where N is the odd model order. In particular, the even basis functions now have no effect on the accuracy of the approximation since $f^e = 0$ and there is no even component of the error term. Since $f_o \in \mathcal{H}^4$ we would not expect to observe the rate $\ln\left(\frac{1+\epsilon}{1-\epsilon}\right)^{4N+1}$ for values of N greater than 2. The results are shown in Figure 3, where it is seen that the convergence rate for odd model orders $N = 1, 2$ are $\ln\left(\frac{1+\epsilon}{1-\epsilon}\right)^5, \approx \ln\left(\frac{1+\epsilon}{1-\epsilon}\right)^9$, respectively, as predicted.

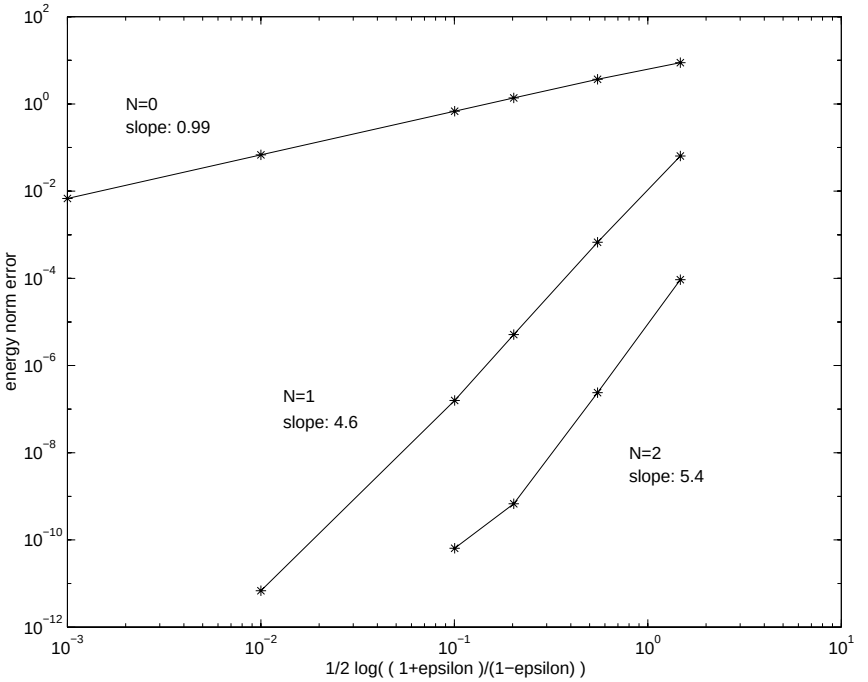


FIG. 2. *Convergence rates for* $f_\epsilon^+ = f_\epsilon^- = x(\pi - x)$ *for even model order* $N_e = 0, 1, 2$. *The gradients of each line are given in brackets below the model order. The results show that as* N_e *is increased from* 0 *to* 1, *the rate of convergence increases from* $\ln\left(\frac{1+\epsilon}{1-\epsilon}\right)$ *to* $\ln\left(\frac{1+\epsilon}{1-\epsilon}\right)^5$ *as predicted in the theory. For* $N_e = 2$ *there is a degradation in the expected convergence rate due to the limited compatibility of the data.*

3. Unsymmetric data: $f_\epsilon^+ = x(\pi - x), f_\epsilon^- = 0$. From (32), it can be seen that there are two parts to the error,

$$\ln\left(\frac{1+\epsilon}{1-\epsilon}\right)^{4N_e+1} \|\triangle^{N_e} f^e\|^2_{L^2(\omega)}, \ln\left(\frac{1+\epsilon}{1-\epsilon}\right)^{4N_o+1} \|\triangle^{N_o} f^o\|^2_{L^2(\omega)},$$

so that merely adding an odd basis function (so that $N_o = N_e + 1$) will not increase the convergence rate: both an odd and an even function need to be added to increase the rate of convergence. The constant of approximation, however, should decrease if only an odd function is added as then the error term has only an even part. Results are given in Figure 4. As can be seen, the numerical results support the theory, since the convergence rate for (0,1) and (1,2) is the same as for $N = 0$ and $(1, 1)\big(\approx \ln\left(\frac{1+\epsilon}{1-\epsilon}\right), \ln\left(\frac{1+\epsilon}{1-\epsilon}\right)^5\big)$, respectively.

5.3. Spherical shells. It follows from Theorem 3.1 that the functions ψ_N, for $N = 1, 2, \ldots$, for a spherical shell are given by

$$(1 + \epsilon r)^2 \frac{d\psi_N^{e,o}}{d\psi} = K_{N-1}^{e,o}, \int_{-1}^{1} \psi_N^{e,o} dr = 0,$$

where K_0^e, K_0^o are as in (24), and $K_N^{e,o}$ for $N = 1, 2, \ldots$ are given by

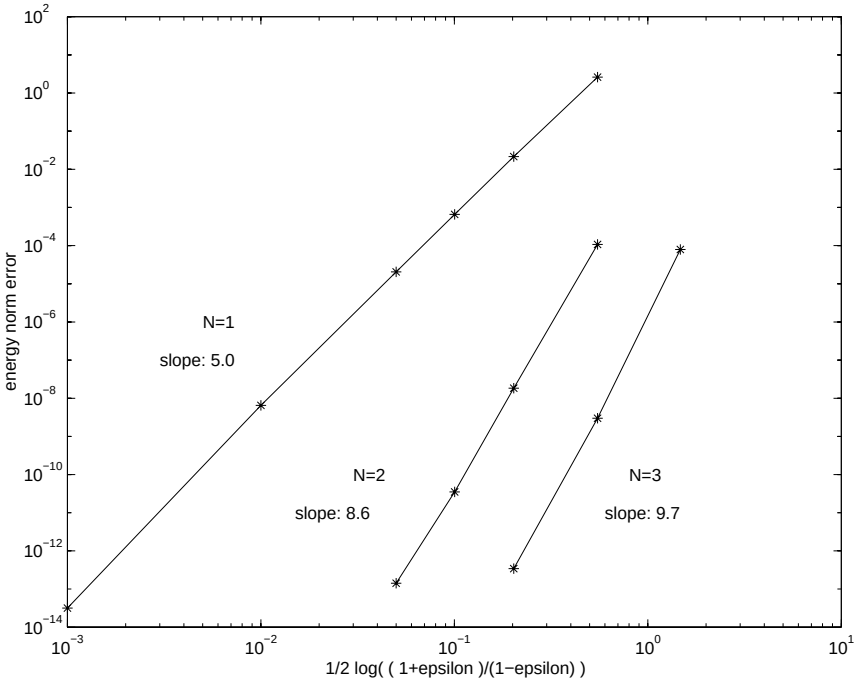


FIG. 3. *Convergence rates for* $f_\epsilon^+ = -f_\epsilon^- = x\pi(x^2-\pi^2) + x^3(\pi - x)$ *for odd model order* $N_o = 1, 2, 3$. *The gradients of each line are given in brackets below the model order. The results show that as* N_o *is increased from* 1 *to* 2, *the rate of convergence increases from* $\ln\left(\frac{1+\epsilon}{1-\epsilon}\right)^5$ *to* $\ln\left(\frac{1+\epsilon}{1-\epsilon}\right)^9$ *as predicted in the theory. For* $N_o = 3$ *there is a degradation in the convergence rate due to the limited compatibility of the data.*

$$(1+\epsilon r)^2 \frac{d^2 K_N^{e,o}}{dr^2} = K_{N-1}^{e,o}; K_N^{e,o}(\pm 1) = 0, \quad N = 1, 2, \ldots.$$

For $N = 0$, ψ_0^e is a constant. For $N > 0$, the solutions to this problem are of the form

$$\psi_N^e = \sum_{j=1}^{N} c_j \frac{(\ln(1+\epsilon r))^{j-1}}{1+\epsilon r} + \sum_{j=0}^{N} d_j (\ln(1+\epsilon r))^j,$$
$$\psi_N^o = \sum_{j=1}^{N} \tilde{c}_j \frac{(\ln(1+\epsilon r))^{j-1}}{1+\epsilon r} + \sum_{j=0}^{N-1} \tilde{d}_j (\ln(1+\epsilon r))^j.$$

For very small values of ϵ, these functions are practically linearly dependent, leading to numerical difficulties. Therefore, a different basis, spanning the same space, is employed:

$$\psi_j^e = \left(\ln\left(\frac{1+\epsilon r}{1-\epsilon}\right)\right)^j, \qquad j = 0, 1, 2 \ldots,$$
$$\psi_j^o = \frac{r}{1+\epsilon r}\left(\ln\left(\frac{1+\epsilon r}{1-\epsilon}\right)\right)^{j-1}, \qquad j = 1, 2, \ldots.$$

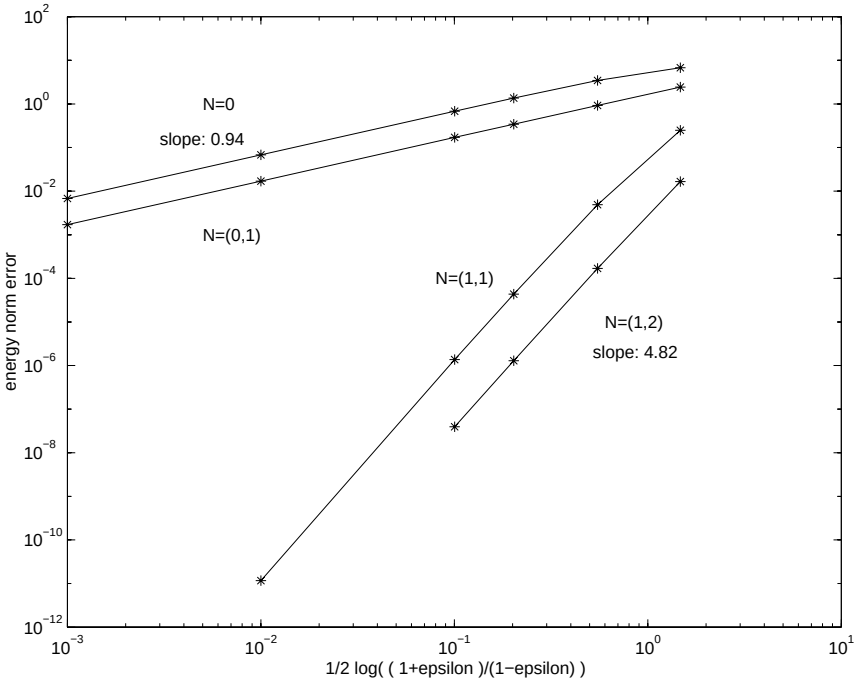


FIG. 4. *Convergence rates for* $f_\epsilon^+ = x(\pi - x), f_\epsilon^- = 0$ *for model orders* $0, (0,1), (1,1), (1,2)$. *The gradients of each line are given in brackets below the model order. The results show that if only* N_o *is increased (so that* $N_o = N_e + 1$*), then there is no increase in the convergence rate.*

5.3.1. Numerical examples. We present numerical results for a problem posed on a complete spherical shell with the following data:

$$f^+ = \left(2(1+\epsilon) - \frac{3}{(1+\epsilon)^4}\right)\sin^2(\theta)\sin(2\phi),$$
$$f^- = -\left(2(1-\epsilon) - \frac{3}{(1-\epsilon)^4}\right)\sin^2(\theta)\sin(2\phi).$$

With this choice of data the true solution u is given by

$$u = (r^2 + r^{-3})\sin^2(\theta)\sin(2\phi).$$

For model orders $N = 0, 1$ (that is, 0,(1,1)) we expect convergence rates in the thickness to be $\ln\left(\frac{1+\epsilon}{1-\epsilon}\right)^{4N+1}$. It can be seen in Figure 5 that this is indeed the case. For the intermediate model orders, there is more rapid convergence than expected, and to see how this happens one has to examine the error term for the constant through the thickness model. By a rearrangement of (21) it can be seen that

$$B(u - u_0, v)$$
$$= \int_\omega d\boldsymbol{\theta} \int_{-1}^{1} dr \frac{r}{2}((1+\epsilon)^2 f^+ + (1-\epsilon)^2 f^-) + \frac{1}{2}((1+\epsilon)^2 f^+ - (1-\epsilon)^2 f^-).$$

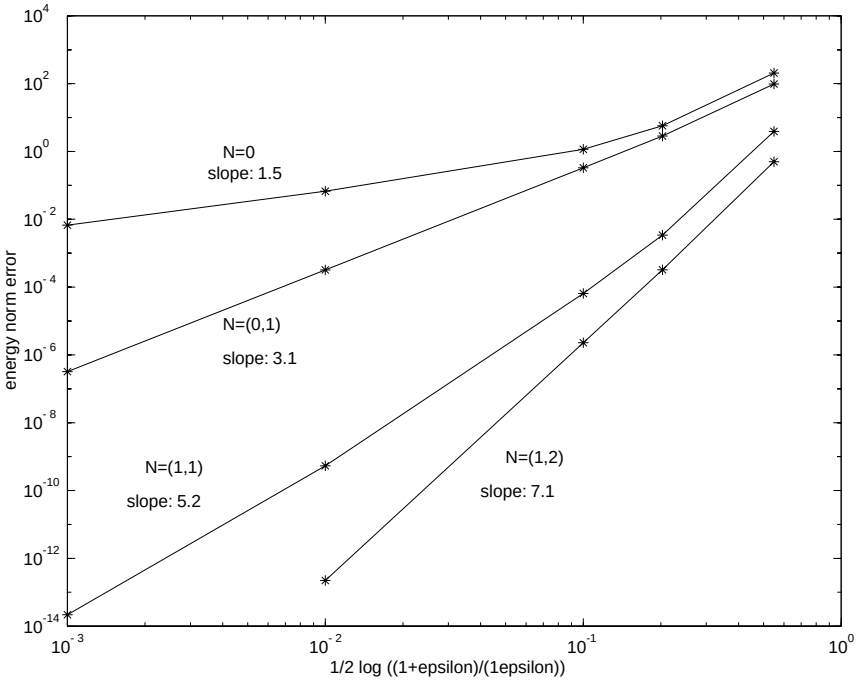


FIG. 5. *Convergence rates for the shell problem using the optimal functions. It can be seen that each time a new function is added to the approximation the convergence rates increase by approximately* $\ln\left(\frac{1+\epsilon}{1-\epsilon}\right)^2$.

Now the function ψ_1^o eliminates the $((1+\epsilon)^2 f^+ - (1-\epsilon)^2 f^-)$ term from $B(u-u_0, v)$. Noting that

$$\begin{aligned}
&(1+\epsilon)^2 f^+ + (1-\epsilon)^2 f^- \\
&= \left(2((1+\epsilon)^3 - (1-\epsilon)^3) - 3\left(\frac{1}{(1+\epsilon)^2} - \frac{1}{(1-\epsilon)^2}\right)\right)\sin^2(\theta)\sin(2\phi) \\
&= \left(2(3\epsilon + \epsilon^3) - \frac{12\epsilon}{(1+\epsilon)^2(1-\epsilon)^2}\right)\sin^2(\theta)\sin(2\phi) \\
&= \epsilon\left(3 + \epsilon^2 - \frac{4\epsilon}{(1+\epsilon)^2(1-\epsilon)^2}\right)\sin^2(\theta)\sin(2\phi) \\
&= O(\epsilon)
\end{aligned}$$

and

$$\begin{aligned}
&(1+\epsilon)^2 f^+ - (1-\epsilon)^2 f^- \\
&= \left(2((1+\epsilon)^3 + (1-\epsilon)^3) - 3\left(\frac{1}{(1+\epsilon)^2} + \frac{1}{(1-\epsilon)^2}\right)\right)\sin^2(\theta)\sin(2\phi) \\
&= \epsilon\left((2+6\epsilon^2) - \frac{6(1+\epsilon^2)}{(1+\epsilon)^2(1-\epsilon)^2}\right)\sin^2(\theta)\sin(2\phi) \\
&= O(1),
\end{aligned}$$

it can be seen that we are left with an $O(\epsilon)$ term, which then produces the high-order convergence rate ϵ^2 in the energy norm. Similarly, this process continues so that

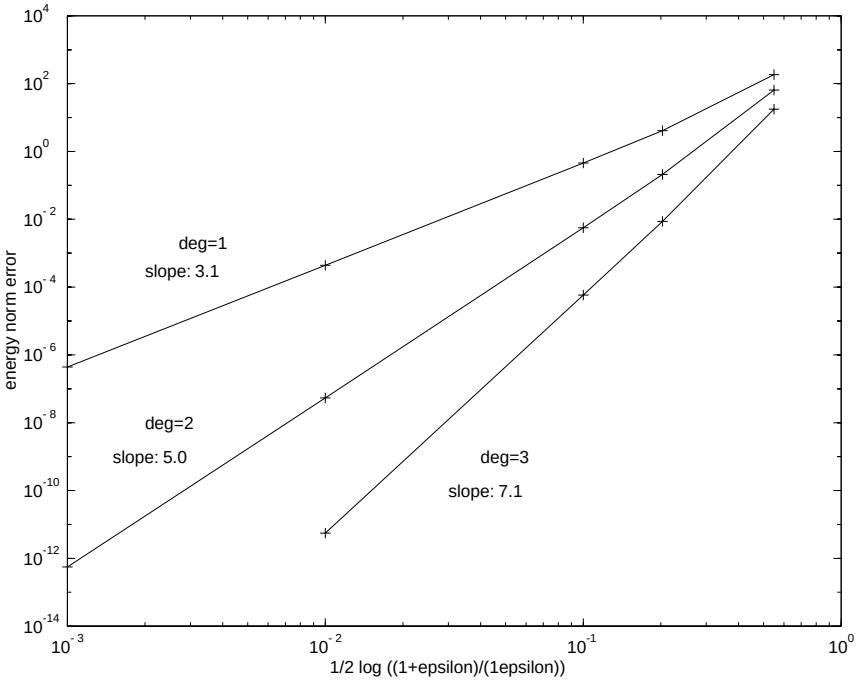


FIG. 6. *A plot showing the convergence rates for an integrated Legendre basis. It can be seen that the convergence rates in* $\ln\left(\frac{1+\epsilon}{1-\epsilon}\right)$ *are approximately the same as for the optimal basis.*

model order (1,2) has convergence of $O(\epsilon^7)$ rather than $O(\epsilon^5)$, as might be expected at first sight.

5.4. Comparison of optimal functions with polynomials. Previous work on dimensional reduction for spherical and cylindrical geometries has involved the use of polynomials in the radial direction. Surana and Abusaleh [25] and Surana and Bose [26] have recommended the use of polynomials in the thickness direction for heat conduction problems on both homogeneous and composite shells, Shen [24] has used polynomials in the radial direction for a Helmholtz problem on a spherical shell, and Cho and Oden [7, 8] have used polynomials to approximate in the radial direction in an elasticity problem on a cylindrical shell. In this section, we compare the error measured in the energy norm for the optimal functions $\{\psi_i\}$ derived earlier with polynomials for heat conduction on a homogeneous spherical shell.

Figures 6 and 7 show the error for the optimal functions compared with polynomials. For model order 0 both methods will be the same since both bases start with the constant function, but for higher model orders, then the constant $C(N)$ is smaller for the optimal functions for both "thin" and "thick" shells. The convergence rates with respect to the thickness are roughly equal for both bases as would be expected since for $\epsilon \ll 1$ the optimal functions behave approximately as polynomials. However, for relatively thick domains, the performance of the optimal functions derived here is superior to that of polynomials. It is worth observing that the cost of using the optimal functions is essentially the same as the cost of a polynomial basis, but the performance is more robust for thick domains.

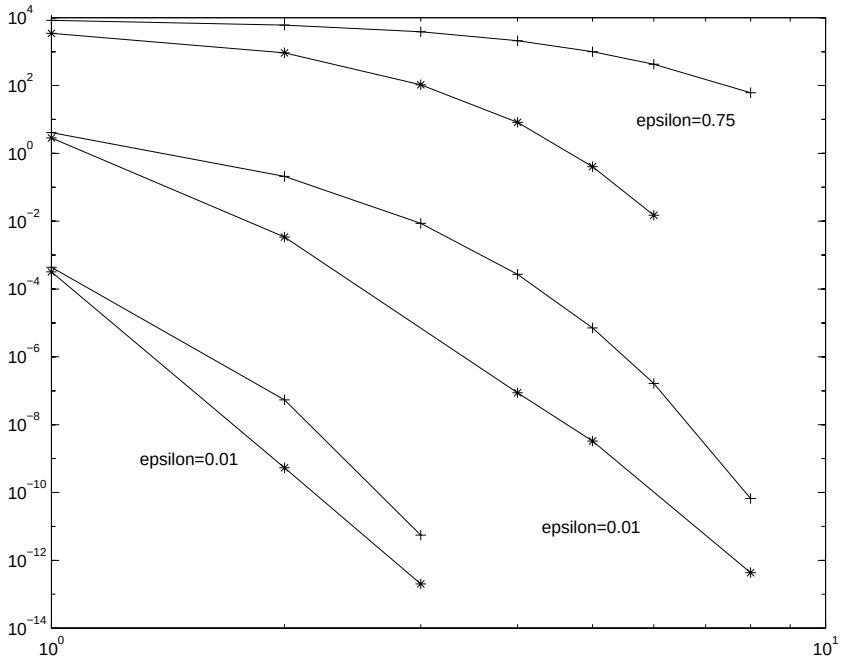


FIG. 7. *A comparison between the convergence rates for the optimal basis and the integrated Legendre basis as the model order (degree for polynomials) increases (for* $\epsilon = 0.75, 0.2, 0.01$*). The plots for the integrated Legendre basis are marked with "+" signs, where the optimal basis is marked with asterisks. It can be seen that the optimal basis is more accurate in each case.*

Acknowledgment. Mark Arnold gratefully acknowledges the support of an EPSRC research studentship.

REFERENCES

[1] R.A. ADAMS, *Sobolev Spaces*, Academic Press, New York, 1978.
[2] M. AINSWORTH, *A posteriori error estimation for fully discrete hierarchic models of elliptic boundary value problems on thin domains*, Numer. Math., 80 1998, pp. 325–362.
[3] M. AINSWORTH AND M. ARNOLD, *Computable error bounds for some simple dimensionally reduced models on thin domains*, IMA J. Numer. Anal., to appear.
[4] I. BABŬSKA AND C. SCHWAB, *A posteriori error estimation for hierarchic models of elliptic boundary value problems on thin domains*, SIAM J. Numer. Anal., 33 1996, pp. 221–246.
[5] I. BABŬSKA AND L. LI, *Hierarchic modelling of plates*, Comput. & Structures, 40 1991, pp. 419–430.
[6] M. BERCOVIER AND O. RICOU, *Dimensional Reduction of a Stokes Flow Using a Polynomial Approximation,* Technical Report 95-10, Hebrew University of Jerusalem, Israel, 1995.
[7] J.R. CHO AND J.T. ODEN, *A priori modelling error estimates of hierarchic models for plate and shell-like structures,* Math. Comput. Modelling, 23 1996, pp. 117–133.
[8] J.R. CHO AND J.T. ODEN, *Adaptive hpq-finite element methods of hierarchic models for plate and shell-like structures,* Comput. Methods Appl. Mech. Engrg., 136 1996, pp. 317–345.
[9] W. ECKHAUS, *Boundary layers in linear elliptic singular perturbation problems*, SIAM Rev., 14 1972, pp. 225–269.
[10] W. FREEDEN, T. GERVENS, AND M. SCHREINER, *Constructive Approximation on the Sphere with Applications to Geomathematics*, Clarendon Press, Oxford, UK, 1988.
[11] I.S. GRADSHTEIN AND I.M. RYZHIK, *Table of Integrals, Series and Products*, translated from the 4th Russian edition by Alan Jeffrey, Academic Press, Boston, 1994.
[12] A.E. GREEN AND W. ZERNA, *Theoretical Elasticity*, Oxford, UK, 1954.

[13] S. Jensen and I. Babŭska, *Dimensional reduction for nonlinear boundary value problems*, SIAM J. Numer. Anal., 25 1988, pp. 644–669.
[14] S. Jensen, *Adaptive dimensional reduction numerical solution of monotone quasilinear boundary value problems*, SIAM J. Numer. Anal., 29 1992, pp. 1294–1320.
[15] L.V. Kantorovich and V.I. Krylov, *Approximate Methods of Higher Analysis*, Noordhoff, Groningen, The Netherlands, 1958.
[16] R.P. Kanwal, *Linear Integral Equations: Theory and Technique*, Academic Press, New York, 1971.
[17] K.M. Liu and I. Babŭska, *Dimensional reduction for Helmholtz's equation on an unbounded domain*, Math. Models Methods Appl. Sci., 7 1997, pp. 81–111.
[18] K.M. Liu and I. Babuska, *Dimensional reduction for Helmholtz's equation on a bounded domain*, Numer. Math., 77 1997, pp. 501–533.
[19] C. Schwab, *Boundary layer resolution in hierarchic models of laminated composites*, RAIRO Modél Math. Anal. Numér., 28 1994, pp. 517–537.
[20] C. Schwab, *Boundary layer resolution in hierarchical plate modelling*, Math. Methods Appl. Sci., 18 1995, pp. 345–370.
[21] C. Schwab and S. Wright, *Boundary layers of hierarchical beam and plate models*, J. Elasticity, 38 1995, pp. 1–40.
[22] C. Schwab, *Hierarchic modelling in mechanics*, in Wavelets, Multilevel Methods and Elliptic PDEs, M. Ainsworth, J. Levesley, W. A. Light, and M. Marletta, eds., Oxford University Press, New York, 1997, pp. 85–160.
[23] Jie Shen, *Efficient spectral–Galerkin methods* III*: Polar and cylindrical geometries*, SIAM J. Sci. Comput., 18 1997, pp. 1583–1604.
[24] Jie Shen, *Efficient spectral–Galerkin Methods* IV*: Spherical geometries*, SIAM J. Sci Comput., 20 (1999), pp. 1438–1455.
[25] K.S. Surana and G. Abusaleh, *Axisymmetric shell elements for heat-conduction with p-approximation in the thickness direction*, Comput. & Structures, 33 1989, pp. 689–705.
[26] K.S. Surana and A. Bose, *p-version piecewise hierarchical axisymmetric shell element for heat conduction in laminated composites*, Comput. & Structures, 48 1993, pp. 33–49.
[27] M. Vogelius and I. Babŭska, *On a dimension reduction method* I. *The optimal selection of basis functions*, Math. Comp., 37 1981, pp. 31–46.
[28] M. Vogelius and I. Babŭska, *On a dimension reduction method* II. *Some approximation-theoretic results*, Math. Comp., 37 1981, pp. 47–68.
[29] E. Zauderer, *Partial Differential Equations*, John Wiley, New York, 1989.

SIAM J. SCI. COMPUT.
Vol. 22, No. 2, pp. 704–723
© 2000 Society for Industrial and Applied Mathematics

SCHUR COMPLEMENT SYSTEMS IN THE MIXED-HYBRID FINITE ELEMENT APPROXIMATION OF THE POTENTIAL FLUID FLOW PROBLEM*

J. MARYŠKA†, M. ROZLOŽNÍK‡, AND M. TŮMA†

Abstract. The mixed-hybrid finite element discretization of Darcy's law and continuity equation describing the potential fluid flow problem in porous media leads to a symmetric indefinite linear system for the pressure and velocity vector components. As a method of solution the reduction to three Schur complement systems based on successive block elimination is considered. The first and second Schur complement matrices are formed eliminating the velocity and pressure variables, respectively, and the third Schur complement matrix is obtained by elimination of a part of Lagrange multipliers that come from the hybridization of a mixed method. The structural properties of these consecutive Schur complement matrices in terms of the discretization parameters are studied in detail. Based on these results the computational complexity of a direct solution method is estimated and compared to the computational cost of the iterative conjugate gradient method applied to Schur complement systems. It is shown that due to special block structure the spectral properties of successive Schur complement matrices do not deteriorate and the approach based on the block elimination and subsequent iterative solution is well justified. Theoretical results are illustrated by numerical experiments.

Key words. sparse linear systems, finite element matrices, preconditioned conjugate residuals, potential fluid flow problem, indefinite linear systems

AMS subject classifications. Primary, 65F10, 65F50, 65N22; Secondary, 15A06

PII. S1064827598339608

1. Introduction.

Let Ω be a bounded domain in $\mathcal{R}^3$ with a Lipschitz continuous boundary $\partial\Omega$. The potential fluid flow in saturated porous media can be described by the velocity $\mathbf{u}$ using Darcy's law and by the continuity equation for incompressible flow

$$\mathbf{A}\mathbf{u} = -\nabla p, \quad \nabla \cdot \mathbf{u} = q, \tag{1.1}$$

where p is the piezometric potential (fluid pressure), $\mathbf{A}$ is a symmetric and uniformly positive definite second rank tensor of the hydraulic resistance of medium with $[\mathbf{A}(\mathbf{x})]_{ij} \in L^\infty(\Omega)$ for all $i, j \in \{1, 2, 3\}$, and q represents the density of potential sources in the medium. The boundary conditions are given by

$$p = p_D \quad \text{on} \quad \partial\Omega_D, \quad \mathbf{u} \cdot \mathbf{n} = u_N \quad \text{on} \quad \partial\Omega_N, \tag{1.2}$$

where $\partial\Omega = \overline{\partial\Omega_D} \cup \overline{\partial\Omega_N}$ are such that $\partial\Omega_D \neq \emptyset$, $\partial\Omega_D \cap \partial\Omega_N = \emptyset$, and $\mathbf{n}$ is the outward normal vector defined (almost everywhere) on the boundary $\partial\Omega$.

Assume that the domain Ω is a polyhedron and it is divided into a collection of subdomains such that every subdomain is a trilateral prism with vertical faces and

*Received by the editors June 1, 1998; accepted for publication (in revised form) January 17, 2000; published electronically August 9, 2000. This work was supported by the Grant Agency of the Czech Republic under grant 201/98/P108 and by the Grant Agency of the Academy of Sciences of the Czech Republic under grant A2030706.

http://www.siam.org/journals/sisc/22-2/33960.html

†Institute of Computer Science, Academy of Sciences of the Czech Republic, Pod vodárenskou věží 2, 182 07 Prague 8, Czech Republic (maryska@cs.cas.cz, tuma@cs.cas.cz).

‡Seminar for Applied Mathematics, Swiss Federal Institute of Technology (ETH) Zurich, ETH-Zentrum, CH-8092 Zurich, Switzerland (miro@sam.math.ethz.ch).

general nonparallel bases (see, e.g., [11], [14], or [15]). We will denote the discretization of the domain Ω by $\mathcal{E}_h$ and assume an uniform regular mesh with the discretization parameter h. Denote also the collection of all faces of elements $e \in \mathcal{E}_h$ which are not adjacent to the boundary $\partial\Omega_D$ by $\Gamma_h = \cup_{e\in\mathcal{E}_h}\partial e - \partial\Omega_D$ and introduce the set of interior faces $int(\Gamma_h) = \Gamma_h - \partial\Omega_N$.

We consider the following low-order finite element approximation. Let

$$\mathbf{RT}^0(e) = \left\{\mathbf{v}^e;\ \mathbf{v}^e(\mathbf{x}) = \sum_{j=1}^{5} \nu_j \mathbf{v}_j^e(\mathbf{x})\ \forall \mathbf{x} = (x_1, x_2, x_3) \in e\right\} \tag{1.3}$$

be the space spanned by the linearly independent basis functions $\mathbf{v}_j^e(\mathbf{x}),\ j = 1, \ldots, 5$ defined on the element $e \in \mathcal{E}_h$ in the form

$$\mathbf{v}_j^e(\mathbf{x}) = k_j^e \begin{pmatrix} 0 \\ 0 \\ x_3 - \alpha_{j3}^e \end{pmatrix},\ j = 1, 2, \quad \mathbf{v}_j^e(\mathbf{x}) = k_j^e \begin{pmatrix} x_1 - \alpha_{j1}^e \\ x_2 - \alpha_{j2}^e \\ \beta_j^e x_3 - \alpha_{j3}^e \end{pmatrix},\ j = 3, 4, 5, \tag{1.4}$$

and such that they are orthonormal with respect to the set of functionals

$$\mathcal{F}_k(\mathbf{v}_j^e) = \int_{f_k^e} \mathbf{n}_k^e \cdot \mathbf{v}_j^e dS = \delta_{jk},\ j, k = 1, \ldots, 5. \tag{1.5}$$

Here f_k^e denotes the kth face of the element $e \in \mathcal{E}_h$ and $\mathbf{n}_k^e = (n_{k,1}^e, n_{k,2}^e, n_{k,3}^e)$ is the outward normal vector with respect to the face f_k^e. The velocity function $\mathbf{u}$ will then be approximated by vector functions linear on every element $e \in \mathcal{E}_h$ from the Raviart–Thomas space

$$\mathbf{RT}_{-1}^0(\mathcal{E}_h) = \left\{\mathbf{v}_h \in \mathbf{L}^2(\Omega);\ \mathbf{v}_h|^e \in \mathbf{RT}^0(e) \quad \forall e \in \mathcal{E}_h\right\}, \tag{1.6}$$

where $\mathbf{v}_h|^e$ denotes the restriction of a function $\mathbf{v}_h$ onto the element $e \in \mathcal{E}_h$. Further denote the space of constant functions on each element $e \in \mathcal{E}_h$ by $M^0(e)$ and denote the space of constant functions on each face $f \in \Gamma_h$ by $M^0(f)$. The piezometric potential p will be approximated by the space which consists of elementwise constant functions

$$M_{-1}^0(\mathcal{E}_h) = \{\phi_h \in L^2(\Omega);\ \phi_h|^e \in M^0(e) \quad \forall e \in \mathcal{E}_h\}, \tag{1.7}$$

where $\phi_h|^e$ is the restriction of a function ϕ_h onto element $e \in \mathcal{E}_h$. The Lagrange multipliers coming from the hybridization of a method will be approximated by the space of all functions constant on every face from Γ_h,

$$M_{-1}^0(\Gamma_h) = \{\mu_h \in L^2(\Omega);\ \Gamma_h \to R;\ \mu_h|^f \in M^0(f) \quad \forall f \in \Gamma_h\}. \tag{1.8}$$

Here $\mu_h|^f$ denotes the restriction of a function μ_h onto the face $f \in \Gamma_h$. Analogously we introduce the spaces $M_{-1}^0(\partial\Omega_D)$ and $M_{-1}^0(\partial\Omega_N)$ as the spaces of functions constant on every face from $\cup_{e\in\mathcal{E}_h}\partial e \cap \partial\Omega_D$ and $\Gamma_h \cap \partial\Omega_N$, respectively. The detailed description of the spaces that we use can be found in [14] (see also [11] or [15]).

The Raviart–Thomas approximation of the mixed-hybrid formulation for the problem (1.1), and (1.2) reads as follows (see [4]):

Find $(\mathbf{u}_h, p_h, \lambda_h) \in \mathbf{RT}^0_{-1}(\mathcal{E}_h) \times M^0_{-1}(\mathcal{E}_h) \times M^0_{-1}(\Gamma_h)$ such that

$$\sum_{e\in\mathcal{E}_h} \{(\mathbf{A}\mathbf{u}_h, \mathbf{v}_h)_{0,e} - (p_h, \nabla\cdot\mathbf{v}_h)_{0,e} + \langle\lambda_h, \mathbf{n}^e\cdot\mathbf{v}_h\rangle_{\partial e\cap\Gamma_h}\}$$

$$= \langle p_{D,h}, \mathbf{n}^e\cdot\mathbf{v}_h\rangle_{\partial e\cap\partial\Omega_D} \quad \forall \mathbf{v}_h \in \mathbf{RT}^0_{-1}(\mathcal{E}_h), \tag{1.9}$$

$$-\sum_{e\in\mathcal{E}_h} (\nabla\cdot\mathbf{u}_h, \phi_h)_{0,e} = -(q_h, \phi_h)_{0,\Omega} \quad \forall \phi_h \in M^0_{-1}(\mathcal{E}_h), \tag{1.10}$$

$$\sum_{e\in\mathcal{E}_h} \langle\mathbf{n}^e\cdot\mathbf{u}_h, \mu_h\rangle_{\partial e} = \langle u_{N,h}, \mu_h\rangle_{\partial e\cap\partial\Omega_N} \quad \forall \mu_h \in M^0_{-1}(\Gamma_h), \tag{1.11}$$

where $p_{D,h}$ and $u_{N,h}$ are approximations to the functions p_D and u_N on the spaces $M^0_{-1}(\partial\Omega_D)$ and $M^0_{-1}(\partial\Omega_N)$, respectively, and where the function q is approximated by $q_h \in M^0_{-1}(\mathcal{E}_h)$. For other details we refer to [14] or [11].

Further denote by $NE = |\mathcal{E}_h|$ the number of elements, by $NIF = |int(\Gamma_h)|$ the number of interior interelement faces, and the number of faces with the prescribed Neumann boundary conditions in the discretization by $NNC = |\partial\Omega_N|$. Let $e_i \in \mathcal{E}_h$, $i = 1, \ldots, NE$, be some numbered ordering of the set of prismatic elements and f_k, $k = 1, \ldots, NIF + NNC$, be the ordering of the set of non-Dirichlet faces from Γ_h. For every element $e_i \in \mathcal{E}_h$ we denote by NIF_{e_i} the number of interior interelement faces and by NNC_{e_i} the number of faces with Neumann boundary conditions imposed on the element e_i. Let the finite-dimensional space $\mathbf{RT}^0_{-1}(\mathcal{E}_h)$ be spanned by $NA = 5 \times NE$ linearly independent basis functions $\mathbf{v}_j$, $j = 1, \ldots, NA$ from the definition (1.6), let the space $M^0_{-1}(\mathcal{E}_h)$ be spanned by NE linearly independent basis functions ϕ_i, $i = 1, \ldots, NE$, and finally the space $M^0_{-1}(\Gamma_h)$ be spanned by $NIF+NNC$ linearly independent basis functions μ_k, $k = 1, \ldots, NIF + NNC$. From this Raviart–Thomas approximation we obtain the system of linear algebraic equations in the form

$$\begin{pmatrix} A & B & C \\ B^T & & \\ C^T & & \end{pmatrix} \begin{pmatrix} u \\ p \\ \lambda \end{pmatrix} = \begin{pmatrix} q_1 \\ q_2 \\ q_3 \end{pmatrix}, \tag{1.12}$$

where $u = (u_1, \ldots, u_{NA})^T$, $p = (p_1, \ldots, p_{NE})^T$, $\lambda = (\lambda_1, \ldots, \lambda_{NIF+NNC})^T$ are unknowns, the symmetric positive definite matrix block $A \in \mathcal{R}^{NA,NA}$ is given by the terms $(\mathbf{A}\mathbf{v}_i, \mathbf{v}_j)_{0,\Omega}$, the outdiagonal block $B \in \mathcal{R}^{NA,NE}$ by $-(\nabla\cdot\mathbf{v}_i, 1)_{0,e_j}$, and the block $C \in \mathcal{R}^{NA,NIF+NNC}$ by $\langle\mathbf{n}_k\cdot\mathbf{v}_i, 1\rangle_{f_k}$. Here $\mathbf{n}_k$ is the outward normal vector to with respect to the face $f_k \in \Gamma_h$ (see [11] and [14]).

Let us denote the system matrix in (1.12) by A. The symmetric matrix A is indefinite due to the zero diagonal block of dimension $NBC = NE + NIF + NNC$. The structure of nonzero elements in the matrix from a small model problem can be seen in Figure 1. Partition the submatrix C in A as $(C_1\ C_2)$, where $C_1 \in \mathcal{R}^{NA,NIF}$ corresponds to the interior interelement faces in the discretized domain and $C_2 \in \mathcal{R}^{NA,NNC}$ is the face-condition incidence matrix corresponding to the element faces with Neumann boundary conditions. Note that every column of C_1 contains only two nonzero entries equal to 1. The singular values of C_1 are all equal to $\sqrt{2}$ and the matrix block C_2 has orthogonal columns. Moreover, the whole matrix block C also has singular values equal to $\sqrt{2}$ or 1. The matrix B has a special structure. The

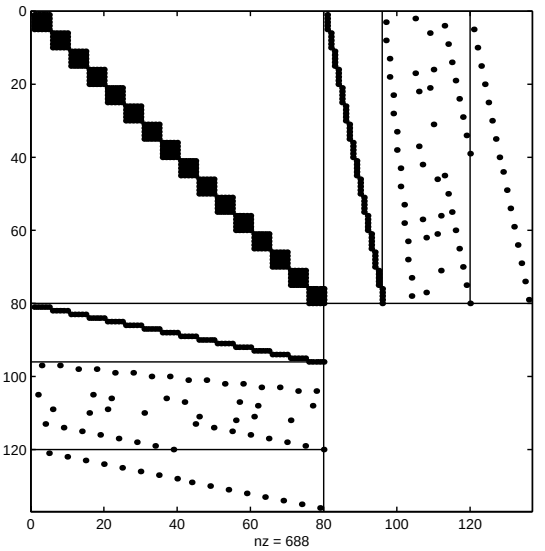


FIG. 1. *Structural pattern of the matrix obtained from mixed hybrid finite element approximation of a model problem with* $h = 1/2$ *(to be discussed in section* 5*).*

nonzero elements correspond to the face-element incidence matrix with values equal to -1. Thus all singular values of the matrix B are equal to $\sqrt{5}$. (The matrix B is, up to the normalization coefficients, orthogonal.)

It is easy to see from the definition of approximation spaces (see [14] or [15]) that the symmetric positive definite block is 5×5 block diagonal, and it was shown in [15] that the spectrum of the matrix block A satisfies

$$\sigma(A) \subset [c_1 \sqrt[3]{NE}, c_2 \sqrt[3]{NE}], \tag{1.13}$$

where c_1 and c_2 are positive constants independent of the discretization parameters and dependent on the domain and the tensor $\mathbf{A}$. It is also easy to see that the system matrix A in (1.12) is nonsingular if and only if the block $(B\ C)$ has a full column rank. Clearly, if the condition $\partial\Omega_D = \emptyset$ holds (all boundary conditions are Neumann conditions), then the matrix block $(B\ C)$ is singular due to the fact that all sums of row elements are zero. In other words, the function p is unique up to a constant function in the case $\partial\Omega_D = \emptyset$. Assuming $\partial\Omega_D \neq \emptyset$ it follows from the analysis presented in [15] that there exist positive constants c_3 and c_4 such that for the singular values of the matrix block $(B\ C)$ we have

$$sv(B\ C) \subset [c_3/\sqrt[3]{NE}, c_4]. \tag{1.14}$$

Moreover, for eigenvalues of the whole symmetric indefinite matrix A it follows asymptotically $(h \to 0)$

$$\sigma(\mathsf{A}) \subset [-c_5/\sqrt[3]{NE}, -c_6/NE] \cup [c_7 \sqrt[3]{NE}, c_8 \sqrt[3]{NE}], \tag{1.15}$$

where c_5, c_6, c_7, and c_8 are positive constants independent of system parameters.

In this paper, for solving the symmetric indefinite systems (1.12), the successive reduction to Schur complement systems is proposed. We consider three successive

Schur complement systems arising during the block elimination of unknowns which correspond to matrix blocks A, B, and C_2, respectively, or in other words, which correspond to the elimination of the velocity variables u, the pressure variables p, and of a part of the Lagrange multipliers λ. While the concept of reduction to the first and second Schur complement systems is well known as a static condensation (described, e.g., in [4], section V, or in [11]), the proposed reduction to the third Schur complement system seems to be new. The main contribution of the paper consists in a detailed investigation of the structure of nonzero entries and the spectral properties of the Schur complement matrices. This enables thorough complexity analysis of the direct or iterative solution of corresponding Schur complement systems. A brief analysis of the structure of the first Schur complement matrix can be found in [4], as well as a straightforward observation that its principal leading block is a diagonal. Here we extend this analysis and discuss the mutual relation between the number of nonzero entries in the first Schur complement matrix and the number of nonzeros in the system matrix (1.12). We show further that no fill-in occurs during the process of reduction to the second and third Schur complement system. Moreover, we prove that the number of nonzeros in both these two Schur complements is always less than the number of nonzeros in (1.12). It is shown also that the spectral properties of matrices in such Schur complement systems do not deteriorate during the successive elimination. Thus an approach based on the block reduction and subsequent iterative solution is well justified.

The outline of the paper is as follows. In section 2, we examine the structural pattern of resulting Schur complement matrices and give estimates for their number of nonzero elements in terms of the discretization parameters listed above. Section 3 is devoted to the solution of the whole indefinite system (1.12) via three Schur complement reductions and subsequent direct solution. Using the graph theoretical results we give the asymptotic bound of the computational complexity for the Cholesky decomposition method applied to the third Schur complement system. In section 4, we concentrate on the spectral properties of the Schur complement system matrices. The theoretical convergence rate of the iterative conjugate gradient-type method in terms of the discretization parameters is estimated. The asymptotic bounds for the computational work of the iterative solution are given. Section 5 contains some numerical experiments illustrating the previously developed theoretical results. Finally, we give some concluding remarks and mention some open questions for our future work.

2. Structural properties of the Schur complement matrices. In this section we take a closer look at the discretized indefinite system and corresponding Schur complements, and we extend the brief analysis from [4]. There are several possibilities for the choice of a block ordering in the consecutive elimination. We shall concentrate on the block ordering which seems to be the most natural and efficient from the point of view of solving the final Schur complement system by a direct solver or by a conjugate gradient-type method. The same ordering for the elimination of the first two blocks was used also in [4, p. 178–181] or in [11]. Note that the static condensation is not the only way to form the successive Schur complements. For example, in [17] the case of the Raviart–Thomas discretization for the closely related nodal methods was studied and reduction to a different second Schur complement system was discussed.

The following simple result gives the number of nonzero elements in the triangular part of the matrix A. By the triangular part of a matrix M we mean its upper (lower) strict triangle + diagonal. We will deal only with the structural nonzero elements here; we do not take into account accidental cancellations and possible initial zero

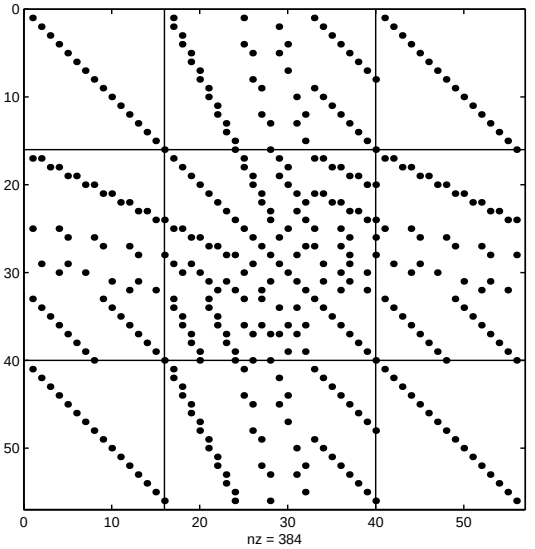


FIG. 2. *Structural pattern of the Schur complement matrix* A/A *for* A *from Figure* 1.

values of the tensor of hydraulic permeability. By the structure of a matrix M we mean $Struct(M) = \{(i,j)|M_{ij} \neq 0\}$.

LEMMA 2.1. *The number of nonzeros in the triangular part of* A *is given by*

$$|sym(A)| = 20NE + 2NIF + NNC. \tag{2.1}$$

Proof. The triangular part of A has $15NE$ nonzeros, the block B contributes by $5NE$ nonzeros, C_1 has $2NIF$ nonzeros, and C_2 contains NNC nonzeros. □

The symmetric positive definite matrix block A in (1.12) is block-diagonal, each 5×5 block corresponds to certain element in the discretization of the domain. Therefore it is straightforward to eliminate the velocity variables u and to obtain the first Schur complement system with the matrix

$$\mathsf{A}/A = -\begin{pmatrix} B^T \\ C_1^T \\ C_2^T \end{pmatrix} A^{-1} \begin{pmatrix} B & C_1 & C_2 \end{pmatrix} = -\begin{pmatrix} A_{11} & A_{12} & A_{13} \\ A_{12}^T & A_{22} & A_{23} \\ A_{13}^T & A_{23}^T & A_{33} \end{pmatrix}.$$

The structure of the matrix A/A for our example problem is shown in Figure 2. For details we also refer to [4, p. 180–181] or [11]. For the number of nonzeros in the matrix A/A we can show the following result.

LEMMA 2.2. *The number of nonzeros in the triangular part of the Schur complement matrix* A/A *is equal to*

$$|sym(\mathsf{A}/A)| = NE + \frac{3}{2}NIF + \frac{3}{2}NNC + \frac{1}{2}\sum_{i \in \mathcal{E}_h}(NIF_i + NNC_i)^2 + \frac{1}{2}\sum_{i \in \mathcal{E}_h}\sum_{j \in Adj(i)} NIF_j.$$

Proof. Clearly, $|sym(A_{11})| = NE$ and $|A_{12}| = |B^T A^{-1} C_1| = 2NIF$. Note that the fill-in for $A^{-1}C_1$ is considerably higher (it is equal to $10NIF$). Further, $|A_{13}| = NNC$ and $|sym(A_{33})| = \frac{1}{2}\sum_{i \in \mathcal{E}_h} NNC_i(NNC_i + 1) = \frac{1}{2}NNC + \frac{1}{2}\sum_{i \in \mathcal{E}_h} NNC_i^2$. The number

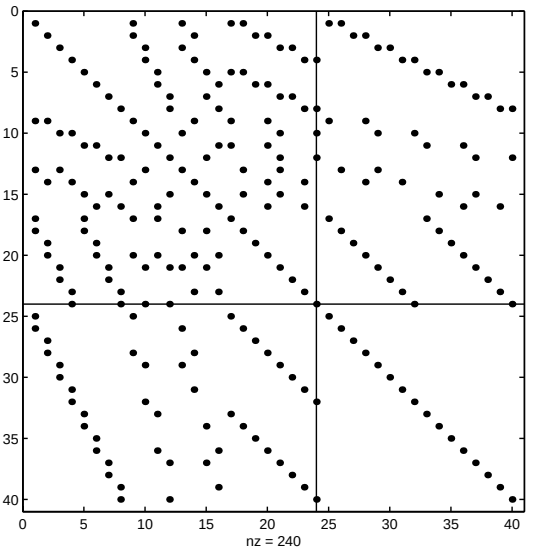


FIG. 3. *Structural pattern of the Schur complement matrix* $(-\mathsf{A}/A)/A_{11}$ *for* A *from Figure* 1.

of nonzeros in A_{23} is equal to $\sum_{i\in\mathcal{E}_h} NNC_i\, NIF_i$. Finally, note that

$$Struct(A_{22}) = Struct(C_1^T A^{-1} C_1) = Struct(C_1^T B B^T A^{-1} C_1) = Struct(C_1^T B B^T C_1).$$

Observe that the directed graph of the matrix $B^T C_1$ has the set of arcs

$$E_{B^T C_1} = \{(i,f) \in \mathcal{E}_h \times int(\Gamma_h) |\ f \text{ is an interior face of } i\}.$$

The undirected graph of $C_1^T B B^T C_1$ therefore expresses element-element adjacency relation based on the connectivity through the interior faces inside the domain. It follows that

$$|A_{22}| = \sum_{f\in int(\Gamma_h)} (NIF_{e(f)} + NIF_{\bar{e}(f)} - 1) = \sum_{i\in\mathcal{E}_h} NIF_i(NIF_i - 1) + \sum_{i\in\mathcal{E}_h} \sum_{j\in Adj(i)} NIF_j,$$

where $e(f)$ and $\bar{e}(f)$ are the two elements from $\mathcal{E}_h$ such that $e(f) \cap \bar{e}(f) = \{f\}$. Therefore, considering the relation $2NIF = \sum_{i\in\mathcal{E}_h} NIF_i$ we obtain $|sym(A_{22})| = \frac{1}{2}(|A_{22}| + NIF) = \frac{1}{2}\sum_{i\in\mathcal{E}_h}(NIF_i)^2 + \frac{1}{2}\sum_{i\in\mathcal{E}_h}\sum_{j\in Adj(i)} NIF_j - \frac{1}{2}NIF$. Putting all the partial sums together we get the desired result. □

Consider now the second Schur complement matrix

$$(-\mathsf{A}/A)/A_{11} = -\begin{pmatrix} A_{12}^T \\ A_{13}^T \end{pmatrix} A_{11}^{-1} \begin{pmatrix} A_{12} & A_{13} \end{pmatrix} + \begin{pmatrix} A_{22} & A_{23} \\ A_{23}^T & A_{33} \end{pmatrix} = \begin{pmatrix} B_{11} & B_{12} \\ B_{12}^T & B_{22} \end{pmatrix}.$$

The structure of $(-\mathsf{A}/A)/A_{11}$ for our example matrix is shown in Figure 3. The matrix block A_{11} in the first Schur complement matrix A/A is diagonal [4], [11]. The following result shows that it is worth to form the Schur complement matrix $(-\mathsf{A}/A)/A_{11}$ from the matrix A/A since no further fill-in appears during the elimination of the block A_{11} corresponding to pressure variables p and so we can further reduce the dimension of the system.

THEOREM 2.1.

$$Struct\begin{pmatrix} B_{11} & B_{12} \\ B_{12}^T & B_{22} \end{pmatrix} = Struct\begin{pmatrix} A_{22} & A_{23} \\ A_{23}^T & A_{33} \end{pmatrix}.$$

Proof. We have the following structural equivalences:

$$\begin{aligned} Struct(B_{11}) &= Struct(A_{22}) + Struct(A_{12}^T A_{11}^{-1} A_{12}) \\ &= Struct(A_{22}) + Struct(C_1^T A^{-1} B A_{11} B^T A^{-1} C_1) \\ &= Struct(A_{22}) + Struct(C_1^T A^{-1} B B^T A^{-1} C_1) \\ &= Struct(A_{22}) + Struct(C_1^T B B^T C_1) = Struct(A_{22}), \end{aligned}$$

$$\begin{aligned} Struct(B_{12}) &= Struct(A_{23}) + Struct(A_{12}^T A_{11}^{-1} A_{13}) \\ &= Struct(A_{23}) + Struct(C_1^T A^{-1} B B^T A^{-1} C_2) \\ &= Struct(A_{23}) + Struct(C_1^T B B^T C_2) \\ &= Struct(A_{23}) + Struct(C_1^T A^{-1} C_2) = Struct(A_{23}), \end{aligned}$$

$$\begin{aligned} Struct(B_{22}) &= Struct(A_{33}) + Struct(A_{13}^T A_{11}^{-1} A_{13}) \\ &= Struct(A_{33}) + Struct(C_2^T A^{-1} B A_{11}^{-1} B^T A^{-1} C_2) \\ &= Struct(A_{33}) + Struct(C_2^T A^{-1} C_2) = Struct(A_{33}). \end{aligned}$$ ∎

From previous theorem it is also easy to see that right lower block B_{22} is block-diagonal with blocks of varying size (depending on number of faces with Neumann conditions in each element) each corresponding to a certain element in the discretization. So in the following we will consider the third Schur complement matrix

$$((-\mathsf{A}/A)/A_{11})/B_{22} = B_{11} - B_{12} B_{22}^{-1} B_{12}^T$$

induced by the block B_{22} in the matrix $(-\mathsf{A}/A)/A_{11}$. We can prove a similar result to the one given in Theorem 2.1. Therefore, the Schur complement system with the matrix $-(\mathsf{A}/A)/A_{11}$ can be reduced to the Schur complement matrix $((-\mathsf{A}/A)/A_{11})/B_{22}$ of dimension equal to *NIF* without inducing any additional fill-in. Moreover, this can be done using incomplete factorization procedures.

THEOREM 2.2.

$$Struct(B_{11} - B_{12} B_{22}^{-1} B_{12}^T) = Struct(A_{22}).$$

Proof. Using Theorem 2.1 we get

$$\begin{aligned} Struct(B_{11} - B_{12} B_{22}^{-1} B_{12}^T) &= Struct(B_{11}) + Struct(B_{12} B_{22}^{-1} B_{12}^T) \\ &= Struct(A_{22}) + Struct(A_{23} A_{33}^{-1} A_{23}^T) \\ = Struct(A_{22}) + Struct(C_1^T A^{-1} C_2 C_2^T A^{-1} C_2 C_2^T A^{-1} C_1). \end{aligned}$$

Since $Struct(C_1^T A^{-1} C_2 C_2^T A^{-1} C_2 C_2^T A^{-1} C_1) \subset Struct(A_{22})$ (with equality only in the trivial singular case with $|\mathcal{E}_h| = 1$ and $NNC = 5$) we get the desired result $Struct(B_{11} - B_{12} B_{22}^{-1} B_{12}^T) = Struct(A_{22})$. ∎

The following simple corollary gives the number of nonzero elements in the second and third Schur complement matrices $(-\mathsf{A}/A)/A_{11}$ and $((-\mathsf{A}/A)/A_{11})/B_{22}$. We shall use these results later.

COROLLARY 2.1. *The number of nonzeros in the triangular part of* $(-\mathsf{A}/A)/A_{11}$ *is given by*

$$|sym((-\mathsf{A}/A)/A_{11})| = \\ = \frac{1}{2}\sum_{i\in\mathcal{E}_h}(NIF_i + NNC_i)^2 + \frac{1}{2}\sum_{i\in\mathcal{E}_h}\sum_{j\in Adj(i)} NIF_j + \frac{1}{2}(NNC - NIF) \tag{2.2}$$

and the number of nonzeros in the triangular part of $(-\mathsf{A}/A)/A_{11})/B_{22}$ *is given by*

$$|sym(((-\mathsf{A}/A)/A_{11})/B_{22})| = \frac{1}{2}\sum_{i\in\mathcal{E}_h} NIF_i^2 + \frac{1}{2}\sum_{i\in\mathcal{E}_h}\sum_{j\in Adj(i)} NIF_j - \frac{1}{2}NIF. \tag{2.3}$$

Apart from explicit assembly of the Schur complement matrices or using them implicitly there is another possibility which may be considered—keeping the Schur complements in factorized form. Consider the following decomposition:

$$\begin{aligned} \mathsf{A}/A &= -[L_A^{-1}\begin{pmatrix} B & C_1 & C_2\end{pmatrix}]^T[L_A^{-1}\begin{pmatrix} B & C_1 & C_2\end{pmatrix}] \\ &= \begin{pmatrix}\hat{B} & \hat{C}_1 & \hat{C}_2\end{pmatrix}^T\begin{pmatrix}\hat{B} & \hat{C}_1 & \hat{C}_2\end{pmatrix}, \end{aligned} \tag{2.4}$$

where $A = L_A L_A^T$. In contrast to the previous case, where the local numbering of the faces corresponding to the individual elements did not play a role, this is not the case now.

THEOREM 2.3. *Assume that all the elements within the diagonal blocks of the matrix* A *are nonzero. The fill-in in* $\begin{pmatrix}\hat{B} & \hat{C}_1 & \hat{C}_2\end{pmatrix}$ *is minimal if the faces with Dirichlet boundary conditions are numbered first in the local ordering of each finite element.*

Proof. Because of the block structure of A we can consider the individual finite elements independently. The minimum value of the nonzero count of $\hat{C}_1 \cup \hat{C}_2$ in five subsequent rows which correspond to the same finite element is $\frac{1}{2}\sum_{i\in\mathcal{E}_h}(NIF_i + NNC_i)(NIF_i + NNC_i + 1)$ and it is easily checked to be minimal in this case. ∎

Therefore, from now we assume that within each element we have first numbered the faces corresponding to Dirichlet boundary conditions, then the interior interelement faces and finally the faces with Neumann boundary conditions. The matrix (2.4) can be written in the form

$$\begin{pmatrix}\hat{B} & \hat{C}_1 & \hat{C}_2\end{pmatrix}^T\begin{pmatrix}\hat{B} & \hat{C}_1 & \hat{C}_2\end{pmatrix} = \begin{pmatrix}\hat{B}^T\hat{B} & \hat{B}^T\hat{C}_1 & \hat{B}^T\hat{C}_2 \\ \hat{C}_1^T\hat{B} & \hat{C}_1^T\hat{C}_1 & \hat{C}_1^T\hat{C}_2 \\ \hat{C}_2^T\hat{B} & \hat{C}_2^T\hat{C}_1 & \hat{C}_2^T\hat{C}_2\end{pmatrix}. \tag{2.5}$$

It is clear that it is more advantageous to keep most of the blocks of (2.5) in the explicit form multiplying the factors directly. A typical example is the block $\hat{B}^T\hat{B}$, which is a diagonal matrix. The main question here is whether we can reduce the system further as in the previous case and at the same time keep the matrix blocks in a factorized form. Unfortunately, there is one basic obstacle. Whereas we are able to embed the structure of $A_{12}^T A_{11}^{-1} A_{12}$ into the structure of A_{22} we cannot in general express $\hat{C}_1^T\hat{C}_1 - C_1^T\hat{B}(\hat{B}^T\hat{B})^{-1}\hat{B}^T C_1 = \hat{C}_1^T(I - \hat{B}(\hat{B}^T\hat{B})^{-1}\hat{B}^T)C_1$ in the factorized form as $\hat{C}_1^T L_{B_{11}} L_{B_{11}}^T \hat{C}_1$, where $L_{B_{11}}$ is factor which can be easily computed.

We have considered the partially factorized structure (2.5) since it is important from a computational point of view. Using a structural prediction based on such factors is exactly the way how to obtain the *sparsity structure* of explicit Schur complement matrices $-\mathsf{A}/A$, $(-\mathsf{A}/A)/A_{11}$, and $((-\mathsf{A}/A)/A_{11})/B_{22}$ in an efficient way. In our implementations we used a procedure similar to the one from [16] to get these structures.

3. Direct solution of the Schur complement systems. In the following we will discuss the direct solution of the Schur complement systems. Namely, we will concentrate on the system with the matrix $((-\mathsf{A}/A)/A_{11})/B_{22} \in \mathcal{R}^{NIF,NIF}$. The following theorem gives a bound on the asymptotic work necessary to solve the linear system (1.12), which is dominated by the decomposition of the matrix $((-\mathsf{A}/A)/A_{11})/B_{22}$.

THEOREM 3.1. *The number of arithmetic operations to solve the symmetric indefinite system* (1.12) *directly via three consecutive block eliminations and using the Cholesky decomposition is* $O(NIF^2)$.

Proof. We will only give a sketch of the proof here. The work is dominated by the decomposition of $B_{11} - B_{12}B_{22}^{-1}B_{12}^T$, which has the same nonzero structure as A_{22}.

Our uniform regular finite element mesh is a well-shaped mesh in a suitable sense (see [19]). The proof of Lemma 2.2 and the statements of Theorems 2.1 and 2.2 imply that the graph G of $((-\mathsf{A}/A)/A_{11})/B_{22}$ is also the graph of a well-shaped mesh. Namely, it is a bounded-degree subgraph of some overlap graph (see [18], [19]). It was shown in [25] that the upper bound on the second-smallest eigenvalue of the Laplacian matrix of G (the Fiedler value) is of the order $O(1/NIF^{2/3})$. Then using the techniques from [25] we obtain that there exists a $O(NIF^{2/3})$-size bisector of G.

Therefore, G satisfies the so-called $NIF^{2/3}$-separator theorem: there exist constants $1/2 \leq \alpha\langle 1, \beta\rangle 0$ such that the vertices of G can be partitioned into sets G_A, G_B and the vertex separator G_C such that $|G_A|, |G_B| \leq \alpha NIF$ and $|G_C| \leq \beta NIF^{2/3}$. Moreover, any subgraph of G satisfies the $NIF^{2/3}$-separator theorem. The technique of recursive partitioning of G called generalized nested dissection and used to reorder the considered Schur complement matrix provides an elimination ordering with an $O(NIF^2)$-bound on the arithmetic work of Cholesky decomposition (see Theorem 6 in [12]). □

Note that the explicit computation of the matrix $((-\mathsf{A}/A)/A_{11})/B_{22}$ is necessary in the framework of direct methods. Theorem 3.1 provides a theoretical result which is based on spectral partitioning methods. The reordering algorithms based on the separators obtained by the spectral partitioning techniques and applied recursively within the nested dissection need not necessarily be the best practical approach to get a reasonable matrix reordering. Nevertheless, experimental results with various partitioning schemes show that high quality reorderings can be efficiently computed in this way (see [7]). Also some other reorderings which combine global procedures (partitioning of large meshes) and local algorithms (like MMD) can provide reasonable strategies to find a fill-in minimizing permutation.

4. The conjugate gradient method applied to the Schur complement systems. In this section we concentrate on the iterative solution of the Schur complement systems discussed in section 3. We consider the conjugate gradient method applied to the symmetric positive definite systems with matrices $-\mathsf{A}/A$, $((-\mathsf{A}/A)/A_{11})$, and $((-\mathsf{A}/A)/A_{11})/B_{22}$. It is well known that the convergence rate of the conjugate gradient method can be bounded in terms of the condition number of the corresponding Schur complement matrix [9], [6], [26]. We show that the condition number of the matrix A/A is asymptotically the same as the conditioning of the negative part of spectrum of the whole indefinite matrix A. Moreover, we prove that condition numbers of the matrices $((-\mathsf{A}/A)/A_{11})$ and $((-\mathsf{A}/A)/A_{11})/B_{22}$ grow like $1/h^2$ with respect to the discretization parameter h and they do not deteriorate during the successive eliminations. Based on these results we estimate the number of iteration steps necessary to achieve the prescribed tolerance in error norm reduction. We show that the number of iteration steps necessary to reduce the error norm by the factor of ε

grows asymptotically like $1/h$ for all three Schur complement systems. Therefore, the total number of flops in the iterative algorithm can be significantly reduced due to decrease of the matrix order during the elimination. First, we consider the following theorem.

THEOREM 4.1. *Let $\mu_1 \geq \mu_2 \geq \cdots \geq \mu_{NA} > 0$ be the eigenvalues of the positive definite block $A \in \mathcal{R}^{NA,NA}$, $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_{NBC} > 0$ be the singular values of the matrix block $(B\ C) \in \mathcal{R}^{NA,NBC}$. Then for the eigenvalues of the Schur complement matrix $-\mathsf{A}/A = (B\ C)^T A^{-1}(B\ C)$ we have*

$$\sigma(-\mathsf{A}/A) \subset [\sigma^2_{NBC}/\mu_1, \sigma^2_1/\mu_{NA}]. \tag{4.1}$$

Moreover, for the eigenvalues of the positive definite matrix blocks $\begin{pmatrix} A_22 & A_23 \\ A_{22}^T & A_33 \end{pmatrix} = (C_1\ C_2)^T A^{-1}(C_1\ C_2)$ and $A_{11} = B^T A^{-1} B$ it follows that

$$\sigma \begin{pmatrix} A_{22} & A_{23} \\ A_{23}^T & A_{33} \end{pmatrix} \subset [1/\mu_1, 2/\mu_{NA}], \tag{4.2}$$

$$\sigma(A_{11}) \subset [5/\mu_1, 5/\mu_{NA}]. \tag{4.3}$$

The condition number of the Schur complement system matrix $-\mathsf{A}/A$ then can be bounded by the expression

$$\kappa(-\mathsf{A}/A) \leq \frac{\sigma_1^2 \mu_1}{\sigma^2_{NBC}\mu_{NA}} = \kappa^2((B\ C))\kappa(A). \tag{4.4}$$

Proof. The positive definite matrix A^{-1} has the spectrum $0 < 1/\mu_1 \leq 1/\mu_2 \leq \cdots \leq 1/\mu_{NA}$. The first inclusion in the theorem follows from the following two inequalities

$$\frac{1}{\mu_1}((B\ C)x, (B\ C)x) \leq ((B\ C)^T A^{-1}(B\ C)x, x) \leq \frac{1}{\mu_{NA}}((B\ C)x, (B\ C)x),$$
$$\sigma^2_{NBC}(x,x) \leq ((B\ C)^T (B\ C)x, x) \leq \sigma_1^2(x,x).$$

Similarly, from the inequalities

$$\frac{1}{\mu_1}(Cx, Cx) \leq (C^T A^{-1} Cx, x) \leq \frac{1}{\mu_{NA}}(Cx, Cx),$$
$$(x,x) \leq (C^T Cx, x) \leq 2(x,x)$$

we obtain the second inclusion. The third part of the proof is completely analogous to the second part. ∎

COROLLARY 4.1. *There exist positive constants c_9 and c_{10} such that for the spectrum of the Schur complement matrix $-\mathsf{A}/A$ we have*

$$\sigma(-\mathsf{A}/A) \subset [c_9/NE, c_{10}/\sqrt[3]{NE}], \tag{4.5}$$

where $c_9 = c_3^2/c_2$ and $c_{10} = c_4^2/c_1$. The condition number of the matrix $-\mathsf{A}/A$ can be bounded as

$$\kappa(-\mathsf{A}/A) \leq c_{11}\sqrt[3]{NE^2}, \quad c_{11} = c_{10}/c_9. \tag{4.6}$$

The Schur complement system with positive definite matrix $-\mathsf{A}/A$ can be solved iteratively by the conjugate gradient method [9] or the conjugate residual method [6]. It is well known that the conjugate gradient method generates the approximate solutions which minimize the energy norm of the error at each iteration step [26], [6]. The closely related conjugate residual method that differ only in the definition of innerproduct, on the other hand, generates the approximate solutions which minimize their residual norm at every iteration [6]. It is also a well known fact that there exists so-called peak/plateau connection between these methods [5] showing that there is no significant difference in the convergence rate of these methods when measured by the residual norm of an approximate solution. In our paper we use the conjugate gradient method together with the minimal residual smoothing procedure applied on its top to get monotonic residual norms [28]. Applying such techniques allows better monitoring of the convergence by residual norm and it is mathematically equivalent to the residual minimizing conjugate residual method [6]. The computational cost of this technique is minimal and it costs only two inner products and one vector update per iteration. In the framework of iterative methods the number of operations in matrix-vector products is what is usually the most important. These products, performed repeatedly in each iteration loop, contribute in a substantial way to the final efficiency of iterative solver. When solving the system with Schur complement matrix $-\mathsf{A}/A$ the number of flops per iteration for an unpreconditioned method is dominated by the matrix vector multiplication with the matrix $-\mathsf{A}/A$. Its number of nonzeros was given by Lemma 2.2. Moreover, using the estimates (1.13) and (1.14), the condition number of the Schur complement matrix $-\mathsf{A}/A$ can be bounded by the term $O(\sqrt[3]{NE^2})$. Consequently, the number of flops for conjugate gradients necessary to achieve a reduction by ε is of order

$$O\left(\left[NE + NIF + NNC + \sum_{i\in\mathcal{E}_h}(NIF_i + NNC_i)^2 + \sum_{i\in\mathcal{E}_h}\sum_{j\in Adj(i)} NIF_j\right]\sqrt[3]{NE}\right).$$

Assuming overestimates $(NIF_i + NNC_i) \le 5$ and $NIF_i \le 5$, we obtain the asymptotic estimate of order $O(NE\sqrt[3]{NE})$.

The previous considerations did not take into account the Schur complement systems with matrices $((-\mathsf{A}/A)/A_{11})$ and $(((-\mathsf{A}/A)/A_{11})/B_{22})$. The convergence rate of the iterative conjugate gradient method applied to the second and third Schur complement systems depend analogously on the condition number of the Schur complement matrices [9], [6], [26]. The analysis of the spectrum of the matrix $((-\mathsf{A}/A)/A_{11})$ is given in the following theorem.

THEOREM 4.2. *Let $\mu_1 \ge \mu_2 \ge \cdots \ge \mu_{NA} > 0$ be the eigenvalues of the positive definite block $A \in \mathcal{R}^{NA,NA}$, $\sigma_1 \ge \sigma_2 \ge \cdots \ge \sigma_{NBC} > 0$ be the singular values of the matrix block $(B\ C) \in \mathcal{R}^{NA,NBC}$. Then for the spectrum of the Schur complement matrix $(-\mathsf{A}/A)/A_{11}$ we have*

$$\sigma((-\mathsf{A}/A)/A_{11}) \subset \left[\sigma^2_{NBC}/\mu_1, 2/\mu_{NA}\right]. \tag{4.7}$$

Consequently, the condition number of the matrix $(-\mathsf{A}/A)/A_{11}$ can be bounded as follows

$$\kappa((-\mathsf{A}/A)/A_{11}) \le \frac{2}{\sigma^2_1}\kappa(-\mathsf{A}/A). \tag{4.8}$$

Proof. From the definition of the Schur complement matrix $(-\mathsf{A}/A)/A_{11}$ and the statement of Theorem 4.1 we have

$$\begin{aligned}((-\mathsf{A}/A)/A_{11}x,x) &= \left(\begin{pmatrix} A_{22} & A_{23} \\ A_{23}^T & A_{33}\end{pmatrix} x, x\right) - (A_{11}^{-1}\,(A_{12}\ A_{13})\,x, (A_{12}\ A_{13})\,x) \\ &\le 2/\mu_{NA}(x,x).\end{aligned}$$

The bound for the minimal eigenvalue can be obtained considering the following result (see [20, p. 201]):

$$(-\mathsf{A}/A)^{-1} = \begin{pmatrix} A_{11} & A_{12} & A_{13} \\ A_{12}^T & A_{22} & A_{23} \\ A_{13}^T & A_{23}^T & A_{33}\end{pmatrix}^{-1} =$$

$$\begin{pmatrix} A_{11}^{-1} + A_{11}^{-1}(A_{12}A_{13})[(-\mathsf{A}/A)/A_{11}]^{-1}\begin{pmatrix} A_{12}^T \\ A_{13}^T\end{pmatrix}A_{11}^{-1} & -A_{11}^{-1}(A_{12}\ A_{13})[(-\mathsf{A}/A)/A_{11}]^{-1} \\ -[(-\mathsf{A}/A)/A_{11}]^{-1}\begin{pmatrix} A_{12}^T \\ A_{13}^T\end{pmatrix}A_{11}^{-1} & [(-\mathsf{A}/A)/A_{11}]^{-1}\end{pmatrix}.$$

Then from the interlacing property of the eigenvalue set of symmetric matrix $-\mathsf{A}/A$ (see, e.g., [8]) it follows that

$$\left\|[(-\mathsf{A}/A)/A_{11}]^{-1}\right\| \le \left\|(-\mathsf{A}/A)^{-1}\right\| \le \frac{\mu_1}{\sigma_{NBC}^2}.$$

Considering the previous inequalities we get the lower bound for the minimal eigenvalue of the matrix $(-\mathsf{A}/A)/A_{11}$, which completes the proof. □

We have shown that the condition number of the Schur complement system matrix $(-\mathsf{A}/A)/A_{11}$ is bounded by a multiple of the condition number of the matrix $-\mathsf{A}/A$. Therefore the number of iteration steps for the conjugate gradient method necessary to reduce the error norm (or after smoothing the residual norm) by some factor is asymptotically the same as before. The complexity of the matrix-vector multiplication is lower and according to Corollary 2.1 is of the order

$$O\left(\sum_{i\in\mathcal{E}_h}(NIF_i + NNC_i)^2 + \sum_{i\in\mathcal{E}_h}\sum_{j\in Adj(i)} NIF_j + (NNC - NIF)\right).$$

Assuming again the overestimates $(NIF_i + NNC_i) \le 5$ and $NIF_i \le 5$, we obtain the asymptotic estimate $O(NE)$. The total number of flops for the conjugate gradients or the conjugate residual method necessary to achieve a reduction by the factor ε is then again of order $O(NE\sqrt[3]{NE})$. From the statements of Theorems 4.1 and 4.2 it is clear that the reduction to the Schur complement systems does not affect the asymptotic conditioning of the positive definite matrices $-\mathsf{A}/A$ and $(-\mathsf{A}/A)/A_{11}$. The same is true for the spectral properties of the third Schur complement system with the matrix $((-\mathsf{A}/A)/A_{11})/B_{22}$. Since the proof is completely analogous to the proof of Theorem 4.2 we shall present only the following statement (cf. [10, p. 256]).

THEOREM 4.3. *The condition number of* $((-\mathsf{A}/A)/A_{11})/B_{22}$ *is bounded by the condition number of the matrix* $(-\mathsf{A}/A)/A_{11}$

$$\kappa(((-\mathsf{A}/A)/A_{11})/B_{22}) \le \kappa((-\mathsf{A}/A)/A_{11}). \tag{4.9}$$

In the following we present two additional results concerning the matrix-vector multiplications with Schur complement matrices. Theorem 4.4 compares the number

of nonzeros in the Schur complement matrices $(-\mathsf{A}/A)/A_{11}$ and $((-\mathsf{A}/A)/A_{11})/B_{22}$ to the number of nonzeros in the original matrix A.

THEOREM 4.4. *The number of nonzero entries in the matrix* $((-\mathsf{A}/A)/A_{11})$ *or the matrix* $(((-\mathsf{A}/A)/A_{11})/B_{22})$ *is smaller than the number of nonzeros in the matrix* A.

Proof. Using the fact that $2NIF = \sum_{i\in\mathcal{E}_h} NIF_i \le 5NE$ and also $\sum_{i\in\mathcal{E}_h}(NIF_i + NNC_i) \le 5NE$, it follows from Lemma 2.1 and Theorem 2.1 that

$$
\begin{aligned}
&|(-\mathsf{A}/A)/A_{11}| - |\mathsf{A}| \\
&= \sum_{i\in\mathcal{E}_h} NIF_i^2 - 2NIF + \sum_{i\in\mathcal{E}_h}\sum_{j\in Adj(i)} NIF_j + \sum_{i\in\mathcal{E}_h} NNC_i^2 \\
&+2\sum_{i\in\mathcal{E}_h} NIF_i NNC_i - 35NE - 4NIF - 2NNC \\
&= \sum_{i\in\mathcal{E}_h}(NIF_i + NNC_i)^2 - 5\sum_{i\in\mathcal{E}_h} NIF_i - 4\sum_{i\in\mathcal{E}_h} NNC_i - \sum_{i\in\mathcal{E}_h}\sum_{j\in Adj(i)} NNC_j \\
&+2\sum_{i\in\mathcal{E}_h}(NIF_i + NNC_i) + \sum_{i\in\mathcal{E}_h}\sum_{j\in Adj(i)}(NIF_j + NNC_j) - 35NE \\
&\le 2\sum_{i\in\mathcal{E}_h}(NIF_i + NNC_i) + \sum_{i\in\mathcal{E}_h}\sum_{j\in Adj(i)}(NIF_j + NNC_j) - 35NE \le 0. \qquad \square
\end{aligned}
$$

Clearly, the number of nonzeros in the matrix $(((-\mathsf{A}/A)/A_{11})/B_{22})$ is even smaller.

Note that the number of nonzeros in the original matrix A can be smaller or larger than the corresponding number of nonzeros in the matrix $-\mathsf{A}/A$. Consider now the factorized Schur complement in the form (2.5). It can be shown also that there is no clear winner between the number of floating-point operations to multiply a dense vector by the matrix $\begin{pmatrix}\hat{B} & \hat{C}_1 & \hat{C}_2\end{pmatrix}^T \begin{pmatrix}\hat{B} & \hat{C}_1 & \hat{C}_2\end{pmatrix}$ or the number of operations to get a product of a matrix $((-\mathsf{A}/A)/A_{11})/B_{22}$ with a dense vector of appropriate dimension, respectively. The result depends on the shape of the domain and its boundary conditions. Nevertheless, the following Theorem 4.5 shows that if we do not form the Schur complement explicitly it is worth it to use the factorized form (2.4) and the reordering of the Schur complement from Theorem 2.3 instead of its implicit form.

THEOREM 4.5. *Let* v *be a dense vector. The number of floating-point operations to compute* $\begin{pmatrix}\hat{B} & \hat{C}_1 & \hat{C}_2\end{pmatrix}^T \begin{pmatrix}\hat{B} & \hat{C}_1 & \hat{C}_2\end{pmatrix} v$ *is smaller than the number of floating-point operations to compute*

$$
\begin{pmatrix} B^T \\ C_1^T \\ C_2^T \end{pmatrix} A^{-1} \begin{pmatrix} B & C_1 & C_2 \end{pmatrix} v.
$$

Proof. Taking into account the local ordering from Theorem 2.3 the difference between these two quantities can be bounded by

$$
\begin{aligned}
&10NE + \sum_{i\in\mathcal{E}_h}(NIF_i + NNC_i)(NIF_i + NNC_i + 1) - 35NE - 4NIF - 2NNC \\
&\le -2NIF - NNC \le 0. \qquad \square
\end{aligned}
$$

5. Numerical experiments. In the following we present numerical experiments which illustrate the results developed in the theoretical part of the paper.

TABLE 1
Model potential fluid flow problem—cubic domain.

Discretization parameters			Matrix dimensions			
h, NE	NIF	NNC	A	$-\mathsf{A}/A$	$(-\mathsf{A}/A)/A_{11}$	$((-\mathsf{A}/A)/A_{11})/B_{22}$
1/5, 250	525	100	2125	875	625	525
1/10, 2000	4600	400	17000	7000	5000	4600
1/15, 6750	15975	900	57375	23625	16875	15975
1/20, 16000	38400	1600	136000	56000	40000	38400
1/25, 31250	75625	2500	265625	109375	78125	75625
1/30, 54000	131400	3600	459000	189000	135000	131400
1/35, 87750	209475	4900	728875	300125	214375	209475
1/40, 128000	313600	6400	1088000	448000	320000	313600

TABLE 2
Model potential fluid flow problem—realistic domain.

Discretization parameter			Matrix dimension			
NE	NIF	NNC	A	$-\mathsf{A}/A$	$(-\mathsf{A}/A)/A_{11}$	$((-\mathsf{A}/A)/A_{11})/B_{22}$
35x35x6	33880	4900	126980	53480	38780	33880
45x45x6	56160	8100	210060	88560	64260	56160
55x55x6	84040	12100	313940	132440	96140	84040
65x65x6	117520	16900	438620	185120	134420	117520
75x75x6	156600	22500	584100	246600	179100	156600
85x85x6	201280	28900	750380	316880	230180	201280
95x95x6	251560	36100	937460	395960	287660	251560
105x105x6	307440	44100	1145340	483840	351540	307440

Two model potential flow problems (1.1) and (1.2) in a rectangular domain with Neumann conditions prescribed on the bottom and on the top of the domain have been considered. Dirichlet conditions that preserve the nonsingularity of the whole system matrix A were imposed on the rest of the boundary. The choice of boundary conditions in these examples is motivated by our application and it comes from a modelling of a confined aquifer (see [3]) between two impermeable layers.

In order to verify the theoretical results derived in previous sections we will restrict our attention first to the simplest geometrical shape—cubic domain and report the results obtained from a uniformly regular mesh refinement. In practical situations, however, relatively thin aquifers with possible cracks in the rock are frequently modeled, and so the number of Neumann conditions may represent a big portion of the whole boundary. As our second model example, we consider a rectangular domain discretized by six layers of elements in the mesh. As we will see later, the reduction to the third Schur complement proposed in this paper can become even more significant than for the cubic domain. Prismatic discretizations of domains with NE elements were used [14], [11]. For the cubic domain we have then $NE = 2/h^3$. Discretization parameters h, NE, NIF, NNC, dimension N of the resulting indefinite system matrix A and the dimensions of the corresponding Schur complement matrices $-\mathsf{A}/A$, $(-\mathsf{A}/A)/A_{11}$, and $((-\mathsf{A}/A)/A_{11})/B_{22}$ are given in Table 1 for a cubic domain and in Table 2 for a more realistic domain. We note again that the difference between dimensions of the second and third Schur complement matrix is significantly larger in the case of modeling of thin layers that arise regularly in our application.

For the example of a cubic domain the spectral properties of the matrix blocks A and $(B\ C)$ as well as of the whole symmetric indefinite matrix A have been investigated. The extremal positive and negative eigenvalues of the matrix A and the extremal singular values of the block $(B\ C)$ (squared roots of the extremal eigenval-

TABLE 3
Spectral properties of the system matrix and its blocks—problem with a cubic domain

	Matrix blocks spectral properties		Eigenvalues of the matrix A	
NE	Spectrum of A	Sing. values of $(B\|C)$	Negative part	Positive part
250	[0.16e-2, 0.1e-1]	[0.181e0, 2.63]	[-2.63 , -0.180e0]	[0.166e-2, 2.63]
2000	[0.33e-2, 0.2e-1]	[0.927e-1, 2.64]	[-2.64, -0.898e-1]	[0.335e-2, 2.64]
6750	[0.50e-2, 0.3e-1]	[0.622e-1, 2.64]	[-2.64, -0.354e-1]	[0.509e-2, 2.65]
16000	[0.66e-2, 0.4e-1]	[0.467e-1, 2.64]	[-2.64, -0.413e-1]	[0.679e-2, 2.65]
31250	[0.83e-2, 0.5e-1]	[0.374e-1, 2.65]	[-2.64, -0.311e-1]	[0.861e-2, 2.65]
54000	[0.99e-2, 0.6e-1]	[0.312e-1, 2.65]	[-2.64, -0.241e-1]	[0.104e-1, 2.65]
87750	[0.11e-1, 0.7e-1]	[0.268e-1, 2.65]	[-2.64, -0.190e-1]	[0.120e-1, 2.65]
128000	[0.13e-1, 0.8e-1]	[0.234e-1, 2.65]	[-2.64, -0.152e-1]	[0.136e-1, 2.65]

ues of the matrix $(B\ C)^T(B\ C)$) were approximated by a reduction to the symmetric tridiagonal form of the matrix using 1500 steps of the symmetric Lanczos algorithm [8] and by a subsequent eigenvalue computation of the resulting tridiagonal matrix using the LAPACK double precision subroutine DSYEV [1]. Extremal eigenvalues of the diagonal matrix block A were computed directly by the LAPACK eigenvalue solver element by element. It can be seen that the computed extremal eigenvalues of the block A are in perfect agreement with the theory (see Table 3). Similarly, we can observe approximately a linear decrease of the computed minimal singular value of the matrix block $(B\ C)$ with respect to the mesh discretization parameter h. From the computed extremal eigenvalues of the whole indefinite system A we can conclude that even if our mesh size parameters h are rather small and give rise to very large system dimensions (see Table 1), they are outside of the asymptotic inclusion set (1.15). Indeed for our example and our mesh size interval we have $c_1/h \ll c_4$, $c_2/h \ll c_4$, and with the exception of $h = 1/35$ and $h = 1/40$ also $c_2/h < c_3 h$. Then using Lemma 2.1 in [22, pp. 3–4] (see also [15]) we obtain the inclusion set in the form

$$\sigma(\mathsf{A}) \subset \left[-c_4, -\frac{1}{2}\left(c_2\sqrt[3]{NE} - \sqrt{(c_2\sqrt[3]{NE})^2 + 4(c_3/\sqrt[3]{NE})^2}\right)\right] \cup [c_1\sqrt[3]{NE}, c_4], \tag{5.1}$$

which is in good agreement with the results in Table 3.

Using the same technique we have approximated the extremal eigenvalues of the Schur complement matrices $-\mathsf{A}/A$, $(-\mathsf{A}/A)/A_{11}$, and $((-\mathsf{A}/A)/A_{11})/B_{22}$ coming from a problem on a cubic domain. From Table 4 it can be seen that the inclusion set for the extremal eigenvalues of the first Schur complement matrix $-\mathsf{A}/A$ coincides with the bounds given in Theorem 4.1. We can see that the extremal eigenvalues of the second Schur complement matrix $(-\mathsf{A}/A)/A_{11}$ are bounded by the extremal eigenvalues of the matrix $-\mathsf{A}/A$. Similarly, the extremal eigenvalues of the third Schur complement matrix $((-\mathsf{A}/A)/A_{11})/B_{22}$ are bounded by the extremal eigenvalues of the matrix $(-\mathsf{A}/A)/A_{11}$. This behavior is in accordance with the asymptotic bounds given in Theorems 4.2 and 4.3.

The smoothed conjugate gradient method has been applied to the resulting three Schur complement systems (see also the discussion in previous section). Unpreconditioned and also preconditioned versions with the IC(0) preconditioner [23], [24] for the solution of these symmetric positive definite systems have been used. For the solution of the whole indefinite system the minimal residual method has been used. For the preconditioned version the positive definite block-diagonal preconditioning with ILUT(0,20) for the decomposition of the block corresponding to constraints (see, e.g., [22], [21]) was used. The choice of ILUT(0,20) was motivated by our effort to ob-

TABLE 4
Spectral properties of Schur complement matrices—problem with a cubic domain.

	Spectral properties of Schur complement matrices		
NE	$-\mathsf{A}/A$	$(-\mathsf{A}/A)/A_{11}$	$((-\mathsf{A}/A)/A_{11})/B_{22}$
250	[0.138e2, 0.343e4]	[0.187e2, 0.117e4]	[0.220e2, 0.117e4]
2000	[0.182e1, 0.173e4]	[0.251e1, 0.596e3]	[0.272e1, 0.596e3]
6750	[0.547e0, 0.115e4]	[0.760e0, 0.399e3]	[0.801e0, 0.399e3]
16000	[0.232e0, 0.868e3]	[0.323e0, 0.299e3]	[0.336e0, 0.299e3]
31250	[0.119e0, 0.694e3]	[0.166e0, 0.239e3]	[0.171e0, 0.239e3]
54000	[0.693e-1, 0.579e3]	[0.966e-1, 0.199e3]	[0.992e-1, 0.199e3]
87750	[0.437e-1, 0.496e3]	[0.610e-1, 0.171e3]	[0.624e-1, 0.171e3]
128000	[0.293e-1, 0.434e3]	[0.409e-1, 0.149e3]	[0.417e-1, 0.149e3]

tain rather precise factorization with restricted memory requirements which should be close to the full decomposition of the block $(B\ C)^T(B\ C)$. This preconditioner was found generally better than the indefinite block-diagonal preconditioning with the same ILUT(0,20) decomposition or than the indefinite preconditioner discussed in [13] or [21]. The initial approximation x_0 was set to zero, the relative residual norm $\frac{\|r_n\|}{\|r_0\|} = 10^{-8}$ was used as the stopping criterion. For the implementation details of iterative solvers we refer to [6]. Our experiments were performed on an SGI Origin 200 with processor R10000. In Tables 5 and 6 we consider iteration counts and CPU times in the minimal residual method (unpreconditioned/preconditioned) applied to the whole system (1.12) and in the conjugate gradient method (unpreconditioned/preconditioned) applied to the Schur complement systems with the matrices $-\mathsf{A}/A$, $(-\mathsf{A}/A)/A_{11}$, and $((-\mathsf{A}/A)/A_{11})/B_{22}$ for a model problem with a cubic and more realistic domain, respectively. The dependence of the iteration counts presented in all columns of Table 5 corresponds surprisingly well to the theoretical order $O(\sqrt[3]{NE})$. The convergence behavior of the smoothed conjugate gradient method applied to the third Schur complement system with the matrix $((-\mathsf{A}/A)/A_{11})/B_{22}$ for this case is presented in Figure 4. From the results in Tables 5 and 6 it follows that while the gain from the solution of the third Schur complement system is rather moderate in the case of a cubic domain and in the case of the realistic flat domain it becomes more significant.

6. Conclusions. Successive block Schur complement reduction for the solution of symmetric indefinite systems has been considered in the paper. It was shown that due to the particular structure of matrices which arise from mixed-hybrid finite element discretization of the potential fluid flow problem, the resulting Schur complement matrices remain sparse. Moreover, their spectral properties do not deteriorate and the iterative conjugate gradient method can be successfully applied. Theoretical bounds for the convergence rate of this method in terms of the discretization parameters have been developed and tested on a model problem example. Numerical experiments indicate that the given theoretical bounds on the eigenvalue set are realistic not only for the system matrix and its blocks, but also for the Schur complement matrices. The iteration counts for the conjugate gradient method are also in a good agreement with the theoretical predictions. Direct solution of the third Schur complement system is also a possible alternative. Nevertheless, its comparison with iterative solvers is outside the scope of this paper.

In case of structured grids, a geometric multigrid solver and/or preconditioner for solving the final Schur complement system can be used. Namely, the stencil from the first Schur complement which expresses element-element connectivity in the domain

TABLE 5
Number of iterations of the conjugate gradient method—problem with a cubic domain.

	Unpreconditioned/preconditioned CG applied to matrix			
NE	A	$-\mathsf{A}/A$	$(-\mathsf{A}/A)/A_{11}$	$((-\mathsf{A}/A)/A_{11})/B_{22}$
250	319/44 0.41/0.09	82/20 0.05/0.02	48/18 0.02/0.01	43/18 0.02/0.01
2000	608/76 6.43/1.56	154/35 0.76/0.25	87/32 0.30/0.16	80/32 0.25/0.14
6750	867/113 48.17/11.91	223/51 3.86/1.50	126/48 1.51/0.92	118/48 1.30/1.01
16000	1031/138 161.75/38.86	288/67 16.07/5.93	164/63 5.59/3.87	155/63 5.02/3.43
31250	1195/165 410.45/100.49	353/82 45.69/16.75	199/78 16.31/9.94	192/78 15.13/9.60
54000	1358/188 926.98/218.78	418/95 104.76/36.92	234/93 39.88/24.04	228/93 37.94/23.13
85750	1503/205 1694.68/396.03	482/108 216.10/74.42	269/108 76.21/47.55	263/108 72.00/45.71
128000	1637/229 2825.94/675.49	546/122 389.37/133.10	303/122 143.28/87.63	298/122 138.72/87.28

TABLE 6
Number of iterations of the conjugate gradient method—realistic model example.

	Unpreconditioned/preconditioned CG applied to matrix			
NE	A	$-\mathsf{A}/A$	$(-\mathsf{A}/A)/A_{11}$	$((-\mathsf{A}/A)/A_{11})/B_{22}$
14700	1688/209 231.51/54.57	444/97 22.96/8.12	248/93 7.52/5.27	233/93 6.89/4.35
24300	2043/265 510.65/123.32	571/122 54.46/18.48	316/117 20.51/11.69	296/117 16.44/10.30
36300	2225/300 919.43/224.74	692/147 118.83/37.24	382/142 38.93/22.75	359/142 31.16/19.57
50700	2053/336 1547.14/370.18	810/172 195.08/62.90	448/166 73.65/40.68	421/166 57.01/34.15
67500	2723/365 2343.57/559.24	927/197 316.33/103.93	513/190 116.30/65.70	482/190 93.15/56.55
86700	2959/403 3374.38/814.61	1042/222 495.97/160.23	578/214 175.55/99.90	543/214 138.81/83.85
108300	3211/429 4621.81/1086.38	1256/247 821.96/240.74	741/254 306.28/158.08	741/255 253.10/132.33
132300	3420/447 6012.54/1382.56	1272/271 1009.94/323.72	706/262 374.90/208.07	663/262 299.87/177.34

(see proof of Lemma 2.2) remains unchanged after the subsequent two reduction and an appropriate method could be based on that.

Another approach for the solution of symmetric indefinite systems seems to be promising. As was pointed out in [2], the classical null-space algorithm can be implemented. QR factorization of the off-diagonal block $(B\ C)$ is considered and the solution of the indefinite system is transformed to the solution of a block lower triangular system, where the subproblem corresponding to the diagonal block can be solved using the Cholesky factorization or an iterative conjugate gradient-type algorithm. This approach has the advantage of performing the matrix-vector multiplication by the Q factor using elementary Householder transformations. Although the Q factor may be structurally full, the elementary Householder vectors may be quite sparse. Moreover, a roundoff error analysis of the algorithm can be carried out.

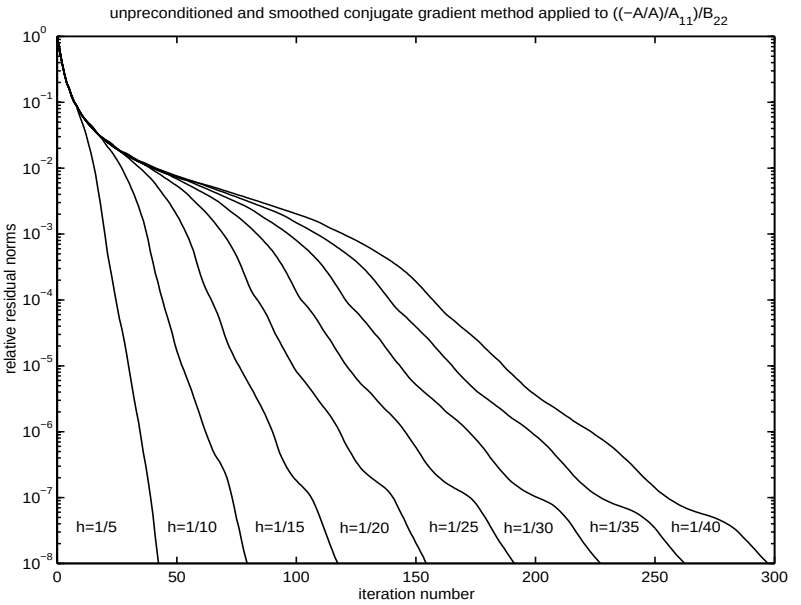


FIG. 4. *Convergence of the smoothed conjugate gradient method applied to the third Schur complement system.*

Acknowledgment. Authors would like to thank Michele Benzi for careful reading of manuscript and anonymous referees for their many useful comments which significantly improved the presentation of the paper. We are indebted to Jiří Mužák from the Department of Mathematical Modeling in DIAMO, s.e., Stráž pod Ralskem for providing us with a model numerical example for the experimental part of this paper and to Jörg Liesen for giving us the reference [20].

REFERENCES

[1] E. ANDERSON, Z. BAI, C. BISCHOF, J. DEMMEL, J. DONGARRA, J. DU CROZ, A. GREENBAUM, S. HAMMARLING, A. MCKENNEY, S. OSTROUCHOV, AND D. SORENSEN, *LAPACK User's Guide*, SIAM, Philadelphia, 1992.

[2] M. ARIOLI, *The use of QR Factorization in Sparse Quadratic Programming*, Technical Report 1070, Istituto di Analisi Numerica del C.N.R., Pavia, Italy, 1998.

[3] J. BEAR, *Dynamics of Fluids in Porous Media*, Elsevier, New York, 1972.

[4] F. BREZZI AND M. FORTIN, *Mixed and Hybrid Finite Element Methods*, Springer Ser. Comput. Math., Springer-Verlag, New York, 1991.

[5] J. CULLUM AND A. GREENBAUM, *Relations between Galerkin and norm-minimizing iterative methods for solving linear systems*, SIAM J. Matrix Anal. Appl., 17 (1996), pp. 223–247.

[6] H.C. ELMAN, *Iterative Methods for Large, Sparse, Nonsymmetric Systems of Linear Equations*, Research Report 229, Yale University, New Haven, CT, 1982.

[7] J.R. GILBERT, G.L. MILLER, AND S.H. TENG, *Geometric mesh partitioning: Implementation and experiments*, in Proceedings 9th International Parallel Processing Symposium, Santa Barbara, CA, IEEE Computer Society Press, Los Alamitos, CA, 1995, pp. 418–427.

[8] G.H. GOLUB AND C.F. VAN LOAN, *Matrix Computations*, 2nd ed., The Johns Hopkins University Press, Baltimore, 1989.

[9] M.R. HESTENES AND E. STIEFEL, *Method of conjugate gradients for solving linear systems*, J. Research Nat. Bur. Standards, 49 (1952), pp. 409–435.

[10] N.J. HIGHAM, *Accuracy and Stability of Numerical Algorithms*, SIAM, Philadelphia, 1996.

[11] E.F. Kaasschieter and A.J.M. Huijben, *Mixed-hybrid finite elements and streamline computation for the potential flow problem*, Numer. Methods Partial Differential Equations, 8 (1992), pp. 221–266.

[12] R.J. Lipton, D.J. Rose, and R.J. Tarjan, *Generalized nested dissection*, SIAM J. Numer. Anal., 16 (1979), pp. 346–358.

[13] L. Lukšan and J. Vlček, *Indefinitely preconditioned inexact Newton method for large sparse equality constrained non-linear programming problems*, Numer. Linear Algebra Appl., 5 (1998), pp. 219–247.

[14] J. Maryška, M. Rozložník, and M. Tůma, *Mixed-hybrid finite element approximation of the potential fluid flow problem*, J. Comput. Appl. Math., 63 (1995), pp. 383–392.

[15] J. Maryška, M. Rozložník, and M. Tůma, *The potential fluid flow problem and the convergence rate of the minimal residual method*, Numer. Linear Algebra Appl., 3 (1996), pp. 525–542.

[16] P. Matstoms, *Sparse QR Factorization with Applications to Linear Least Squares Problems*, Dissertations 337, Linköping Studies in Science and Technology, Department of Mathematics, Linköping University, Linköping, 1994.

[17] J.D. Moulton, J.E. Morel, and U.M. Ascher, *Approximate Schur complement preconditioning of the lowest-order nodal discretizations*, SIAM J. Sci. Comput., 19 (1998), pp. 185–205.

[18] G.L. Miller, S.H. Teng, W. Thurston, and S.A. Vavasis, *Automatic mesh partitioning*, in Graph Theory and Sparse Matrix Computation, A. George, J. Gilbert, and J. Liu, eds., IMA Vol. Math. Appl. 56, Springer-Verlag, New York, 1993, pp. 57–84.

[19] G.L. Miller, S.H. Teng, W. Thurston, and S.A. Vavasis, *Geometric separators for finite element meshes*, SIAM J. Sci. Comput., 19 (1998), pp. 364–396.

[20] D.V. Ouellette, *Schur complement and statistics*, Linear Algebra Appl., 36 (1981), pp. 187–295.

[21] I. Perugia and V. Simoncini, *Block-diagonal and indefinite symmetric preconditioners for mixed finite element formulations*, Technical report 1159, IAN CNR, Pavia, Italy, Numer. Linear Algebra Appl., to appear.

[22] T. Rusten and R. Winther, *A preconditioned iterative method for saddle point problems*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 887–905.

[23] Y. Saad, *Iterative Methods for Sparse Linear Systems*, PWS Publishing Company, Boston, 1996.

[24] Y. Saad, *ILUT: A dual threshold incomplete ILU factorization*, Numer. Linear Algebra Appl., 1 (1994), pp. 387–402.

[25] D.A. Spielman and S.H. Teng, *Spectral Partitioning Works: Planar Graphs and Finite Element Meshes*, Technical Report UCB//CSD-96-898, University of Berkeley, Berkeley, 1996.

[26] H. van der Vorst, *Parallel Iterative Solution Methods for Linear Systems Arising from Discretized PDE's*, Workshop Lecture Notes, in Special Course on Parallel Computing in CFD, AGARD-R-807, AGARD, Neuilly-sur-Seine, France, 1995; also available online at http://www.math.ruu.nl.people/vorst/agard.ps.gz.

[27] C. Wagner, W. Kinzelbach, and G. Wittum, *Schur-complement multigrid: A robust method for groundwater flow and transport problems*, Numer. Math., 75 (1997), pp. 523–545.

[28] R. Weiss, *Parameter-Free Iterative Linear Solvers*, Mathematical Res. Ser. 97, Akademie Verlag, Berlin, 1996.

SIAM J. SCI. COMPUT.
Vol. 22, No. 2, pp. 724–746
© 2000 Society for Industrial and Applied Mathematics

A MULTIRESOLUTION TENSOR SPLINE METHOD FOR FITTING FUNCTIONS ON THE SPHERE*

TOM LYCHE† AND LARRY L. SCHUMAKER‡

Abstract. We present the details of a multiresolution method we proposed at the Taormina Wavelet Conference in 1993 (see "L-spline wavelets" in *Wavelets: Theory, Algorithms, and Applications*, C. Chui, L. Montefusco, and L. Puccio, eds., Academic Press, New York, pp. 197–212) which is suitable for fitting functions or data on the sphere. The method is based on tensor products of polynomial splines and trigonometric splines and can be adapted to produce surfaces that are either tangent plane continuous or almost tangent plane continuous. The result is a convenient compression algorithm for dealing with large amounts of data on the sphere. We give full details of a computer implementation that is highly efficient with respect to both storage and computational cost. We also demonstrate the performance of the method on several test examples.

Key words. multiresolution, spherical data compression, tensor splines

AMS subject classifications. 41A15, 42A10, 41A63, 65D07, 65D17, 65T60

PII. S1064827598344388

1. Introduction.

In many applications (e.g., in geophysics, meteorology, medical modeling, etc.), one needs to construct smooth functions defined on the unit sphere S which approximate or interpolate data. As shown in [13], one way to do this is to work with tensor-product functions of the form

$$f(\theta,\phi) := \sum_{i=1}^{m}\sum_{j=1}^{\widetilde{m}} c_{ij}\varphi_i(\theta)\tilde{\varphi}_j(\phi) \tag{1.1}$$

defined on the rectangle

$$H := \{(\theta,\phi):\ -\pi/2 \le \theta \le \pi/2 \ \text{ and } \ 0 \le \phi \le 2\pi\},$$

where the φ_i are quadratic polynomial B-splines on $[-\pi/2,\pi/2]$, and the $\tilde{\varphi}_j$ are periodic trigonometric splines of order 3 on $[0,2\pi]$. With some care in the choice of the coefficients (see section 2), the associated *surface*

$$\mathcal{S}_f := \{f(\theta,\phi)\boldsymbol{v}(\theta,\phi):\ (\theta,\phi)\in H\}$$

with $\boldsymbol{v}(\theta,\phi) = [\cos(\theta)\cos(\phi), \cos(\theta)\sin(\phi), \sin(\theta)]^T$ will be tangent plane continuous.

In practice we often encounter very large data sets, and to get good fits using tensor product splines (1.1), a large number of knots is required, resulting in many basis functions and many coefficients. Since two spline spaces are nested if their

*Received by the editors September 10, 1998; accepted for publication (in revised form) January 31, 2000; published electronically August 24, 2000.
http://www.siam.org/journals/sisc/22-2/34438.html

†Institutt for Informatikk, University of Oslo, P.O. Box 1080, Blindern 0316 Oslo, Norway (tom@ifi.uio.no). The research of this author was supported by NATO grant CRG951291. Part of the work was completed during a stay at Institut National des Sciences Appliquées and Laboratoire Approximation et Optimisation of Université Paul Sabatier, Toulouse, France.

‡Department of Mathematics, Vanderbilt University, Nashville, TN 37240 (s@mars.cas.vanderbilt.edu). The research of this author was supported by the National Science Foundation under grant DMS-9803340, by NATO grant CRG951291, and by the Army Research Office under grant DAAD 19-99-1-0160.

knot sequences are nested, one way to achieve a more efficient fit without sacrificing quality is to look for a multiresolution representation of (1.1), i.e., to recursively decompose it into splines on coarser meshes and corresponding correction (wavelet) terms. Then compression can be achieved in the standard way by thresholding out small coefficients; see [1], for example.

The paper is organized as follows. In section 2 we introduce notation and give details on the tensor product splines to be used here. In section 3 we describe the general decomposition and reconstruction algorithm in matrix form, while in section 4 we present a tensor version of the algorithms. The required matrices corresponding to the polynomial and trigonometric spline spaces, respectively, are derived in sections 5 and 6. Section 7 is devoted to details of implementing the algorithm, while in section 8 we discuss the case of almost tangent plane continuity. In section 9 we present test examples, and in section 10, several concluding remarks.

The method of this paper was presented at the Taormina Wavelet Conference in October 1993 but was not described in detail in the corresponding proceedings paper [8]. A very similar method based on exponential splines was developed independently in [2], [15], but with no implementation details. Alternative wavelet methods for the sphere can be found in [4], [5], [6], [10], [11]. The method in [4] uses discretizations of certain continuous wavelet transforms based on singular integral operators, while the method in [10] uses tensor functions based on polynomials and trigonometric polynomials. The methods in [6], [11] are based on triangulations. For a complete review of the various methods, see [5].

2. Tangent plane continuous tensor splines. Let $\varphi_1, \dots, \varphi_m$ be the standard quadratic B-splines associated with the knot sequence

$$-\pi/2 = x_1 = x_2 = x_3 < x_4 < \cdots < x_m < x_{m+1} = x_{m+2} = x_{m+3} = \pi/2.$$

Recall that φ_i is supported on the interval $[x_i, x_{i+3}]$ and that the B-splines form a partition of unity on $[-\pi/2, \pi/2]$. Let $T_1, \dots, T_{\widetilde{m}}$ be the classical trigonometric B-splines of order 3 defined on the knot sequence $\tilde{x}_1, \dots, \tilde{x}_{\widetilde{m}+3}$, where

$$0 = \tilde{x}_1 < \tilde{x}_2 < \cdots < \tilde{x}_{\widetilde{m}} < 2\pi$$

and $\tilde{x}_{\tilde{m}+i} := \tilde{x}_i + 2\pi$, $i = 1, \dots, 3$; see section 6. Recall that T_j is supported on the interval $[\tilde{x}_j, \tilde{x}_{j+3}]$. Let

$$\tilde{\varphi}_j(x) = \begin{cases} T_j(x), & j = 1, \dots, \widetilde{m} - 2, \\ T_j(x) + T_j(x - 2\pi), & j = \widetilde{m} - 1, \widetilde{m}, \end{cases}$$

be the associated 2π-periodic trigonometric B-splines; see [12]. These splines can be normalized so that for $\phi \in [0, 2\pi]$

$$1 = \sum_{j=1}^{\widetilde{m}} \cos\left(\frac{\tilde{x}_{j+2} - \tilde{x}_{j+1}}{2}\right) \tilde{\varphi}_j(\phi), \tag{2.1}$$

$$\cos(\phi) = \sum_{j=1}^{\widetilde{m}} \cos\left(\frac{\tilde{x}_{j+1} + \tilde{x}_{j+2}}{2}\right) \tilde{\varphi}_j(\phi), \tag{2.2}$$

$$\sin(\phi) = \sum_{j=1}^{\widetilde{m}} \sin\left(\frac{\tilde{x}_{j+1} + \tilde{x}_{j+2}}{2}\right) \tilde{\varphi}_j(\phi). \tag{2.3}$$

Since the boundaries of H corresponding to $\theta = \pm\pi/2$ map to the north and south poles, respectively, a function f of the form (1.1) will be well defined on S if and only if

$$c_{1,j} = f_{\mathcal{S}} \cos\left(\frac{\tilde{x}_{j+2} - \tilde{x}_{j+1}}{2}\right), \quad j = 1, \ldots, \widetilde{m}, \tag{2.4}$$

and

$$c_{m,j} = f_{\mathcal{N}} \cos\left(\frac{\tilde{x}_{j+2} - \tilde{x}_{j+1}}{2}\right), \quad j = 1, \ldots, \widetilde{m}, \tag{2.5}$$

where $f_{\mathcal{S}}$ and $f_{\mathcal{N}}$ are the values at the poles. Now since f is 2π-periodic in the ϕ variable and is C^1 continuous in both variables, we might expect that the corresponding surface $\mathcal{S}_f$ has a continuous tangent plane at nonpolar points. However, since we are working in a parametric setting, more is needed. The following theorem shows that under mild conditions on f which are normally satisfied in practice, we do get tangent plane continuity except at the poles.

THEOREM 2.1. *Suppose f is a spline as in* (1.1) *that satisfies the conditions* (2.4) *and* (2.5)*, and that in addition $f(\theta, \phi) > 0$ for all $(\theta, \phi) \in H$. Then the corresponding surface $\mathcal{S}_f$ is tangent plane continuous at all nonpolar points of S.*

Proof. Since f is a C^1 spline, the partial derivatives f_θ and f_ϕ are continuous on H. Now $t_1(\theta, \phi) := D_\theta[f(\theta, \phi)\boldsymbol{v}(\theta, \phi)]$ and $t_2(\theta, \phi) := D_\phi[f(\theta, \phi)\boldsymbol{v}(\theta, \phi)]$ are two tangents to the surface $\mathcal{S}_f$ at the point $f(\theta, \phi)\boldsymbol{v}(\theta, \phi)$. The normal vector to the surface at this point is given by the cross product $\boldsymbol{n} := t_1 \times t_2$. By the hypotheses, $\boldsymbol{n}$ is continuous, and thus to ensure a continuous tangent plane, it suffices to show that $\boldsymbol{n}$ has positive length (which insures that the surface does not have singular points or cusps). Using Mathematica, it is easy to see that

$$|\boldsymbol{n}(\theta, \phi)|^2 = f(\theta, \phi)^2\big[\cos(\theta)^2 f(\theta, \phi)^2 + f_\phi(\theta, \phi)^2 + \cos(\theta)^2 f_\theta(\theta, \phi)^2\big],$$

which is clearly positive for all values of $(\theta, \phi) \in H$ with $\theta \neq \pm\pi/2$. ∎

With some additional side conditions on the coefficients of f, we can make the surface $\mathcal{S}_f$ also be tangent plane continuous at the poles. The required conditions (cf. [3], [13]) are that

$$c_{2,j} = c_{1,j} + \frac{(x_4 - x_3)}{2}\left[A_{\mathcal{S}} \cos\left(\frac{\tilde{x}_{j+1} + \tilde{x}_{j+2}}{2}\right) + B_{\mathcal{S}} \sin\left(\frac{\tilde{x}_{j+1} + \tilde{x}_{j+2}}{2}\right)\right]$$

and

$$c_{m-1,j} = c_{m,j} - \frac{(x_{m+1} - x_m)}{2}\left[A_{\mathcal{N}} \cos\left(\frac{\tilde{x}_{j+1} + \tilde{x}_{j+2}}{2}\right) + B_{\mathcal{N}} \sin\left(\frac{\tilde{x}_{j+1} + \tilde{x}_{j+2}}{2}\right)\right]$$

for $j = 1, \ldots, \widetilde{m}$, where $A_{\mathcal{S}}$, $B_{\mathcal{S}}$, $A_{\mathcal{N}}$, and $B_{\mathcal{N}}$ are constants.

3. Basic decomposition and reconstruction formulae. Suppose $\mathcal{V}_0, \mathcal{V}_1, \ldots$ is a nested sequence of finite-dimensional linear subspaces of an inner-product space X, i.e.,

$$\mathcal{V}_0 \subset \mathcal{V}_1 \subset \cdots \subset \mathcal{V}_k \subset \cdots.$$

Let

$$\mathcal{V}_k = \mathcal{V}_{k-1} \oplus \mathcal{W}_{k-1}$$

be the corresponding orthogonal decompositions.

For our application, it is convenient to express decomposition and reconstruction in matrix form, as is done, e.g., in [14]. Let $\varphi_{k,1}, \ldots, \varphi_{k,m_k}$ be a basis for $\mathcal{V}_k$, and let $\psi_{k-1,1}, \ldots, \psi_{k-1,n_{k-1}}$ be a basis for $\mathcal{W}_{k-1}$, where $n_{k-1} = m_k - m_{k-1}$. Then by the nestedness, there exists an $m_k \times m_{k-1}$ matrix P_k such that

$$\varphi_{k-1}^T = \varphi_k^T P_k, \tag{3.1}$$

where

$$\varphi_k = (\varphi_{k,1}, \ldots, \varphi_{k,m_k})^T. \tag{3.2}$$

The equation (3.1) is the usual *refinement relation.* Similarly, there exists an $m_k \times n_{k-1}$ matrix Q_k such that

$$\psi_{k-1}^T = \varphi_k^T Q_k, \tag{3.3}$$

where

$$\psi_{k-1} = (\psi_{k-1,1}, \ldots, \psi_{k-1,n_{k-1}})^T. \tag{3.4}$$

Let

$$\begin{aligned} G_k =& (\langle \varphi_{k,i}, \varphi_{k,j} \rangle), \\ H_{k-1} =& (\langle \psi_{k-1,i}, \psi_{k-1,j} \rangle) \end{aligned}$$

be the Gram matrices of size $m_k \times m_k$ and $n_{k-1} \times n_{k-1}$, respectively. It is easy to see that

$$H_{k-1} = Q_k^T G_k Q_k. \tag{3.5}$$

Clearly, the Gram matrices G_k and H_{k-1} are symmetric. The linear independence of the basis functions $\phi_{k,i}$ and of $\psi_{k,i}$ implies that both G_k and H_{k-1} are positive definite and thus nonsingular.

The following lemma shows how to decompose and reconstruct functions in $\mathcal{V}_k$ in terms of functions in $\mathcal{V}_{k-1}$ and $\mathcal{W}_{k-1}$.

LEMMA 3.1. *Let $f_k = \varphi_k^T a_k$ be a function in $\mathcal{V}_k$ associated with a coefficient vector $a_k \in \mathbb{R}^{m_k}$, and let*

$$f_k = f_{k-1} + g_{k-1} \tag{3.6}$$

be its orthogonal decomposition, where

$$f_{k-1} = \varphi_{k-1}^T a_{k-1} \in \mathcal{V}_{k-1}, \qquad g_{k-1} = \psi_{k-1}^T b_{k-1} \in \mathcal{W}_{k-1}.$$

Then

$$\begin{aligned} a_{k-1} =& G_{k-1}^{-1} P_k^T G_k a_k, \\ b_{k-1} =& H_{k-1}^{-1} Q_k^T G_k a_k. \end{aligned}$$

Moreover,

$$a_k = P_k a_{k-1} + Q_k b_{k-1}. \tag{3.7}$$

Proof. To find a_{k-1}, we take the inner-product of both sides of (3.6) with $\varphi_{k-1,i}$ for $i = 1, \ldots, m_{k-1}$. Using the refinement relation (3.1) and the orthogonality of the $\varphi_{k-1,i}$ with $\psi_{k-1,j}$, we get

$$P_k^T G_k a_k = G_{k-1} a_{k-1},$$

which gives the formula for a_{k-1}. If we instead take the inner-products with ψ_{k-1}, we get the formula for b_{k-1}. In view of the linear independence of the functions $\varphi_{k,1}, \ldots, \varphi_{k,m_k}$, the reconstruction formula (3.7) follows immediately from (3.6) and the refinement relations. ∎

Throughout the remainder of this paper we work with inner-products of the form $\langle f, g\rangle = \int_a^b fg$. In this case the function f_{k-1} in the representation (3.6) is nothing more than the best approximation of f_k from $\mathcal{V}_{k-1}$ in the L_2 norm on $[a, b]$.

4. Tensor-product decomposition and reconstruction. In this section we discuss decomposition and reconstruction of functions in tensor product spaces $\mathcal{V}_k \times \widetilde{\mathcal{V}}_\ell$, where the $\mathcal{V}_k$ are as in the previous section and where $\widetilde{\mathcal{V}}_\ell$ are similar subspaces of an inner-product space $\widetilde{X}$. In particular, suppose

$$\widetilde{\mathcal{V}}_0 \subset \widetilde{\mathcal{V}}_1 \subset \cdots \subset \widetilde{\mathcal{V}}_\ell \subset \cdots$$

and that

$$\widetilde{\mathcal{V}}_\ell = \widetilde{\mathcal{V}}_{\ell-1} \oplus \widetilde{\mathcal{W}}_{\ell-1}.$$

Let P_k, Q_k, G_k, and H_k be as in the previous section, and let $\widetilde{P}_\ell$, $\widetilde{Q}_\ell$, $\widetilde{G}_\ell$, and $\widetilde{H}_\ell$ be the analogous matrices associated with the spaces $\widetilde{\mathcal{V}}_\ell$.

THEOREM 4.1. *Let $f_{k,\ell} = \varphi_k^T A_{k,\ell} \tilde{\varphi}_\ell$ be a function in $\mathcal{V}_k \times \widetilde{\mathcal{V}}_\ell$ associated with a coefficient matrix $A_{k,\ell}$. Then $f_{k,\ell}$ has the orthogonal decomposition*

$$f_{k,\ell} = f_{k-1,\ell-1} + g^{(1)}_{k-1,\ell-1} + g^{(2)}_{k-1,\ell-1} + g^{(3)}_{k-1,\ell-1}, \tag{4.1}$$

with

$$\begin{aligned}
f_{k-1,\ell-1} &= \varphi_{k-1}^T A_{k-1,\ell-1} \tilde{\varphi}_{\ell-1} \in \mathcal{V}_{k-1} \times \widetilde{\mathcal{V}}_{\ell-1},\\
g^{(1)}_{k-1,\ell-1} &= \varphi_{k-1}^T B^{(1)}_{k-1,\ell-1} \tilde{\psi}_{\ell-1} \in \mathcal{V}_{k-1} \times \widetilde{\mathcal{W}}_{\ell-1},\\
g^{(2)}_{k-1,\ell-1} &= \psi_{k-1}^T B^{(2)}_{k-1,\ell-1} \tilde{\varphi}_{\ell-1} \in \mathcal{W}_{k-1} \times \widetilde{\mathcal{V}}_{\ell-1},\\
g^{(3)}_{k-1,\ell-1} &= \psi_{k-1}^T B^{(3)}_{k-1,\ell-1} \tilde{\psi}_{\ell-1} \in \mathcal{W}_{k-1} \times \widetilde{\mathcal{W}}_{\ell-1},
\end{aligned}$$

where the matrices $A_{k-1,\ell-1}$, $B^{(1)}_{k-1,\ell-1}$, $B^{(2)}_{k-1,\ell-1}$, and $B^{(3)}_{k-1,\ell-1}$ are computed from the system of equations

$$\begin{aligned}
G_{k-1} A_{k-1,\ell-1} \widetilde{G}_{\ell-1} &= P_k^T D_{k,\ell} \widetilde{P}_\ell,\\
G_{k-1} B^{(1)}_{k-1,\ell-1} \widetilde{H}_{\ell-1} &= P_k^T D_{k,\ell} \widetilde{Q}_\ell,\\
H_{k-1} B^{(2)}_{k-1,\ell-1} \widetilde{G}_{\ell-1} &= Q_k^T D_{k,\ell} \widetilde{P}_\ell,\\
H_{k-1} B^{(3)}_{k-1,\ell-1} \widetilde{H}_{\ell-1} &= Q_k^T D_{k,\ell} \widetilde{Q}_\ell
\end{aligned} \tag{4.2}$$

with

$$D_{k,\ell} := G_k A_{k,\ell} \widetilde{G}_\ell.$$

Moreover,

$$A_{k,\ell} = P_k A_{k-1,\ell-1} \widetilde{P}_\ell^T + P_k B^{(1)}_{k-1,\ell-1} \widetilde{Q}_\ell^T + Q_k B^{(2)}_{k-1,\ell-1} \widetilde{P}_\ell^T + Q_k B^{(3)}_{k-1,\ell-1} \widetilde{Q}_\ell^T. \tag{4.3}$$

Proof. To find the formula for $A_{k-1,\ell-1}$, we take the inner-product of both sides of (4.1) with $\varphi_{k-1,i}$ for $i = 1, \ldots, m_{k-1}$ and with $\tilde{\varphi}_{\ell-1,j}$ for $j = 1, \ldots, \tilde{m}_{\ell-1}$. The formulae for the $B^{(i)}_{k-1,\ell-1}$ are obtained in a similar way. The reconstruction formula (4.3) follows directly from (4.1) after inserting the refinement relations and using the linear independence of the components of the vectors φ_k and in $\tilde{\varphi}_\ell$. ∎

Note that computing the matrices $A_{k-1,\ell-1}$ and $B^{(i)}_{k-1,\ell-1}$ in a decomposition step can be done quite efficiently since several matrix products occur more than once, and we need only solve linear systems of equations involving the four matrices G_{k-1}, H_{k-1}, $\widetilde{G}_{\ell-1}$, and $\widetilde{H}_{\ell-1}$. As we shall see below, in our setting the first two of these are banded matrices, and the second two are periodic versions of banded matrices. All of them can be precomputed and stored in compact form.

5. The decomposition matrices for the polynomial splines. In this section we construct the matrices P_k, Q_k, and G_k needed for the decomposition and reconstruction of quadratic polynomial splines on the closed interval $[-\pi/2, \pi/2]$. Consider the nested sequence of knots

$$-\pi/2 = x_1^k = x_2^k = x_3^k < x_4^k < \cdots < x_{m_k}^k < x_{m_k+1}^k = x_{m_k+2}^k = x_{m_k+3}^k = \pi/2,$$

where

$$x_i^k = -\pi/2 + (i-3)h_k, \qquad i = 4, \ldots, m_k, \tag{5.1}$$

with $h_k = \pi/(m_k - 2)$ and $m_k = 3 \cdot 2^k + 2$. Let $\{\varphi_{k,i}\}_{i=1}^{m_k}$ be the associated quadratic B-splines with supports on the intervals $[x_i^k, x_{i+3}^k]$, $i = 1, \ldots, m_k$. We assume they are normalized so that

$$\sum_{i=1}^{m_k} \varphi_{k,i} \equiv 1. \tag{5.2}$$

We choose this normalization since in performing compression we want to control the error in the uniform norm. For each k, the span $\mathcal{V}_k$ of $\varphi_{k,1}, \ldots, \varphi_{k,m_k}$ is the m_k dimensional linear space of C^1 quadratic splines with knots at the x_i^k. These spaces are clearly nested. In addition to the well-known *refinement relations*

$$\varphi_{k-1,i} = \frac{(\varphi_{k,2i-3} + 3\varphi_{k,2i-2} + 3\varphi_{k,2i-1} + \varphi_{k,2i})}{4}, \qquad i = 3, \ldots, m_{k-1} - 2, \tag{5.3}$$

a simple computation shows that

$$\begin{aligned}
\varphi_{k-1,1} &= (4\varphi_{k,1} + 2\varphi_{k,2})/4, \\
\varphi_{k-1,2} &= (2\varphi_{k,2} + 3\varphi_{k,3} + \varphi_{k,4})/4, \\
\varphi_{k-1,m_{k-1}-1} &= (2\varphi_{k,m_k-1} + 3\varphi_{k,m_k-2} + \varphi_{k,m_k-3})/4, \\
\varphi_{k-1,m_{k-1}} &= (4\varphi_{k,m_k} + 2\varphi_{k,m_k-1})/4.
\end{aligned} \tag{5.4}$$

Equations (5.3), (5.4) provide the entries for the matrix P_k. In particular, the first two and last two columns are determined by (5.4), while for any $3 \le i \le m_{k-1} - 2$, the ith column of P_k contains all zeros except for the four rows $2i-3, \dots, 2i$ which contain the numbers 1/4, 3/4, 3/4, and 1/4. For example,

$$P_1 = \frac{1}{4}\begin{pmatrix} 4 & 0 & 0 & 0 & 0 \\ 2 & 2 & 0 & 0 & 0 \\ 0 & 3 & 1 & 0 & 0 \\ 0 & 1 & 3 & 0 & 0 \\ 0 & 0 & 3 & 1 & 0 \\ 0 & 0 & 1 & 3 & 0 \\ 0 & 0 & 0 & 2 & 2 \\ 0 & 0 & 0 & 0 & 4 \end{pmatrix}.$$

In general, P_k has at most two nonzero entries in each row and at most four nonzero entries in each column.

In order to compute the entries of the matrices Q_k, we need explicit bases for the *wavelet spaces* $\mathcal{W}_{k-1}$ on the interval $[-\pi/2, \pi/2]$. Although their construction is well known, for completeness we give details. Let $n_{k-1} = m_k - m_{k-1} = 3 \cdot 2^{k-1}$.

THEOREM 5.1. *Given $k \ge 1$, let*

$$\psi_{k-1,1} := \sum_{j=1}^{6} q_{j,1}\varphi_{k,j},$$

$$\psi_{k-1,n_{k-1}} := \sum_{j=1}^{6} q_{j,1}\varphi_{k,m_k-j+1},$$

and for $k \ge 2$, let

$$\psi_{k-1,2} := \sum_{j=2}^{8} q_{j,2}\varphi_{k,j},$$

$$\psi_{k-1,n_{k-1}-1} := \sum_{j=2}^{8} q_{j,2}\varphi_{k,m_k-j+1},$$

where

$$\begin{pmatrix} q_{1,1} \\ q_{2,1} \\ q_{3,1} \\ q_{4,1} \\ q_{5,1} \\ q_{6,1} \end{pmatrix} = \frac{1}{14}\begin{pmatrix} -6864 \\ 8346 \\ -4967 \\ 2083 \\ -406 \\ 14 \end{pmatrix}, \qquad \begin{pmatrix} q_{2,2} \\ q_{3,2} \\ q_{4,2} \\ q_{5,2} \\ q_{6,2} \\ q_{7,2} \\ q_{8,2} \end{pmatrix} = \frac{1}{11}\begin{pmatrix} 780 \\ -1949 \\ 3481 \\ -3362 \\ 1618 \\ -319 \\ 11 \end{pmatrix}. \tag{5.5}$$

In addition, for $k \ge 2$, let

$$\begin{aligned} \psi_{k-1,i} := {} & -\varphi_{k,2i-3} + 29\varphi_{k,2i-2} - 147\varphi_{k,2i-1} + 303\varphi_{k,2i} - 303\varphi_{k,2i+1} \\ & + 147\varphi_{k,2i+2} - 29\varphi_{k,2i+3} + \varphi_{k,2i+4} \end{aligned} \tag{5.6}$$

for $i = 3, \dots, n_{k-1} - 2$, and set

$$\psi_{0,2} := -\varphi_{1,2} + \frac{5}{2}\varphi_{1,3} - \frac{9}{2}\varphi_{1,4} + \frac{9}{2}\varphi_{1,5} - \frac{5}{2}\varphi_{1,6} + \varphi_{1,7}.$$

Then $\psi_{k-1,1}, \dots, \psi_{k-1,n_{k-1}}$ form a basis for $\mathcal{W}_{k-1}$.

Proof. The wavelets in (5.6) are just the well-known quadratic spline wavelets; see, e.g., [1]. The coefficients of the boundary wavelets $\psi_{k-1,1}$, $\psi_{k-1,n_{k-1}}$, $\psi_{k-1,2}$, and $\psi_{k-1,n_{k-1}-1}$ can be computed by forcing orthogonality to $\mathcal{V}_{k-1}$. In view of (3.3), the wavelets $\psi_{k-1,1}, \dots, \psi_{k-1,n_{k-1}}$ are linearly independent if and only if the matrix Q_k is of full rank. This follows since the submatrix of Q_k obtained by taking rows $2, 4, \dots, 3 \cdot 2^{k-1}$ and $3 \cdot 2^{k-1} + 3, 3 \cdot 2^{k-1} + 5, \dots, 3 \cdot 2^k + 1$ can easily be seen to be diagonally dominant. For an alternate proof of linear independence, see Lemma 11 of [7]. □

Using elementary properties of B-splines, it is easy to see that

$$\begin{aligned} \psi_{k-1,i}(\pm\pi/2) &= 0, \qquad i = 2, \dots, n_{k-1} - 1, \\ D_\theta \psi_{k-1,i}(\pm\pi/2) &= 0, \qquad i = 3, \dots, n_{k-1} - 2. \end{aligned} \tag{5.7}$$

We now describe the matrices Q_k. By Theorem 5.1,

$$Q_1 = \begin{pmatrix} q_{1,1} & 0 & 0 \\ q_{2,1} & -1 & 0 \\ q_{3,1} & 5/2 & q_{6,1} \\ q_{4,1} & -9/2 & q_{5,1} \\ q_{5,1} & 9/2 & q_{4,1} \\ q_{6,1} & -5/2 & q_{3,1} \\ 0 & 1 & q_{2,1} \\ 0 & 0 & q_{1,1} \end{pmatrix}$$

and

$$Q_2 = \begin{pmatrix} q_{1,1} & 0 & 0 & 0 & 0 & 0 \\ q_{2,1} & q_{2,2} & 0 & 0 & 0 & 0 \\ q_{3,1} & q_{3,2} & -1 & 0 & 0 & 0 \\ q_{4,1} & q_{4,2} & 29 & 0 & 0 & 0 \\ q_{5,1} & q_{5,2} & -147 & -1 & 0 & 0 \\ q_{6,1} & q_{6,2} & 303 & 29 & 0 & 0 \\ 0 & q_{7,2} & -303 & -147 & q_{8,2} & 0 \\ 0 & q_{8,2} & 147 & 303 & q_{7,2} & 0 \\ 0 & 0 & -29 & -303 & q_{6,2} & q_{6,1} \\ 0 & 0 & 1 & 147 & q_{5,2} & q_{5,1} \\ 0 & 0 & 0 & -29 & q_{4,2} & q_{4,1} \\ 0 & 0 & 0 & 1 & q_{3,2} & q_{3,1} \\ 0 & 0 & 0 & 0 & q_{2,2} & q_{2,1} \\ 0 & 0 & 0 & 0 & 0 & q_{1,1} \end{pmatrix}.$$

For general $k \geq 2$, the nonzero elements in the third column of Q_k are repeated in columns $4, \dots, n_{k-1} - 2$, where in each successive column they are shifted down by two rows. The first two and last two columns of Q_k contain the same nonzero elements as Q_2. Clearly, Q_k has at most four nonzero entries in each row and at most eight nonzero entries in each column.

We now describe the Gram matrices G_k, which in general are symmetric and five-banded. To get G_k, we start with the matrix with $66h_k/120$ on the diagonal, $26h_k/120$ on the first subdiagonal, and $h_k/120$ on the second subdiagonal. Then replace the entries in the 3×3 submatrices in the upper left and lower right corners by

$$UL_k := \frac{h_k}{120}\begin{pmatrix} 24 & 14 & 2 \\ 14 & 40 & 25 \\ 2 & 25 & 66 \end{pmatrix}, \qquad LR_k := \frac{h_k}{120}\begin{pmatrix} 66 & 25 & 2 \\ 25 & 40 & 14 \\ 2 & 14 & 24 \end{pmatrix}.$$

For example,

$$G_0 = \frac{h_0}{120}\begin{pmatrix} 24 & 14 & 2 & 0 & 0 \\ 14 & 40 & 25 & 1 & 0 \\ 2 & 25 & 66 & 25 & 2 \\ 0 & 1 & 25 & 40 & 14 \\ 0 & 0 & 2 & 14 & 24 \end{pmatrix}$$

and

$$G_1 = \frac{h_1}{120}\begin{pmatrix} 24 & 14 & 2 & 0 & 0 & 0 & 0 & 0 \\ 14 & 40 & 25 & 1 & 0 & 0 & 0 & 0 \\ 2 & 25 & 66 & 26 & 1 & 0 & 0 & 0 \\ 0 & 1 & 26 & 66 & 26 & 1 & 0 & 0 \\ 0 & 0 & 1 & 26 & 66 & 26 & 1 & 0 \\ 0 & 0 & 0 & 1 & 26 & 66 & 25 & 2 \\ 0 & 0 & 0 & 0 & 1 & 25 & 40 & 14 \\ 0 & 0 & 0 & 0 & 0 & 2 & 14 & 24 \end{pmatrix}.$$

6. The decomposition matrices for the trigonometric splines. In this section we present the matrices $\widetilde{P}_\ell$, $\widetilde{Q}_\ell$, and $\widetilde{G}_\ell$ needed for the decomposition and reconstruction of periodic trigonometric splines of order 3 defined on the interval $[0, 2\pi]$. Suppose $\ell \geq 1$ and that

$$\tilde{x}_i^\ell = (i-1)\tilde{h}_\ell, \qquad i = 1, \ldots, \widetilde{m}_\ell + 3, \tag{6.1}$$

is a nested sequence of knots, where $\tilde{h}_\ell = 2^{(1-\ell)}\pi/3$ and $\widetilde{m}_\ell = 3 \cdot 2^\ell$. Let

$$M_{\ell,i}(\phi) := T_{\tilde{h}_\ell}(\phi - \tilde{x}_i^\ell),$$

where

$$T_h(\phi) := \begin{cases} \frac{\sin(\phi/2)^2}{\sin(h/2)\sin(h)}, & 0 \leq \phi \leq h, \\ \frac{1}{\cos(h/2)} - \frac{\sin((\phi-h)/2)^2 + \sin((2h-\phi)/2)^2}{\sin(h/2)\sin(h)}, & h \leq \phi \leq 2h, \\ \frac{\sin((3h-\phi)/2)^2}{\sin(h/2)\sin(h)}, & 2h \leq \phi \leq 3h, \\ 0, & \text{otherwise} \end{cases} \tag{6.2}$$

is the usual trigonometric B-spline of order 3 associated with uniformly spaced knots $(0, h, 2h, 3h)$. Set

$$\tilde{\varphi}_{\ell,i}(\phi) = \begin{cases} M_{\ell,i}(\phi), & i = 1, \ldots, \widetilde{m}_\ell - 2, \\ M_{\ell,i}(\phi) + M_{\ell,i}(\phi - 2\pi), & i = \widetilde{m}_\ell - 1, \widetilde{m}_\ell. \end{cases}$$

For later use we define $\tilde{\varphi}_{\ell,\widetilde{m}_\ell+i} = \tilde{\varphi}_{\ell,i}$ for $i = 1, \ldots, 6$. With this normalization, the $\tilde{\varphi}_{\ell,i}$ satisfy (2.1)–(2.3), and it follows that

$$\sum_{i=1}^{\widetilde{m}_\ell} \tilde{\varphi}_{\ell,i} \equiv \frac{1}{\cos(\tilde{h}_\ell)} \le 2 \tag{6.3}$$

for $\ell \ge 1$.

The span $\widetilde{\mathcal{V}}_\ell$ of $\tilde{\varphi}_{\ell,1}, \ldots, \tilde{\varphi}_{\ell,\widetilde{m}_\ell}$ is the space of periodic trigonometric splines of order 3. Clearly, these spaces are nested, and in fact we have the following *refinement relation*.

THEOREM 6.1. *For all $\ell \ge 1$ and $1 \le i \le \tilde{m}_{\ell-1}$,*

$$\tilde{\varphi}_{\ell-1,i} = u(\tilde{h}_\ell)\tilde{\varphi}_{\ell,2i-1} + v(\tilde{h}_\ell)\tilde{\varphi}_{\ell,2i} + v(\tilde{h}_\ell)\tilde{\varphi}_{\ell,2i+1} + u(\tilde{h}_\ell)\tilde{\varphi}_{\ell,2i+2}, \tag{6.4}$$

where

$$u(h) := \frac{1}{4\cos(h/2)\cos(h)}, \qquad v(h) := \frac{\cos(h/2)}{\cos(h)} - u(h).$$

Proof. By nestedness and the nature of the support of T_h,

$$T_{2h}(\phi) = u(h)T_h(\phi) + v(h)T_h(\phi - h) + w(h)T_h(\phi - 2h) + z(h)T_h(\phi - 3h)$$

for some numbers u, v, w, z. By symmetry, it is enough to compute u and v. To find u, we note that on $[0, h]$,

$$T_{2h}(\phi) = u(h)T_h(\phi).$$

Then using (6.2) we can solve for u. To find v we note that

$$T_{2h}(2h) = u(h)T_h(2h) + v(h)T_h(h),$$

and then solve for v using (6.2). □

Theorem 6.1 can now be used to find the entries in the matrix $\widetilde{P}_\ell$ needed in section 2. In particular, each column has exactly the four nonzero elements $u(\tilde{h}_\ell), v(\tilde{h}_\ell), v(\tilde{h}_\ell), u(\tilde{h}_\ell)$, starting in the first row in column 1, and shifted down by two rows each time we move one column to the right (where in the last column the last two elements are moved to the top of the column). For example,

$$\widetilde{P}_1 := \begin{pmatrix} u(\tilde{h}_1) & 0 & v(\tilde{h}_1) \\ v(\tilde{h}_1) & 0 & u(\tilde{h}_1) \\ v(\tilde{h}_1) & u(\tilde{h}_1) & 0 \\ u(\tilde{h}_1) & v(\tilde{h}_1) & 0 \\ 0 & v(\tilde{h}_1) & u(\tilde{h}_1) \\ 0 & u(\tilde{h}_1) & v(\tilde{h}_1) \end{pmatrix}.$$

Next we describe a basis for the *wavelet space* $\widetilde{\mathcal{W}}_{\ell-1}$ on $[0, 2\pi]$ which has dimension $\tilde{n}_{\ell-1} = 3 \cdot 2^{\ell-1}$ for $\ell \ge 1$.

THEOREM 6.2. *Given $\ell \ge 1$, let*

$$\tilde{\psi}_{\ell-1,i} = \sum_{j=0}^{7} \tilde{q}_j(\tilde{h}_\ell)\tilde{\varphi}_{\ell,2i+j-1}, \quad i = 1, \ldots, \tilde{n}_{\ell-1}, \tag{6.5}$$

where

$$\begin{aligned}
\tilde{q}_0(h) &= 1,\\
\tilde{q}_1(h) &= \frac{-h + 5h\cos(h) - h\cos(2h) - 3\sin(h)}{D(h)},\\
\tilde{q}_2(h) &= \frac{3h - 7h\cos(h) - 5h\cos(2h) + 3\sin(3h)}{D(h)},\\
\tilde{q}_3(h) &= \frac{-2h - 7h\cos(h) + 4h\cos(2h) - 4h\cos(3h) + 3\sin(3h)}{D(h)},
\end{aligned}$$

and

$$\tilde{q}_{7-j}(h) = -\tilde{q}_j(h), \qquad j = 0, \ldots, 3,$$

with

$$D(h) := 2h + h\cos(h) - 3\sin(h).$$

Then $\tilde{\psi}_{\ell-1,1}, \ldots, \tilde{\psi}_{\ell-1,\tilde{n}_{\ell-1}}$ *is a basis for the space* $\widetilde{\mathcal{W}}_{\ell-1}$.

Proof. To construct wavelets in $\widetilde{\mathcal{W}}_{\ell-1}$, we apply [8, Theorem 5.1], which gives explicit formulae for the $\tilde{q}_i$ in terms of inner-products of $\tilde{\varphi}_{\ell,i}$ with $\tilde{\varphi}_{\ell-1,j}$. To show that $\tilde{\psi}_{\ell-1,1}, \ldots, \tilde{\psi}_{\ell-1,\tilde{n}_{\ell-1}}$ are linearly independent, it suffices to show that $\widetilde{Q}_\ell$ is of full rank. To see this, we construct an $\tilde{n}_{\ell-1} \times \tilde{n}_{\ell-1}$ matrix B_ℓ by moving the last column of $\widetilde{Q}_\ell$ in front of the first column and then selecting rows $2, 4, \ldots, \widetilde{m}_\ell$. We now show that this matrix is strictly diagonally dominant, and thus of full rank.

First, we note that in each row of B_ℓ the element on the diagonal is $\tilde{q}_3(\tilde{h}_\ell)$ while the sum of the absolute values of the off-diagonal elements is $|\tilde{q}_1(\tilde{h}_\ell)| + |\tilde{q}_5(\tilde{h}_\ell)| + |\tilde{q}_7(\tilde{h}_\ell)|$. A simple computation shows that each of the functions $D(h)$ and $r_i(h) := \tilde{q}_i(h)D(h)$ has a Taylor expansion which is an alternating series. In particular, using the first two terms of each series, we get

$$\begin{aligned}
D(h) &> h^5[1/60 - h^2/1260] > 0,\\
r_1(h) &< h^5[-29/60 + 26h^2/315] < 0,\\
r_2(h) &> h^5[49/20 - 89h^2/105] > 0,\\
r_3(h) &< h^5[-101/20 + 1009h^2/420] < 0
\end{aligned}$$

for $0 \le h \le \pi/3 = \tilde{h}_1$. Now it is easy to see that

$$a(h) := \big[|\tilde{q}_3(h)| - |\tilde{q}_1(h)| - |\tilde{q}_5(h)| - |\tilde{q}_7(h)|\big]D(h) = -r_3(h) - r_2(h) + r_1(h) - D(h)$$

also has an alternating series expansion, and we get

$$a(h) > h^5[71/10 - 103h^2/70] > 0$$

for the same range of h. This shows that B_ℓ is strictly diagonally dominant, and the proof is complete. ∎

The formulae for the $\tilde{q}_i$ in Theorem 6.2 are not appropriate for small values of $\tilde{h}_{\ell-1}$. In this case we can use the following Taylor expansions:

$$\begin{aligned}
\tilde{q}_0(h) &= 1,\\
\tilde{q}_1(h) &= -29 + \frac{25}{7}h^2 - \frac{103}{588}\,h^4 + \frac{1255}{271656}\,h^6 + \cdots,\\
\tilde{q}_2(h) &= 147 - \frac{307}{7}\,h^2 + \frac{3301}{588}\,h^4 - \frac{545273}{1358280}\,h^6 + \cdots,\\
\tilde{q}_3(h) &= -303 + \frac{908}{7}\,h^2 - \frac{3131}{147}\,h^4 + \frac{642583}{339570}\,h^6 + \cdots.
\end{aligned}$$

Rather than computing them each time we need them, we can precompute and store the necessary values of $\tilde{q}_1(\tilde{h}_\ell)$, $\tilde{q}_2(\tilde{h}_\ell)$, and $\tilde{q}_3(\tilde{h}_\ell)$ for various levels ℓ; see Table 1. We can now describe the matrix $\widetilde{Q}_\ell$ needed in section 2 for decomposing and reconstructing with trigonometric splines. For $\ell = 1$ we have

$$\widetilde{Q}_1 = \begin{pmatrix}
\tilde{q}_0 + \tilde{q}_6 & \tilde{q}_4 & \tilde{q}_2\\
\tilde{q}_1 + \tilde{q}_7 & \tilde{q}_5 & \tilde{q}_3\\
\tilde{q}_2 & \tilde{q}_0 + \tilde{q}_6 & \tilde{q}_4\\
\tilde{q}_3 & \tilde{q}_1 + \tilde{q}_7 & \tilde{q}_5\\
\tilde{q}_4 & \tilde{q}_2 & \tilde{q}_0 + \tilde{q}_6\\
\tilde{q}_5 & \tilde{q}_3 & \tilde{q}_1 + \tilde{q}_7
\end{pmatrix},$$

where all $\tilde{q}_i$ are evaluated at $\tilde{h}_1$. For $\ell \geq 2$, each column of $\widetilde{Q}_\ell$ contains the eight entries $\tilde{q}_0, \tilde{q}_1, \tilde{q}_2, \tilde{q}_3, \tilde{q}_4, \tilde{q}_5, \tilde{q}_6, \tilde{q}_7$, evaluated at $\tilde{h}_\ell$. In particular, these entries start in row 1 in column 1 and are shifted down by two each time we move one column to the right (where in the last three columns, entries falling below the last row are moved to the top). Clearly, $\widetilde{Q}_\ell$ has exactly four nonzero entries in each row. For example,

$$\widetilde{Q}_2 = \begin{pmatrix}
\tilde{q}_0 & 0 & 0 & \tilde{q}_6 & \tilde{q}_4 & \tilde{q}_2\\
\tilde{q}_1 & 0 & 0 & \tilde{q}_7 & \tilde{q}_5 & \tilde{q}_3\\
\tilde{q}_2 & \tilde{q}_0 & 0 & 0 & \tilde{q}_6 & \tilde{q}_4\\
\tilde{q}_3 & \tilde{q}_1 & 0 & 0 & \tilde{q}_7 & \tilde{q}_5\\
\tilde{q}_4 & \tilde{q}_2 & \tilde{q}_0 & 0 & 0 & \tilde{q}_6\\
\tilde{q}_5 & \tilde{q}_3 & \tilde{q}_1 & 0 & 0 & \tilde{q}_7\\
\tilde{q}_6 & \tilde{q}_4 & \tilde{q}_2 & \tilde{q}_0 & 0 & 0\\
\tilde{q}_7 & \tilde{q}_5 & \tilde{q}_3 & \tilde{q}_1 & 0 & 0\\
0 & \tilde{q}_6 & \tilde{q}_4 & \tilde{q}_2 & \tilde{q}_0 & 0\\
0 & \tilde{q}_7 & \tilde{q}_5 & \tilde{q}_3 & \tilde{q}_1 & 0\\
0 & 0 & \tilde{q}_6 & \tilde{q}_4 & \tilde{q}_2 & \tilde{q}_0\\
0 & 0 & \tilde{q}_7 & \tilde{q}_5 & \tilde{q}_3 & \tilde{q}_1
\end{pmatrix},$$

where all $\tilde{q}_i$ are evaluated at $\tilde{h}_2$. Table 1 gives the values of $\tilde{q}_1(\tilde{h}_k)$, $\tilde{q}_2(\tilde{h}_\ell)$, and $\tilde{q}_3(\tilde{h}_\ell)$ for $\ell = 1, \ldots, 12$ needed for the $\widetilde{Q}_\ell$.

Finally, we describe the Gram matrices.

THEOREM 6.3. *For $\ell \geq 1$, the $3 \cdot 2^\ell \times 3 \cdot 2^\ell$ Gram matrix $\widetilde{G}_\ell$ associated with the*

TABLE 1
Trigonometric spline wavelet coefficients for various ℓ.

ℓ	$\tilde{q}_1(\tilde{h}_\ell)$	$\tilde{q}_2(\tilde{h}_\ell)$	$\tilde{q}_3(\tilde{h}_\ell)$
1	−25.288158402784911895	105.15263361113964758	−184.01710881949438326
2	−28.033943811096385992	135.39009725820806026	−269.00057271914225083
3	−28.756039535012008061	144.02032194736046124	−294.20897139729685258
4	−28.938855942719881876	146.25016593522229565	−300.78362470238002386
5	−28.984704348047217637	146.81223291013457079	−302.44473588115810747
6	−28.996175484404513950	146.95303891951439472	−302.86111072242944246
7	−28.999043833434183593	146.98825852278080426	−302.96527310098241998
8	−28.999760956004299506	146.99706455524617238	−302.99131798899351388
9	−28.999940238853933503	146.99926613409589417	−302.99782947935722381
10	−28.999985059704287025	146.99981653322924416	−302.99945736872110255
11	−28.999996264925496984	146.99995413328889043	−302.99986434211038783
12	−28.999999066231338323	146.99998853332107132	−302.99996608552322897

$\tilde{\varphi}_{\ell,i}$ is given by

$$\widetilde{G}_\ell := \begin{pmatrix} I_{00} & I_{01} & I_{02} & 0 & \cdots & 0 & I_{02} & I_{01} \\ I_{01} & I_{00} & I_{01} & I_{02} & 0 & \cdots & 0 & I_{02} \\ I_{02} & I_{01} & I_{00} & I_{01} & I_{02} & \cdots & 0 & 0 \\ & \ddots & & \ddots & & \ddots & & \\ & & \ddots & & \ddots & & \ddots & \\ 0 & 0 & \cdots & I_{02} & I_{01} & I_{00} & I_{01} & I_{02} \\ I_{02} & 0 & \cdots & 0 & I_{02} & I_{01} & I_{00} & I_{01} \\ I_{01} & I_{02} & 0 & \cdots & 0 & I_{02} & I_{01} & I_{00} \end{pmatrix},$$

where

$$I_{02} := \int_{\tilde{x}_i^\ell}^{\tilde{x}_{i+1}^\ell} \tilde{\varphi}_{\ell,i}\ \tilde{\varphi}_{\ell,i-2} = \gamma_\ell[4\tilde{h}_\ell + 2\tilde{h}_\ell\cos(2\tilde{h}_\ell) - 3\sin(2\tilde{h}_\ell)],$$

$$I_{01} := \int_{\tilde{x}_i^\ell}^{\tilde{x}_{i+2}^\ell} \tilde{\varphi}_{\ell,i}\ \tilde{\varphi}_{\ell,i-1} = \gamma_\ell[-4\tilde{h}_\ell - 20\tilde{h}_\ell\cos(2\tilde{h}_\ell) + 6\sin(2\tilde{h}_\ell) + 3\sin(4\tilde{h}_\ell)],$$

$$I_{00} := \int_{\tilde{x}_i^\ell}^{\tilde{x}_{i+3}^\ell} \tilde{\varphi}_{\ell,i}^2 = \gamma_\ell[4\tilde{h}_\ell\cos(2\tilde{h}_\ell) + 8\tilde{h}_\ell\cos(4\tilde{h}_\ell) + 24\tilde{h}_\ell - 6\sin(2\tilde{h}_\ell) - 6\sin(4\tilde{h}_\ell)],$$

with

$$\gamma_\ell := \frac{1}{64\sin(\tilde{h}_\ell)^4\cos(\tilde{h}_\ell)^2}.$$

Moreover,

$$\widetilde{G}_0 := \begin{pmatrix} I_{00} & I_{01}+I_{02} & I_{01}+I_{02} \\ I_{01}+I_{02} & I_{00} & I_{01}+I_{02} \\ I_{01}+I_{02} & I_{01}+I_{02} & I_{00} \end{pmatrix}.$$

TABLE 2
Inner products of trigonometric B-splines for various ℓ.

ℓ	$I_{00}/\tilde{h}_\ell$	$I_{01}/\tilde{h}_\ell$	$I_{02}/\tilde{h}_\ell$
0	2.000000000000000000	0.9423111143775626914	0.05766888562243730858
1	0.7173865882718287392	0.29529339212946177894	0.012679980401290518133
2	0.5863256235682111689	0.23350674787359713392	0.009228825204542694601
3	0.5587848830466661676,	0.22072647211850900468	0.008547276418657547047
4	0.5521783423263619826	0.21767257645539869275	0.008386225140326574126
5	0.5505434780490859489	0.21691758424366982979	0.008346519562452568337
6	0.5501358004443718685	0.21672936114999970712	0.008336627601639628844
7	0.5500339457967328606	0.21668233810682990252	0.008334156757445556228
8	0.5500084861795729324	0.21667058439043531220	0.008333539180427623907
9	0.5500021215280431720	0.21666764608909212581	0.008333384794548569195
10	0.5500005303809576733	0.21666691152174074237	0.008333346198602246618
11	0.5500001325951735985	0.21666672788040191760	0.008333336549648380680
12	0.5500000331487892859	0.21666668197009840015	0.008333334137411958859

Proof. Using (6.2), the necessary integrals can be computed directly. ∎

The formulae in Theorem 6.3 are clearly not appropriate for small values of $\tilde{h}_\ell$, in which case the following formulae can be used:

$$I_{02} = \frac{\tilde{h}_\ell}{120}\left[1 + \frac{31}{21}\tilde{h}_\ell^2 + \frac{134}{105}\tilde{h}_\ell^4 + \frac{2971}{3465}\tilde{h}_\ell^6 + \cdots\right],$$

$$I_{01} = \frac{13\tilde{h}_\ell}{60}\left[1 + \frac{295}{273}\tilde{h}_\ell^2 + \frac{146}{195}\tilde{h}_\ell^4 + \frac{299}{693}\tilde{h}_\ell^6 + \cdots\right],$$

$$I_{00} = \frac{11\tilde{h}_\ell}{20}\left[1 + \frac{71}{77}\tilde{h}_\ell^2 + \frac{674}{1155}\tilde{h}_\ell^4 + \frac{12233}{38115}\tilde{h}_\ell^6 + \cdots\right].$$

Table 2 contains the values of $I_{00}/\tilde{h}_\ell$, $I_{01}/\tilde{h}_\ell$, and $I_{02}/\tilde{h}_\ell$ for $\ell = 1, \ldots, 12$.

7. Implementation.

7.1. Decomposition. The decomposition procedure begins with a tensor spline of the form (1.1) based on polynomial splines $\varphi_{k,i}(\theta)$ at a given level $k \geq 1$ and periodic trigonometric splines $\tilde{\varphi}_{\ell,j}(\phi)$ at a given level $\ell \geq 1$ with coefficient matrix $C := A_{k,\ell}$ of size $m_k \times \widetilde{m}_\ell$. To carry out one step of the decomposition, we solve the systems (4.2) for $A_{k-1,\ell-1}$, $B^{(1)}_{k-1,\ell-1}$, $B^{(2)}_{k-1,\ell-1}$, $B^{(3)}_{k-1,\ell-1}$ and set

$$C = \begin{pmatrix} A_{k-1,\ell-1} & B^{(1)}_{k-1,\ell-1} \\ B^{(2)}_{k-1,\ell-1} & B^{(3)}_{k-1,\ell-1} \end{pmatrix}. \tag{7.1}$$

To continue the decomposition, we now carry out the same procedure on the matrix $A_{k-1,\ell-1}$. This process can be repeated at most $\min(k,\ell)-1$ times, where at each step the new spline coefficients and wavelet coefficients are stored in C. Thus, the entire decomposition process requires no additional storage beyond the original coefficient matrix.

The systems of equations (4.2) which have to be solved to carry out each step of the decomposition process can be efficiently solved due to their tensor nature. All of the matrices that have to be inverted are symmetric positive definite and are either banded or periodic banded. In particular, G_k is five-banded and H_k is seven-banded, while $\widetilde{G}_\ell$ and $\widetilde{H}_\ell$ are periodic five-banded and seven-banded, respectively. In our implementation we created special code to compute the band-Cholesky decompositions.

The matrices $H_k, G_k, \widetilde{G}_\ell$, and $\widetilde{H}_\ell$ are fixed and do not depend on the particular spline being processed. Thus, they can be precomputed and stored in the program.

7.2. Thresholding. Typically, at each step of the decomposition, many of the entries in the matrices $B^{(i)}_{k-j,\ell-j}$ of wavelet coefficients will be quite small. Thus, to achieve compression, these can be removed by a thresholding process. In view of (5.7), tangent plane continuity will be maintained at the poles if we retain all coefficients in the first two and last two rows of these matrices. As a guide to the thresholding process, we now estimate the effect of removing a set of wavelet coefficients.

THEOREM 7.1. *Suppose*

$$s = (\,\varphi^T_{k-1} \quad \psi^T_{k-1}\,)\, C \begin{pmatrix} \tilde{\varphi}_{\ell-1} \\ \tilde{\psi}_{\ell-1} \end{pmatrix},$$

where C is the matrix in (7.1) *and $\varphi^T_{k-1}, \psi^T_{k-1}, \tilde{\varphi}_{\ell-1}, \tilde{\psi}_{\ell-1}$ are vectors of basis functions as in* (3.2) *and* (3.4). *Given $\varepsilon > 0$, let $\hat{s}$ be the associated spline where the coefficients c_{ij} in C are set to zero whenever*

$$|c_{ij}| < \begin{cases} \varepsilon & \text{if } c_{ij} \in B^{(1)}_{k-1,\ell-1} \cup B^{(2)}_{k-1,\ell-1}, \\ \varepsilon/300 & \text{if } c_{ij} \in B^{(3)}_{k-1,\ell-1}. \end{cases}$$

Then

$$\|s - \hat{s}\|_\infty \le 4000\ \varepsilon. \tag{7.2}$$

Proof. Let Z_ν be the sets of indices i, j of coefficients in $B^{(\nu)}_{k-1,\ell-1}$ which are set to zero. Then

(7.3)

$$s - \hat{s} = \sum_{(i,j)\in Z_1} c_{ij}\varphi_{k-1,i}\tilde{\psi}_{\ell-1,j} + \sum_{(i,j)\in Z_2} c_{ij}\psi_{k-1,i}\tilde{\varphi}_{\ell-1,j} + \sum_{(i,j)\in Z_3} c_{ij}\psi_{k-1,i}\tilde{\psi}_{\ell-1,j}.$$

We now estimate the size of the first sum. Let $\tilde{q}^{(\ell)}_{rj}$ be the entries of the matrix $\widetilde{Q}_\ell$, so that $\tilde{\psi}_{\ell-1,j} = \sum_r \tilde{q}^{(\ell)}_{rj}\tilde{\varphi}_{\ell,r}$. Then using $|c_{ij}| < \varepsilon$ along with (6.3) and (5.2), we have

$$\left| \sum_{(i,j)\in Z_1} c_{ij}\varphi_{k-1,i} \sum_r \tilde{q}^{(\ell)}_{rj}\tilde{\varphi}_{\ell,r} \right| \le \varepsilon \sum_r \left(\sum_j |\tilde{q}^{(\ell)}_{rj}| \right) \tilde{\varphi}_{\ell,r} \le 2\varepsilon \|\widetilde{Q}_\ell\|_\infty,$$

where $\|\widetilde{Q}_\ell\|_\infty$ is the matrix norm subordinate to the vector infinity norm.

Similar estimates hold for the other two sums in (7.3), and thus

$$|(s - \hat{s})(\theta, \phi)| \le \varepsilon \left(2\|\widetilde{Q}_\ell\|_\infty + \|Q_k\|_\infty + \frac{2\|\widetilde{Q}_\ell\|_\infty \|Q_k\|_\infty}{300} \right). \tag{7.4}$$

It is clear that

$$\|Q_k\|_\infty = \frac{51363}{77} < 668,$$

and using Mathematica, it can be seen that

$$\|\widetilde{Q}_\ell\|_\infty = \frac{128\tilde{h}_\ell \cos(\frac{\tilde{h}_\ell}{2})^2 \sin(\frac{\tilde{h}_\ell}{2})^4}{2\tilde{h}_\ell + \tilde{h}_\ell \cos(\tilde{h}_\ell) - 3\sin(\tilde{h}_\ell)} \le 480.$$

The last inequality follows by expanding the fraction into an alternating series in $\tilde{h}_\ell$ around 0. Inserting these estimates in (7.4) gives (7.2). □

The large size of the constant in (7.2) is due to the way in which the wavelets have been scaled, and in fact, is quite pessimistic. In practice we have observed numerically that $\|s - \hat{s}\|_\infty$ is usually about 100 times as large as ε.

Theorem 7.1 leads immediately to several reasonable thresholding algorithms. For example, we can simply remove coefficients as described in the theorem. Alternatively (and this is what we do in the experiments below), at the jth level we can use a threshold of $\varepsilon/2^{j-1}$ for the coefficients in $B^{(1)}_{k-j,\ell-j} \cup B^{(2)}_{k-j,\ell-j}$ and a threshold of $\varepsilon/(300 \cdot 2^{j-1})$ for the coefficients in $B^{(3)}_{k-j,\ell-j}$. This level-dependent thresholding scheme retains more coefficients at the coarser levels.

7.3. Reconstruction. In view of (4.3), to carry out one reconstruction step simply involves matrix multiplication involving the matrices $P_k, Q_k, \widetilde{P}_\ell$, and $\widetilde{Q}_\ell$. These matrices are fixed and do not depend on the particular spline being operated on. Since they are also very sparse, we can simply store their entries in the program.

7.4. Storage. Due to their special structure, all of the fixed matrices needed to describe both decomposition and reconstruction can be stored in band form. This will require $\mathcal{O}(\max(N, \tilde{N}))$ storage, assuming that the largest starting coefficient matrix is of size $N \times \tilde{N}$. It is clear that in carrying out both decomposition and reconstruction, all of the coefficients arising in the intermediate steps can be stored in the original coefficient matrix C. Thus, it follows that the *total amount of storage* needed is $N \times \tilde{N} + \mathcal{O}(\max(N, \tilde{N}))$.

7.5. Complexity. To carry out one decomposition step on an initial coefficient matrix C of size $N \times \tilde{N}$, we need to solve $\tilde{N}$ systems of size $N \times N$ and then N systems of size $\tilde{N} \times \tilde{N}$. Since these systems are at most seven-banded (or periodic seven-banded) and positive definite, the total operation count is $\mathcal{O}(N \times \tilde{N})$. Since at each level of decomposition the number of coefficients to be dealt with is reduced by a factor of 4, it follows that the *total operation count for decomposition* is of order $\mathcal{O}(N \times \tilde{N})$.

To compute the total operation count for reconstruction, we first discuss the step that produces the coefficients at the finest level. This step requires multiplication by matrices with at most eight nonzero elements in each row or column. Thus the complexity for this step is $\mathcal{O}(N \times \tilde{N})$. Adding up the operations required for the coarser steps, it follows that the *total operation count for reconstruction* is also $\mathcal{O}(N \times \tilde{N})$.

7.6. Conditioning. Using Mathematica, we computed the sup-norm condition numbers of the matrices G_k, H_k, $\widetilde{G}_\ell$, and $\widetilde{H}_\ell$ for $1 \le k \le 9$. The results are shown in Table 3. Since none of the condition numbers exceeds 13, we conclude that the algorithm is highly robust.

TABLE 3
Condition numbers of the Gram matrices.

k	G_k	$\widetilde{G}_k$	H_k	$\widetilde{H}_k$
1	12.9273	8.76272	4.09545	4.85975
2	12.8056	7.77962	4.94326	4.36433
3	12.8048	7.56795	5.01714	4.25737
4	12.8048	7.51687	5.01762	4.23155
5	12.8048	7.50421	5.01762	4.22515
6	12.8048	7.50105	5.01762	4.22355
7	12.8048	7.50026	5.01762	4.22316
8	12.8048	7.50007	5.01762	4.22306
9	12.8048	7.50002	5.01762	4.22303

TABLE 4
Computational times.

.

k	ℓ	Coefficients	Seconds
6	6	37,248	.87
7	7	148,224	3.51
8	8	591,360	14.19
8	9	1,182,720	28.76
9	9	2,362,368	57.41

7.7. Timing. Due to the efficient coding, both decomposition and reconstruction are extremely fast. Table 4 lists the computational times for full decomposition and reconstruction for various choices of k and ℓ. The table clearly shows that the computational time depends linearly on the number of coefficients in the initial spline. The computation was done on a Sun Ultra 1 with a 167 MHz Sun UltraSPARC CPU with 512 KB cache.

8. Almost tangent plane continuity. In the method described above, we do not allow the thresholding process to remove coefficients in the first two and last two rows of the wavelet matrices $B^{(i)}_{k-j,\ell-j}$. This is required to guarantee tangent plane continuity at the poles and is a disadvantage since retaining these coefficients reduces compression rates.

It is easy to see that the number of retained coefficients at the poles at level k, ℓ is at most $6\widetilde{m}_\ell$, and thus after performing $\min(k,\ell)-1$ decomposition steps, at most $12\widetilde{m}_\ell$ coefficients are retained at the poles. Since the number of initial coefficients is $m_k \times \widetilde{m}_\ell$, it follows that the compression rate is bounded above by

$$\frac{m_k \times \widetilde{m}_\ell}{12\widetilde{m}_\ell} = \frac{m_k}{12} \approx 2^{k-2}.$$

The limit on compression rates with tangent plane continuity is thus 64 and 128 for $k = 8$ and $k = 9$, respectively.

It is easy to modify the thresholding process described in section 7.2 to get much higher compression rates at the expense of losing exact smoothness at the poles. There are two possibilities:

(T^0) We allow coefficients in the second and next-to-last rows of the matrices $B^{(i)}_{k-j,\ell-j}$ to be removed by thresholding,

(T^{-1}) We allow all coefficients in matrices $B^{(i)}_{k-j,\ell-j}$ to be removed by thresholding.

In both cases we get a reconstructed surface that is still tangent plane continuous at all nonpolar points. With strategy T^0, we lose exact tangent-plane continuity at the poles (but still have it approximately). With strategy T^{-1}, we may even lose continuity of the surface at the poles. If we are willing to accept minor artifacts at the poles, then much higher compression rates can be achieved. The corresponding algorithms based on thresholding strategies T^0 and T^{-1} both require exactly the same storage as in the restricted case, and also have the same efficiency and stability properties. We compare the performance of the various algorithms in the following section.

9. Examples. To test the general performance of the algorithms, we begin with the following simple example.

Example 1. Let $k = 8$ and $\ell = 9$, and let s be the tensor spline with coefficients

$$c_{ij} = \cos((\tilde{x}^9_{j+2} - \tilde{x}^9_{j+1})/2), \qquad i = 1, \ldots, m_8, \quad j = 1, \ldots, \widetilde{m}_9.$$

Discussion. Since the normalized quadratic B-splines form a partition of unity, it follows from (2.1) that with these coefficients, $s \equiv 1$ for all $(\theta, \phi) \in H$, i.e, the corresponding surface is exactly the unit sphere. In this case the coefficient matrix is of size 770×1536 and involves 1,182,720 coefficients. To test the algorithms, we performed decomposition with various values of ε, including zero. In all cases, after reconstruction we got coefficients that were correct to machine accuracy (working in double precision). □

To illustrate the ability of our multiresolution approach to achieve high levels of compression while retaining important features of a surface, we created a tensor spline fit to a smooth surface with a number of bumps.

Example 2. Let $\mathcal{B}$ be the surface shown in the upper left-hand corner of Figure 1.

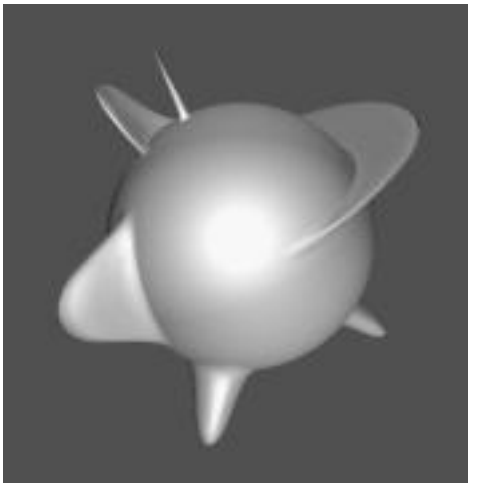
Discussion. The surface $\mathcal{B}$ was created by fitting a spline $f_{8,8}$ to data created by choosing 10 random sized subrectangles at random positions in H and adding tensor product quadratic B-splines of maximum height 3/4 with support on each such rectangle to the constant values corresponding to the unit sphere. For $k = \ell = 8$, the coefficient matrix is of size 770×768 and involves 591,360 coefficients. To test the algorithms, we performed decomposition with the thresholding values $\varepsilon_r = 10^{-r}$ for $r = 1, \ldots, 9$. Table 5 shows the number of coefficients *nco* remaining at each level for the case where $\varepsilon = .001$. Almost 3/4 of the coefficients are removed in the first step of decomposition, and after seven steps we end up with only 9734 coefficients (which amounts to a 60:1 compression ratio). Table 6 shows the differences between the original coefficients and the coefficients obtained after reconstruction. The table lists both the *maximum norm*

$$e_\infty := \max_{ij} |c_{ij} - \tilde{c}_{ij}|$$

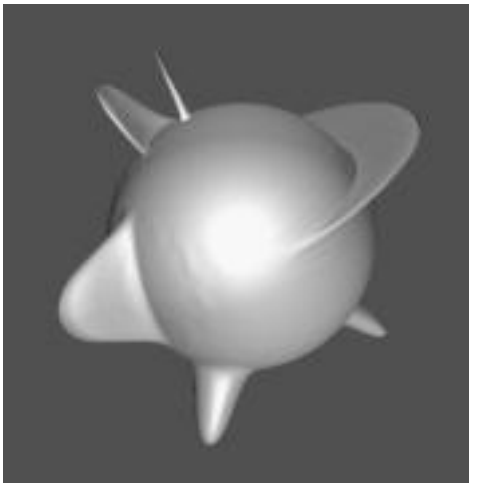
and the *average* ℓ_1 *norm*

$$e_1 := \frac{\sum_{ij} |c_{ij} - \tilde{c}_{ij}|}{m\widetilde{m}},$$

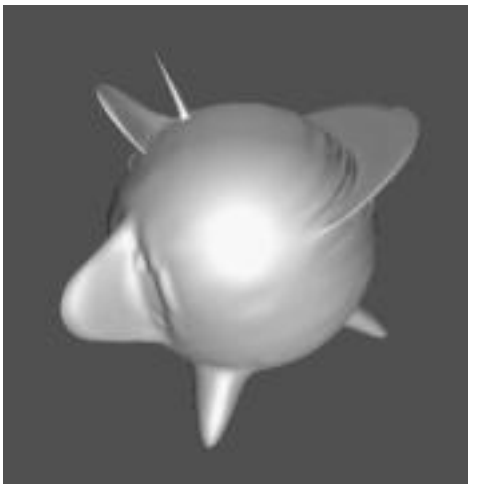
where c_{ij} are the original coefficients and $\tilde{c}_{ij}$ are the reconstructed ones.



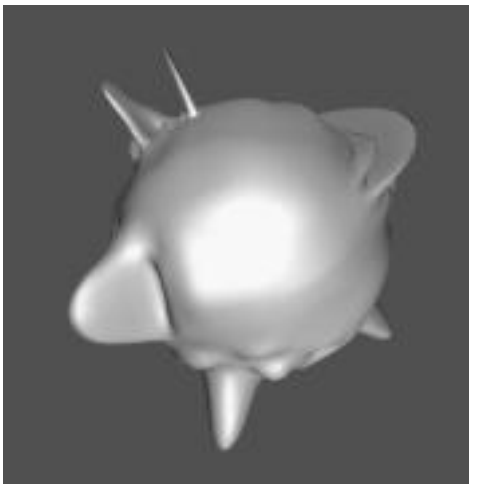
$\varepsilon = 0,\ nco = 591360$



$\varepsilon = .0001,\ nco = 11089$



$\varepsilon = .001,\ nco = 9734$



$\varepsilon = .01,\ nco = 9261$

FIG. 1. *Compressed surfaces for Example* 2.

TABLE 5
Number, nco, of retained coefficients for Example 2 *with* $\varepsilon = .001$.

Step	*nco*
0	591360
1	152832
2	44160
3	17486
4	11209
5	9863
6	9737
7	9734

TABLE 6
Coefficient errors in Example 2 for selected ε.

ε	*nco*	e_∞	e_1
0	591360	0	0
10 (-9)	110466	5.54 (-7)	2.75 (-8)
10 (-8)	74332	6.93 (-6)	2.34 (-7)
10 (-7)	44967	5.16 (-5)	1.80 (-6)
10 (-6)	25409	3.69 (-4)	1.38 (-5)
10 (-5)	14992	3.07 (-3)	7.93 (-5)
10 (-4)	11089	1.16 (-2)	4.56 (-4)
10 (-3)	9734	3.92 (-2)	2.62 (-3)
10 (-2)	9261	1.80 (-1)	1.49 (-2)
10 (-1)	9192	6.22 (-1)	3.93 (-2)

TABLE 7
Example 2 with $\varepsilon = .0001$ using T^0 and T^{-1} thresholding.

Step	*nco*
0	591360
1	150528
2	40846
3	13945
4	7812
5	6728
6	6704
7	6703

Step	*nco*
0	591360
1	148224
2	37390
3	9929
4	3521
5	2316
6	2278
7	2277

The surfaces corresponding to the values $\varepsilon = 0, 10^{-4}, 10^{-3}, 10^{-2}$ are shown in Figure 1. At $\varepsilon = .0001$ we get near perfect looking reconstruction, while at $\varepsilon = .001$ the major features are reproduced with only small wiggles in the surface. At $\varepsilon = .01$ we have somewhat larger errors in the surface. □

We now present analogous results for Example 2 using the modified thresholding procedures of section 8. Table 7 shows the numbers of coefficients remaining after thresholding with $\varepsilon = .0001$ using the strategies T^0 and T^{-1}. In this case the original 591,360 coefficients are reduced to 6703 and 2277, respectively, after seven decomposition steps. This corresponds to compression ratios of 88.22 and 259.71, respectively.

To illustrate the accuracy of the modified thresholding procedures, in Table 8 we list the maximum and average ℓ_1 errors in the coefficient vector of Example 2 after reconstruction using the thresholding strategy T^{-1}. As can be seen by comparing with Table 6, the accuracies are essentially the same as with restricted thresholding, but with much higher compression ratios. The errors corresponding to T^0 thresholding are very similar.

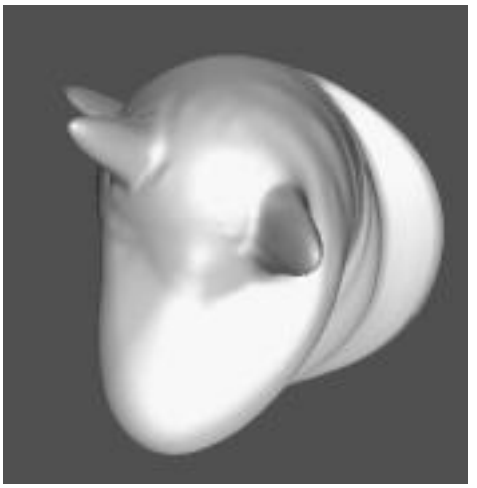
The reconstructed surfaces corresponding to T^0 and T^{-1} thresholding look almost identical to those shown in Figure 1, with only small disturbances at the poles for larger values of ε. To get an idea of what such disturbances look like, in Figure 2 we show polar views of the reconstructed surface for Example 2 in the case $\varepsilon = .001$. The results for T^0 and T^{-1} are shown on the left and right, respectively. The pole is located midway between the "ears" and is virtually invisible for T^0 thresholding.

10. Remarks.

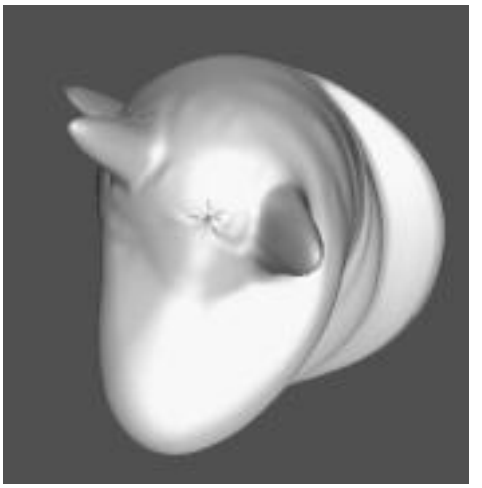
Remark 10.1. For convenience, we have elected to use orthogonal wavelets in both the polynomial and the trigonometric spline spaces. This leads to linear systems with very small band size which can be solved efficiently. The need to solve

TABLE 8
Coefficient errors in Example 2 with T^{-1} thresholding.

ε	nco	e_∞	e_1
0	591360	0	0
10 (-9)	103251	5.54 (-7)	2.77 (-8)
10 (-8)	66720	6.93 (-6)	2.37 (-7)
10 (-7)	37015	5.80 (-5)	1.82 (-6)
10 (-6)	17089	3.69 (-4)	1.42 (-5)
10 (-5)	6391	3.08 (-3)	8.31 (-5)
10 (-4)	2277	1.39 (-2)	4.86 (-4)
10 (-3)	776	5.85 (-2)	3.07 (-3)
10 (-2)	156	2.88 (-1)	1.78 (-2)
10 (-1)	48	6.21 (-1)	4.85 (-2)



$\varepsilon = .001,\ nco = 5276$



$\varepsilon = .001,\ nco = 776$

FIG. 2. *Surfaces with T^0 and T^{-1} thresholding.*

such systems could be avoided altogether with the use of appropriate (but possibly complicated) biorthogonal bases. However, using our bases leads to a decomposition and reconstruction algorithm whose complexity is of the same order as the number of initial coefficients. Thus, no gain in complexity is possible regardless of what other bases one chooses.

Remark 10.2. An alternative way of making sure that tangent plane continuity is maintained at the poles is to decompose the original tensor product function s into two parts s_H and s_P, where

$$s_H := \sum_{i=3}^{m_k-2} \sum_{j=1}^{\widetilde{m}_\ell} c_{ij} \varphi_{k,i} \tilde{\varphi}_{\ell,j}$$

and $s_P := s - s_H$. Then decomposition, thresholding, and reconstruction can be performed on s_H. After adding s_P, the reconstructed spline possesses tangent plane continuity at the poles. Our implementation of this method exhibits essentially the same performance in terms of compression and accuracy as the method described

here, but for higher compression ratios, it produces surfaces that are not quite as visually pleasing near the poles.

Remark 10.3. The method described here can be extended to the case of nonuniform knots in both the θ and ϕ variables. This can be advantageous when the surface is given on nonuniform knots to start with. There is some computational cost in using nonuniform knots, however, since the various matrices appearing in the decomposition and reconstruction processes can no longer be precomputed and stored.

Remark 10.4. In section 4 we have presented the details of the tensor-product decomposition and reconstruction algorithms assuming that the initial function $f_{k,\ell}$ lies in the space $\mathcal{V}_k \times \widetilde{\mathcal{V}}_\ell$, with k and ℓ not necessarily the same. Since these spaces can always be reindexed, this is not strictly necessary in the abstract setting, but was convenient for our application, where there is a natural indexing for our spaces.

Remark 10.5. In computing the coefficients needed in sections 5 and 6, we found it convenient to use Mathematica.

Remark 10.6. There are several methods for computing approximations of the form (1.1). An explicit quasi-interpolation method using data on a regular grid (along with derivatives at the north and south poles) can be found in [13]. The same paper also describes a two-stage method that can be used to interpolate scattered data and a least squares method that can be used to fit noisy data. A general theory of quasi-interpolation operators based on trigonometric splines can be found in [9].

Remark 10.7. A closed, bounded, connected set U in $\mathbb{R}^3$ which is topologically equivalent to a sphere is called a *sphere-like surface*. This means that there exists a one-to-one mapping of U onto the unit sphere S. Moreover, there exists a point O inside the volume surrounded by U, such that every point on the surface U can be seen from O. Such surfaces are also called *star-like*. For applications, we can focus on the class of sphere-like surfaces of the form

$$U = \{v\rho(v) : \ v \in S\},$$

where ρ is a smooth function defined on S. Then each function f defined on U is just the composition $f(\cdot) = g(\rho(\cdot))$ with ρ of a function g defined on S.

Remark 10.8. As indicated in [11], compression methods on the sphere can be adapted to the problem of creating multiresolution representations of *bidirectional reflection distribution functions* (BRDFs), although the basic domain for such functions is actually a hemisphere. We are currently exploring the use of our method for this purpose.

Remark 10.9. It is well known that the polynomial B-splines are stable. In particular, for quadratic B-splines (φ_i) with general knots

$$\frac{1}{3}\|c\|_\infty \le \|\sum c_i\varphi_i\|_{L_\infty} \le \|c\|_\infty$$

for all coefficient vectors c. The same bounds hold for trigonometric splines since the linear functionals

$$\lambda_i f := \left[-f(\tilde{x}_{i+1}) + 2(1+\sigma_i)f\Big(\frac{\tilde{x}_{i+1}+\tilde{x}_{i+2}}{2}\Big) - f(\tilde{x}_{i+2})\right]/2$$

introduced in [13] are dual to the $\tilde{\varphi}_i$, i.e.,

$$\lambda_i\tilde{\varphi}_j = \delta_{ij}, \qquad i,j = 1,\dots,\widetilde{m},$$

where $\sigma_i := \cos((\tilde{x}_{i+2})-\tilde{x}_{i+1})/2)$. Analogous stability results hold for general p-norms with appropriately normalized B-splines.

REFERENCES

[1] C. K. Chui, *An Introduction to Wavelets*, Academic Press, Boston, 1992.

[2] S. Dahlke, W. Dahmen, I. Weinreich, and E. Schmitt, *Multiresolution analysis and wavelets on S^2 and S^3*, Numer. Funct. Anal. Optim., 16 (1995), pp. 19–41.

[3] P. Dierckx, *Algorithms for smoothing data on the sphere with tensor product splines*, Computing, 32 (1984), pp. 319–342.

[4] W. Freeden and U. Windheuser, *Spherical wavelet transform and its discretization*, Adv. Comput. Math., 5 (1996), pp. 51–94.

[5] J. Goettelmann, *Construction of Splines and Wavelets on the Sphere and Numerical Solutions to the Shallow Water Equations of Global Atmospheric Dynamics*, dissertation, Johannes Gutenberg Univ., Mainz, Germany, 1998.

[6] M. Lounsbery, T. D. DeRose, and J. Warren, *Multiresolution analysis for surfaces of arbitrary topological type*, ACM Trans. Graphics, 16 (1997), pp. 34–73.

[7] T. Lyche and K. Mørken, *Spline-wavelets of minimal support*, in Numerical Methods in Approximation Theory, D. Braess and L. L. Schumaker, eds., Birkhäuser, Basel, 1992, pp. 177–194.

[8] T. Lyche and L. Schumaker, *L-spline wavelets*, in Wavelets: Theory, Algorithms, and Applications, C. Chui, L. Montefusco, and L. Puccio, eds., Academic Press, New York, 1994, pp. 197–212.

[9] T. Lyche, L. Schumaker, and S. Stanley, *Quasi-interpolants based on trigonometric splines*, J. Approx. Theory, 95 (1998), pp. 280–309.

[10] D. Potts and M. Tasche, *Interpolatory wavelets on the sphere*, in Approximation Theory VIII, Vol. 2: Wavelets, C. K. Chui and L. L. Schumaker, eds., World Scientific, Singapore, 1995, pp. 335–342.

[11] P. Schröder and W. Sweldens, *Spherical wavelets: Efficiently representing functions on the sphere*, in Computer Graphics Proceedings, Annual Conference Series, 1995, ACM SIGGRAPH, pp. 161–172.

[12] L. L. Schumaker, *Spline Functions: Basic Theory*, Wiley, New York, 1981.

[13] L. L. Schumaker and C. Traas, *Fitting scattered data on spherelike surfaces using tensor products of trigonometric and polynomial splines*, Numer. Math., 60 (1991), 133–144.

[14] E. Stollnitz, T. D. DeRose, and D. Salesin, *Wavelets for Computer Graphics*, Morgan Kaufmann, San Francisco, 1996.

[15] I. Weinreich, *Biorthogonale Wavelets auf der Sphäre*, dissertation, Tech. Univ. Aachen, Germany, 1997.

SIAM J. SCI. COMPUT.
Vol. 22, No. 2, pp. 747–766
© 2000 Society for Industrial and Applied Mathematics

ITERATIVE METHODS FOR NEARLY SINGULAR LINEAR SYSTEMS*

WILLIAM W. HAGER†

Abstract. Iterative methods are developed and studied for near-singular linear systems $\mathbf{Cx} = \mathbf{b}$. Our approach, called the transformed minimal residual algorithm (TMRES), is derived from any convergent iterative scheme $\mathbf{Sx}_{k+1} = \mathbf{Tx}_k + \mathbf{b}$ associated with a splitting $\mathbf{C} = \mathbf{S} - \mathbf{T}$. In each step of TMRES, the transformed residual $\mathbf{S}^{-1}(\mathbf{b} - \mathbf{Cx})$ is minimized over a Krylov space generated by $\mathbf{S}^{-1}\mathbf{T}$. The original iterative scheme typically converges slowly when $\mathbf{C}$ is nearly singular, while a Krylov space generated by $\mathbf{S}^{-1}\mathbf{T}$ often contains a much better approximation to a solution. TMRES is algebraically equivalent to the generalized minimal residual algorithm (GMRES) preconditioned by $\mathbf{S}^{-1}$, although there are numerical differences since a different matrix $\mathbf{S}^{-1}\mathbf{C}$ is used to generate the Krylov space in preconditioned GMRES. Special attention is given to sparsity and convergence issues related to linear systems of the form $(\mathbf{AA}^{\mathsf{T}} + \sigma\mathbf{I})\mathbf{x} = \mathbf{b}$, where $\sigma \geq 0$.

Key words. singular linear system, ill-conditioned system, Krylov space, matrix splitting, preconditioning, generalized minimal residual, successive overrelaxation, Gauss–Seidel, conjugate gradients, linear programming, sparse matrices

AMS subject classifications. 65F10, 65F50, 65Y20

PII. S106482759834634X

1. Introduction.

Iterative methods are developed and analyzed for singular or near-singular linear systems $\mathbf{Cx} = \mathbf{b}$, where $\mathbf{C}$ is an $m \times m$ matrix, possibly nonsymmetric. In the generalized minimal residual algorithm (GMRES) of Saad and Schultz [43], the residual $\mathbf{r}(\mathbf{x}) = \mathbf{b} - \mathbf{Cx}$ is minimized over a Krylov space generated by the matrix $\mathbf{C}$. Our transformed minimal residual approach (TMRES) is based on a splitting $\mathbf{C} = \mathbf{S} - \mathbf{T}$, where $\mathbf{S}$ is nonsingular and the spectral radius of $\mathbf{S}^{-1}\mathbf{T}$ is less than one. The transformed residual $\mathbf{S}^{-1}\mathbf{r}(\mathbf{x})$ is minimized over $\mathbf{x} = \mathbf{x}_0 + \mathbf{z}$ with $\mathbf{z}$ in the Krylov space

$$\mathcal{K}(\mathbf{S}^{-1}\mathbf{T}, \mathbf{g}, k) = \text{span}\{\mathbf{g}, (\mathbf{S}^{-1}\mathbf{T})\mathbf{g}, (\mathbf{S}^{-1}\mathbf{T})^2\mathbf{g}, \ldots, (\mathbf{S}^{-1}\mathbf{T})^{k-1}\mathbf{g}\}$$

generated by $\mathbf{S}^{-1}\mathbf{T}$ starting from the vector $\mathbf{g} = \mathbf{S}^{-1}(\mathbf{b} - \mathbf{Cx}_0)$. Even though the associated iterative scheme $\mathbf{Sx}_{k+1} = \mathbf{Tx}_k + \mathbf{b}$ converges slowly when $\mathbf{C}$ is nearly singular, we observe that $\mathcal{K}(\mathbf{S}^{-1}\mathbf{T}, \mathbf{g}, k)$ often yields a good approximation to a solution of $\mathbf{Cx} = \mathbf{b}$ for relatively small k. Theoretically, the Krylov space generated by $\mathbf{S}^{-1}\mathbf{T}$ is the same as that generated by the GMRES preconditioned matrix $\mathbf{S}^{-1}\mathbf{C}$. Numerically, these spaces differ since the matrices $\mathbf{S}^{-1}\mathbf{T}$ and $\mathbf{S}^{-1}\mathbf{C}$ are different.

The splittings that we consider include successive overrelaxation (SOR), damped Jacobi (see Hageman and Young [22]), and a new splitting applicable to situations where the columns in a matrix are selected from the columns of a larger matrix. Some early work on preconditioned conjugate gradient methods generated by splittings was developed by Concus, Golub, and O'Leary [12, 13]. Work leading up to GMRES includes that of Paige and Saunders [40], who developed a minimum residual algorithm MINRES for symmetric systems. Other related work includes [31] and [50]. In [18]

*Received by the editors October 26, 1998; accepted for publication (in revised form) January 3, 2000; published electronically August 24, 2000. This work was supported by the National Science Foundation.

http://www.siam.org/journals/sisc/22-2/34634.html

†Department of Mathematics, University of Florida, Gainesville, FL 32611 (hager@math.ufl.edu, http://www.math.ufl.edu/∼hager).

Greenbaum examines various preconditioners for the conjugate gradient method in the context of partial differential equations. In [7] Brown and Walker examine the convergence (or lack of it) for GMRES applied to singular or near-singular matrices. The use of deflation with GMRES is studied in [37].

In [4] Baglama et al. develop preconditioned restarted GMRES algorithms in which the preconditioner is generated by Sorensen's implicitly restarted Arnoldi method [46]. Their approach approximates an invariant subspace of the matrix associated with eigenvalues close to the origin. This subspace is used in a preconditioner that moves these small eigenvalues to one, leading to more rapid convergence in the GMRES algorithm. In this paper, we observe that for any convergent splitting $\mathbf{C} = \mathbf{S} - \mathbf{T}$, the eigenvalues of $\mathbf{C}$ close to the origin are often associated with eigenvalues of $\mathbf{S}^{-1}\mathbf{T}$ of nearly largest magnitude; as a result, a good approximation to the solution of a nearly singular system $\mathbf{Cx} = \mathbf{b}$ is often obtained from the Krylov space $\mathcal{K}(\mathbf{S}^{-1}\mathbf{T}, \mathbf{g}, k)$ for relatively small k.

One possible application of the methods developed in this paper is to quadratic programming with a sphere constraint. Problems of this form must be solved in each iteration of the trust region algorithm [8, 10, 16, 38, 42]. The first-order optimality system for these quadratic programs leads to a linear system whose matrix becomes more singular as we increase either the radius of the sphere or the norm of the linear term in the cost function. Hence, the quadratic programs arising in trust region methods can lead to near-singular linear systems for which TMRES is well suited. Other possible application areas include homotopy continuation methods [2], nonlinear eigenvalue problems [11], and seismic inversion problems.

Throughout this paper, we illustrate convergence properties in the near-singular setting using a prototype linear system of the form

$$(\mathbf{A}\mathbf{A}^{\mathsf{T}} + \sigma\mathbf{I})\mathbf{x} = \mathbf{b}, \tag{1.1}$$

where $\sigma \geq 0$ and $\mathbf{A}$ is an $m \times n$ real matrix. Systems of this form, with σ a small positive number, are solved in each iteration of the LP dual active set algorithm [23]. In interior point methods for linear programming [33, 51], each iteration involves solving a linear system with matrix of the form $\mathbf{Z}\mathbf{\Sigma}\mathbf{Z}^{\mathsf{T}}$. This system has the form (1.1) when we take $\mathbf{A} = \mathbf{Z}\mathbf{\Sigma}^{1/2}$ and $\sigma = 0$. The test matrices in this paper are obtained from the LP problems in David Gay's Netlib collection (www.netlib.org/lp). These matrices, which are all sparse, can be obtained in a variety of formats through the COAP Software Forum (www.math.ufl.edu/∼coap).

For symmetric linear systems of the form (1.1), the TMRES scheme with SSOR preconditioning and an SSOR preconditioned conjugate gradient scheme [6, 12, 13] exhibit similar convergence when applied to ill-conditioned matrices emanating from linear programming. In theory, these two schemes generate iterates in the same Krylov space, but they differ in the merit function used to select the approximation from the Krylov space.

To illustrate convergence problems that are encountered when solving nearly singular linear systems, we consider a netlib/lp matrix $\mathbf{A}$ for the (small) problem beaconfd ($m = 173$, $n = 295$). The right side $\mathbf{b}$ was randomly generated on the unit sphere[1] in $\mathbf{R}^m$, $\sigma = 0$, and the columns of $\mathbf{A}$ in (1.1) were scaled to be unit vectors. This column scaling is the one we use in the LP dual active set algorithm. In interior

[1] A vector can be randomly generated on the Euclidean unit sphere by first randomly generating its components using a Gaussian distribution with mean 0 then dividing the resulting vector by its length.

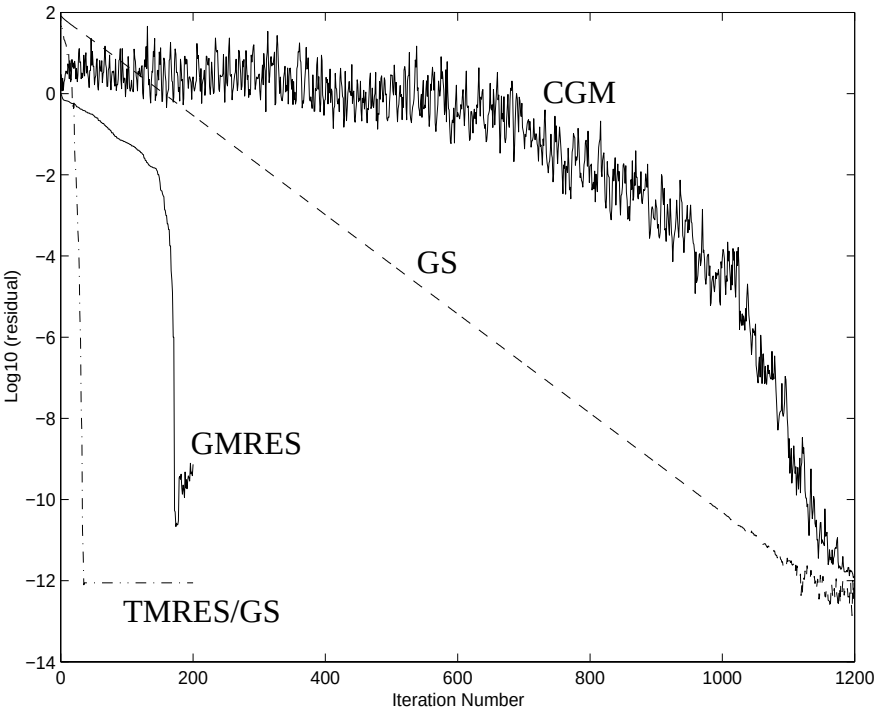


FIG. 1.1. *Convergence for the conjugate-gradient method (CGM), the Gauss–Seidel method (GS), the generalized minimal residual algorithm (GMRES) without restarts, and the transformed minimal residual algorithm (TMRES) obtained from Gauss–Seidel splitting when* $\mathbf{A}$ *in* (1.1) *is the test problem beaconfd.*

point methods, the diagonal scaling matrix $\mathbf{\Sigma}$ becomes increasingly ill-conditioned, and the condition number of the associated $\mathbf{A}\mathbf{A}^{\mathsf{T}}$ could be much larger than that for the normalized $\mathbf{A}$ used in our experiments.

We solve the linear system (1.1) using three different iterative methods: Gauss–Seidel (GS), conjugate gradients (CGM), and GMRES. Although various Lanczos and conjugate gradient type methods [41, 44, 45] have been developed for special versions of (1.1), we do not take into account special structure for (1.1) in our computations; rather, we treat this system as having the form $\mathbf{C}\mathbf{x} = \mathbf{b}$, where $\mathbf{C}$ is an $m \times m$ symmetric, positive definite matrix. All of the schemes, GS, CGM, and GMRES, are guaranteed to converge (see [49, p. 77], [36, Chap. 8], and [43]). Moreover, the conjugate gradient method yields the best approximation to the solution from an associated Krylov space, where the approximation quality is measured using the C-norm defined by $\|\mathbf{x}\|_C = \sqrt{\mathbf{x}^{\mathsf{T}}\mathbf{C}\mathbf{x}}$. GMRES minimizes the residual norm $\|\mathbf{b} - \mathbf{C}\mathbf{x}\|$, where $\|\cdot\|$ is the Euclidean norm, over the same Krylov space. According to the convergence theory, both CGM and GMRES solve $\mathbf{C}\mathbf{x} = \mathbf{b}$ in at most m iterations, assuming exact arithmetic.

With the starting guess $\mathbf{x} = \mathbf{0}$, we plot in Figure 1.1 the iteration number versus the base 10 logarithm of the residual norm for the associated iterate. The computations were done in double precision in Matlab on a Sun workstation. For this small problem, the Gauss–Seidel and conjugate gradient methods converge slowly to the solution, taking nearly 1200 iterations. The generalized minimal residual algorithm

requires about 173 iterations, the number of rows in $\mathbf{A}$. Paige and Saunders' MINRES routine would probably yield results similar to GMRES because $\mathbf{C} = \mathbf{A}\mathbf{A}^\mathsf{T} + \sigma\mathbf{I}$ is symmetric and the two routines are algebraically equivalent in this case. TMRES will be developed in sections 3 and 4. Its convergence behavior shown in Figure 1.1 for a Gauss–Seidel splitting is typical for many netlib/lp test problems.

With GMRES and TMRES, restarts may be essential to reduce the storage requirements when the matrix is nonsymmetric. In this paper, we do not analyze restarts.

2. Convergence properties. Let us study more closely the convergence behavior illustrated in Figure 1.1. The matrix associated with the test problem beaconfd, like many of the problems in netlib/lp, has some rows that are nearly linear combinations of other rows. The 10 smallest and largest eigenvalues of $\mathbf{A}\mathbf{A}^\mathsf{T}$ are the following:

0.00000177	2.8954
0.00000415	3.5490
0.00001338	4.9565
0.00002115	5.1235
0.00002367	6.5875
0.00004273	7.1127
0.00004399	7.5612
0.00004847	10.9200
0.00005424	43.9156
0.00006267	63.2578

The matrix $\mathbf{A}\mathbf{A}^\mathsf{T}$ is nearly singular in the sense that the ratio between the largest and the smallest eigenvalue of $\mathbf{A}\mathbf{A}^\mathsf{T}$ is of order 10^7, much larger than 1.

Let $\mathbf{C} = \mathbf{L} + \mathbf{U}$ be symmetric and positive definite, where

$$l_{ij} = c_{ij} \text{ and } u_{ij} = 0 \text{ for } i \geq j, \quad l_{ij} = 0 \text{ and } u_{ij} = c_{ij} \text{ for } i < j. \tag{2.1}$$

The Gauss–Seidel method for solving $\mathbf{C}\mathbf{x} = \mathbf{b}$ is given by the iteration

$$\mathbf{x}_{k+1} = -\mathbf{L}^{-1}(\mathbf{U}\mathbf{x}_k - \mathbf{b}).$$

If $\mathbf{x}_*$ is the exact solution to $\mathbf{C}\mathbf{x} = \mathbf{b}$, then the error $\mathbf{e}_k = \mathbf{x}_k - \mathbf{x}_*$ satisfies the recurrence $\mathbf{e}_{k+1} = \mathbf{M}\mathbf{e}_k$, where $\mathbf{M} = -\mathbf{L}^{-1}\mathbf{U}$. When $\mathbf{C}$ is positive definite, the spectral radius of $\mathbf{M}$, denoted $\rho(\mathbf{M})$, is strictly less than 1 [49, p. 77]. The spectral radius measures the convergence speed in the sense that the magnitude of error components associated with eigenvectors whose eigenvalue magnitudes are equal to the spectral radius is multiplied by the spectral radius in each iteration.

If $\mathbf{C}$ is singular and $\mathbf{z}$ is a nonzero vector such that $\mathbf{C}\mathbf{z} = \mathbf{0}$, then the relation $(\mathbf{L} + \mathbf{U})\mathbf{z} = \mathbf{0}$ implies that $\mathbf{z} = -\mathbf{L}^{-1}\mathbf{U}\mathbf{z}$, or $\mathbf{z} = \mathbf{M}\mathbf{z}$. Hence, $\mathbf{z}$ is an eigenvector of $\mathbf{M}$ corresponding to the eigenvalue 1, and the spectral radius of $\mathbf{M}$ is at least 1. Consequently, the error $\mathbf{e}_k$ in the Gauss–Seidel iteration may not tend to zero. If $\mathbf{C}$ is nearly singular, then the distance between the eigenvalues of $\mathbf{M}$ and 1 can be estimated using Gerschgorin's theorem.

PROPOSITION 2.1. *Let $\mathbf{C}_0$ be a square singular matrix, and consider the perturbed matrix $\mathbf{C} = \mathbf{C}_0 + \tau\mathbf{I}$. Let $\mathbf{S}$ be any nonsingular matrix for which $\mathbf{S}^{-1}\mathbf{C}_0$ is diagonalizable: $\mathbf{S}^{-1}\mathbf{C}_0 = \mathbf{F}\mathbf{\Phi}\mathbf{F}^{-1}$, where $\mathbf{\Phi}$ is a diagonal matrix containing the eigenvalues. If $\mathbf{T}$ is chosen so that $\mathbf{C} = \mathbf{S} - \mathbf{T}$ (i.e., $\mathbf{T} = \mathbf{S} - \mathbf{C}_0 - \tau\mathbf{I}$), then for τ sufficiently close*

to zero, the eigenvalues $\mu_1, \mu_2, \ldots, \mu_n$ *of* $\mathbf{S}^{-1}\mathbf{T}$ *satisfy*

$$\min_i |\mu_i - 1| \leq |\tau| \|\mathbf{F}\|_1 \|\mathbf{F}^{-1}\|_1 \|\mathbf{S}^{-1}\|_1,$$

where $\|\cdot\|_1$ *is the matrix* 1*-norm (largest absolute column sum).*

Proof. Observe that

$$\mathbf{F}^{-1}(\mathbf{S}^{-1}\mathbf{T})\mathbf{F} = \mathbf{F}^{-1}(\mathbf{I} - \mathbf{S}^{-1}\mathbf{C}_0 - \tau\mathbf{S}^{-1})\mathbf{F} = \mathbf{I} - \mathbf{\Phi} - \tau\mathbf{F}^{-1}\mathbf{S}^{-1}\mathbf{F}. \tag{2.2}$$

Since $\mathbf{C}_0$ is singular, one of the diagonal elements of $\mathbf{\Phi}$ is zero. Hence, one of the diagonal elements of $\mathbf{I} - \mathbf{\Phi}$ is 1. Let δ be the minimum absolute difference between distinct diagonal elements of $\mathbf{\Phi}$ and choose τ close enough to zero that

$$|\tau| \|\mathbf{F}\|_1 \|\mathbf{F}^{-1}\|_1 \|\mathbf{S}^{-1}\|_1 < \delta.$$

By Gerschgorin's theorem and the fact that $\mathbf{S}^{-1}\mathbf{T}$ and $\mathbf{F}^{-1}(\mathbf{S}^{-1}\mathbf{T})\mathbf{F}$ are similar, it follows from (2.2) that $\mathbf{S}^{-1}\mathbf{T}$ has an eigenvalue in the sphere in the complex plane with center 1 and radius $|\tau| \|\mathbf{F}\|_1 \|\mathbf{F}^{-1}\|_1 \|\mathbf{S}^{-1}\|_1$. This completes the proof. ∎

Loosely speaking, eigenvalues of $\mathbf{C}$ near 0 correspond to eigenvalues of $\mathbf{M} = \mathbf{S}^{-1}\mathbf{T}$ near 1 and to slow convergence of the Gauss–Seidel method.

A number of papers [32, 35, 47] have appeared in the literature relating the spectrum of the conjugate gradient method to its convergence. More recent work is surveyed in [1]. Here we focus on the effect of near singularity on convergence. The following result is the key to understanding the behavior for CGM depicted in Figure 1.1.

PROPOSITION 2.2. *If* $\mathbf{C}$ *is a symmetric matrix, then as* σ *tends to zero, we have*

$$\sigma(\sigma\mathbf{I} + \mathbf{C})^{-1} = \mathbf{P} + O(\sigma),$$

where $\mathbf{P}$ *denotes the orthogonal projection into the null space of* $\mathbf{C}$ *and* $O(\sigma)$ *denotes a term that can be bounded in magnitude by a constant times* $|\sigma|$.

Proof. Let $\mathbf{C} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^{\mathsf{T}}$ denote the diagonalization of $\mathbf{C}$, where $\mathbf{Q}$ is the orthogonal matrix of eigenvectors and $\mathbf{\Lambda}$ is a diagonal matrix whose diagonal elements $\lambda_1, \lambda_2, \ldots, \lambda_m$ are the eigenvalues. We assume that the eigenvalues are ordered so that

$$|\lambda_1| \geq |\lambda_2| \geq \cdots \geq |\lambda_k| > 0 \quad \text{and } \lambda_i = 0 \text{ for } i > k.$$

Hence,

$$\sigma(\sigma\mathbf{I} + \mathbf{C})^{-1} = \sigma\mathbf{Q}(\mathbf{\Lambda} + \sigma\mathbf{I})^{-1}\mathbf{Q}^{\mathsf{T}} \tag{2.3}$$

and

$$\sigma(\mathbf{\Lambda} + \sigma\mathbf{I})^{-1} = \mathbf{J} + O(\sigma), \tag{2.4}$$

where

$$\mathbf{J} = \begin{pmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{pmatrix}.$$

Above, the lower right $\mathbf{I}$ is $(m-k) \times (m-k)$, and $\mathbf{0}$ stands for a block of zeros. Since $\mathbf{Q}\mathbf{J}\mathbf{Q}^{\mathsf{T}} = \mathbf{P}$, (2.3) and (2.4) complete the proof. ∎

The CGM applied to the linear system $\mathbf{Cx} = \mathbf{b}$, where $\mathbf{C}$ is symmetric and positive definite, is the following [25, p. 340]: $\mathbf{d}_0 = \mathbf{r}_0 = \mathbf{b} - \mathbf{Cx}_0$ and for $k \geq 0$,

$$\begin{array}{ll} \mathbf{p}_k & \leftarrow \mathbf{Cd}_k, \\ \mathbf{x}_{k+1} & \leftarrow \mathbf{x}_k + \alpha_k \mathbf{d}_k, \quad \text{where } \alpha_k = \mathbf{r}_k^{\mathsf{T}}\mathbf{r}_k / \mathbf{d}_k^{\mathsf{T}}\mathbf{p}_k, \\ \mathbf{r}_{k+1} & \leftarrow \mathbf{r}_k - \alpha_k \mathbf{p}_k, \\ \mathbf{d}_{k+1} & \leftarrow \mathbf{r}_{k+1} + \beta_k \mathbf{d}_k, \quad \text{where } \beta_k = \mathbf{r}_{k+1}^{\mathsf{T}}\mathbf{r}_{k+1} / \mathbf{r}_k^{\mathsf{T}}\mathbf{r}_k. \end{array}$$

In each step of the conjugate gradient method, the current direction $\mathbf{d}_k$ is multiplied by $\mathbf{C}$ in the process of obtaining the new direction $\mathbf{d}_{k+1}$. By induction, $\mathbf{x}_k = \mathbf{x}_0 + \mathbf{z}_k$, where $\mathbf{z}_k$ lies in the Krylov space $\mathcal{K}(\mathbf{C}, \mathbf{r}_0, k)$.

Let $\mathbf{x}_* = \mathbf{C}^{-1}\mathbf{b}$ denote the solution to $\mathbf{Cx} = \mathbf{b}$. The conjugate gradient method is designed to minimize $\|\mathbf{x}_k - \mathbf{x}_*\|_C$ over $\mathbf{x}_k = \mathbf{x}_0 + \mathbf{z}_k$ with $\mathbf{z}_k \in \mathcal{K}(\mathbf{C}, \mathbf{r}_0, k)$. Observe that

$$\|\mathbf{x} - \mathbf{x}_*\|_C = \sqrt{(\mathbf{x} - \mathbf{x}_*)\mathbf{C}(\mathbf{x} - \mathbf{x}_*)} = \sqrt{(\mathbf{Cx} - \mathbf{b})^{\mathsf{T}}\mathbf{C}^{-1}(\mathbf{Cx} - \mathbf{b})} = \sqrt{\mathbf{r}(\mathbf{x})^{\mathsf{T}}\mathbf{C}^{-1}\mathbf{r}(\mathbf{x})},$$

where $\mathbf{r}(\mathbf{x}) = \mathbf{b} - \mathbf{Cx}$. If $(\lambda_i, \mathbf{q}_i)$, $i = 1, \ldots, m$, are orthonormal eigenpairs of $\mathbf{C}$, then

$$\|\mathbf{x} - \mathbf{x}_*\|_C^2 = \sum_{i=1}^{m} c_i(\mathbf{x})^2 / \lambda_i, \quad \text{where } c_i(\mathbf{x}) = \mathbf{r}(\mathbf{x})^{\mathsf{T}}\mathbf{q}_i. \tag{2.5}$$

Since the reciprocal of the eigenvalues appears in this expression, the components c_i of the residual $\mathbf{r}$ associated with the smallest eigenvalues are amplified the most in (2.5).

Suppose that we apply CGM to a linear system of the form (1.1), where the rows of $\mathbf{A}$ are linearly dependent and σ is small. By Proposition 2.2, the solution $\mathbf{x} \approx \mathbf{Pb}/\sigma$, the projection of $\mathbf{b}$ into the null space of $\mathbf{A}^{\mathsf{T}}$, divided by σ. On the other hand, the kth conjugate gradient iterate lies in the Krylov space $\mathcal{K}(\mathbf{C}, \mathbf{b}, k)$ if $\mathbf{x}_0 = \mathbf{0}$. The vectors forming this space are all contained in the range of $\mathbf{C}$, and if σ is small, then each of these vectors is nearly in the range of $\mathbf{A}$. We saw in (2.5) that CGM attaches the greatest weight to those components of the error associated with the smallest eigenvalues. Hence, the conjugate gradient method is using vectors that nearly lie in the range of $\mathbf{A}$ to approximate solution components that are nearly orthogonal to the range. In theory, CGM is guaranteed to reach the solution in a finite number of steps. This convergence must involve the subtraction of nearly equal numbers followed by the division of numbers that are nearly zero.

For the example depicted in Figure 1.1 with $\sigma = 0$, the effect is very similar. The rows of $\mathbf{A}$ are nearly dependent, the solution nearly lies in the space spanned by eigenvectors associated with the smallest eigenvalues of $\mathbf{AA}^{\mathsf{T}}$, while the conjugate gradient iterates nearly lie in the space spanned by eigenvectors associated with the largest eigenvalues of $\mathbf{AA}^{\mathsf{T}}$. For a nice survey of results concerning the convergence of the conjugate gradient method in finite precision arithmetic, see the paper [20] of Greenbaum and Strakoš. There they observe that the finite precision conjugate gradient method behaves similarly to the exact algorithm applied to a matrix with nearby eigenvalues.

In the generalized minimal residual algorithm applied to $\mathbf{Cx} = \mathbf{b}$, the kth approximation $\mathbf{x}_k$ to the solution is obtained by solving the problem

$$\min \ \|\mathbf{r}(\mathbf{x})\| \quad \text{subject to } \mathbf{x} = \mathbf{x}_0 + \mathbf{z}, \quad \mathbf{z} \in \mathcal{K}(\mathbf{C}, \mathbf{r}_0, k). \tag{2.6}$$

Thus when $\mathbf{C}$ is symmetric, the GMRES iterates are contained in the same Krylov space used by the conjugate gradient method. Unlike CGM, GMRES does not weight the components of the residual in (2.6). As k increases, the components of the vector $\mathbf{C}^k\mathbf{r}_0$ associated with the large eigenvalues of $\mathbf{C}$ become much larger in magnitude than the components associated with the small eigenvalues. Hence, we expect that GMRES approximates the solution components associated with the largest eigenvalues first. As the number of iterations approaches the dimension of the matrix, GMRES generates vectors orthogonal to the eigenvectors associated with the large eigenvalues. These orthogonal vectors correspond to the space associated with the eigenvectors of the smallest eigenvalues. Observe in Figure 1.1 that when the approximating space approaches the space corresponding to the smallest eigenvalues, the GMRES error quickly decreases. In order to improve on GMRES in this near-singular setting, we will strive to reach this space sooner. Note that for nonsymmetric matrices, the convergence of GMRES depends on both eigenvalues and departure from normality. For example, see [9, 19, 39] for related results and discussion.

3. Krylov, Arnoldi, and TMRES. With CGM or GMRES, the approximate solution lies in a Krylov space, and an orthonormal basis for this space is generated by the Arnoldi (Gram–Schmidt) process. The following algorithm generates such an orthonormal basis $\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k\}$ for the Krylov space $\mathcal{K}(\mathbf{M}, \mathbf{g}, k)$:

ALGORITHM 1 (Arnoldi).

$\mathbf{v}_1 \leftarrow \mathbf{g}/\|\mathbf{g}\|$
for $j = 1 : k-1$
  $\mathbf{s} \leftarrow \mathbf{M}\mathbf{v}_j$
  for $i = 1 : j$
    $h_{ij} \leftarrow \mathbf{s}^{\mathsf{T}}\mathbf{v}_i \quad (= \mathbf{v}_i^{\mathsf{T}}\mathbf{M}\mathbf{v}_j)$
    $\mathbf{s} \leftarrow \mathbf{s} - h_{ij}\mathbf{v}_i$
  end
  $h_{j+1,j} \leftarrow \|\mathbf{s}\|$
  $\mathbf{v}_{j+1} \leftarrow \mathbf{s}/h_{j+1,j}$
end
end Algorithm 1

Obviously, if $h_{k+1,k} = 0$ for some k, the Arnoldi process should stop because

$$\mathcal{K}(\mathbf{M}, \mathbf{g}, j+1) = \mathcal{K}(\mathbf{M}, \mathbf{g}, j) \quad \text{for all } j \geq k. \tag{3.1}$$

CGM or GMRES corresponds to the choice $\mathbf{M} = \mathbf{C}$.

In order to more quickly generate vectors near the space spanned by the eigenvectors of $\mathbf{C}$ associated with the smallest eigenvalues, we consider the matrix $\mathbf{M} = \mathbf{S}^{-1}\mathbf{T}$ associated with a splitting $\mathbf{C} = \mathbf{S} - \mathbf{T}$, where $\mathbf{S}$ is nonsingular. This splitting leads to an iterative method

$$\mathbf{S}\mathbf{x}_{k+1} = \mathbf{T}\mathbf{x}_k + \mathbf{b}, \tag{3.2}$$

which converges to a solution of $\mathbf{C}\mathbf{x} = \mathbf{b}$ for all choices of the initial condition $\mathbf{x}_0$ if and only if $\rho(\mathbf{S}^{-1}\mathbf{T}) < 1$. The terminology "convergent splitting" will mean that $\rho(\mathbf{S}^{-1}\mathbf{T}) < 1$. From the discussion of section 2, we know that the eigenpairs of $\mathbf{C}$ whose eigenvalues are near zero correspond to eigenpairs of $\mathbf{S}^{-1}\mathbf{T}$ whose eigenvalues are near 1. Hence, the eigenvectors of $\mathbf{S}^{-1}\mathbf{T}$ associated with its eigenvalues of largest magnitude include approximations to the eigenvectors of $\mathbf{C}$ associated with its eigenvalues of smallest magnitude. Suppose that $\mathbf{S}^{-1}\mathbf{T}$ has m linearly independent eigenvectors $\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_m$ and associated eigenvalues $\mu_1, \mu_2, \ldots, \mu_m$. If

$\mathbf{g} = c_1\mathbf{f}_1 + c_2\mathbf{f}_2 + \cdots + c_m\mathbf{f}_m$ is the expansion of $\mathbf{g}$ in terms of the eigenvectors, then

$$\mathbf{M}^k\mathbf{g} = (\mathbf{S}^{-1}\mathbf{T})^k\mathbf{g} = c_1\mu_1^k\mathbf{f}_1 + c_2\mu_2^k\mathbf{f}_2 + \cdots + c_m\mu_m^k\mathbf{f}_m.$$

It follows that the components of $\mathbf{g}$ associated with the absolute largest eigenvalues of $\mathbf{M}$ are amplified more rapidly than the components of $\mathbf{g}$ associated with the absolute smallest eigenvalues. Loosely speaking, the Krylov space $\mathcal{K}(\mathbf{M}, \mathbf{g}, k)$ converges, as k grows, to those eigenvectors associated with the absolute largest eigenvalues of $\mathbf{M}$ much more quickly than to those eigenvectors associated with the absolute smallest eigenvalues of $\mathbf{M}$. Since those eigenvectors associated with the large eigenvalues of $\mathbf{M}$ correspond to eigenvalues of $\mathbf{C}$ associated with its smallest eigenvalues, we converge to that space associated with the small eigenvalues of $\mathbf{C}$ that is critical in the near-singular setting.

After generating a Krylov space and a basis for it using Algorithm 1, we now return to the linear system $\mathbf{Cx} = \mathbf{b}$. If we were to minimize the norm of the residual $\mathbf{r} = \mathbf{b} - \mathbf{Cx}$ over the space spanned by the basis vectors $\mathbf{v}_i$, $1 \le i \le k$, we would need to multiply each $\mathbf{v}_i$ by $\mathbf{C}$. To circumvent this multiplication, we minimize the transformed residual $\mathbf{t}$ given by

$$\mathbf{t}(\mathbf{x}) = \mathbf{S}^{-1}\mathbf{r}(\mathbf{x}) = \mathbf{S}^{-1}(\mathbf{b} - \mathbf{Cx}) = \mathbf{S}^{-1}\mathbf{b} - (\mathbf{I} - \mathbf{M})\mathbf{x}.$$

Now when $\mathbf{x}$ is expanded in the basis vectors $\mathbf{v}_i$ and we minimize the norm of $\mathbf{t}$, we need to compute the product $\mathbf{Mv}_i$, which is available from Algorithm 1.

At step k, the TMRES approximation $\mathbf{x}_k$ to $\mathbf{x} = \mathbf{C}^{-1}\mathbf{b}$ is the solution to the transformed problem

$$\min \|\mathbf{t}(\mathbf{x})\| \quad \text{subject to } \mathbf{x} = \mathbf{z} + \mathbf{x}_0, \quad \mathbf{z} \in \mathcal{K}(\mathbf{M}, \mathbf{g}, k). \tag{3.3}$$

Making the special choice $\mathbf{g} = \mathbf{S}^{-1}(\mathbf{b} - \mathbf{Cx}_0)$, and substituting $\mathbf{x} = \mathbf{z} + \mathbf{x}_0$, we obtain the problem

$$\min \|\mathbf{g} - (\mathbf{I} - \mathbf{M})\mathbf{z}\| \quad \text{subject to} \quad \mathbf{z} \in \mathcal{K}(\mathbf{M}, \mathbf{g}, k). \tag{3.4}$$

If $\mathbf{V}_k$ denotes the matrix whose columns are the vectors $\mathbf{v}_1$, $\mathbf{v}_2$, ... , $\mathbf{v}_k$ generated by the Arnoldi process, then (3.4) reduces to the following equivalent problem:

$$\min_{\mathbf{y}} \|\mathbf{g} - (\mathbf{I} - \mathbf{M})\mathbf{V}_k\mathbf{y}\|. \tag{3.5}$$

The solution to the least-squares problem (3.5) can be obtained by a process similar to that described in [43]. If $\mathbf{H}$ is the $(k+1) \times k$ upper Hessenberg matrix whose elements are given by the Arnoldi process, then the following relation holds:

$$\mathbf{MV}_k = \mathbf{V}_{k+1}\mathbf{H}. \tag{3.6}$$

Let $\mathbf{e}$ denote the vector whose first component is $\|\mathbf{g}\|$ and whose remaining components are zero. After substituting $\mathbf{g} = \mathbf{V}_{k+1}\mathbf{e}$ in (3.5) and utilizing (3.6), we obtain the equivalent problem

$$\min_{\mathbf{y}} \|\mathbf{V}_{k+1}\mathbf{e} - \mathbf{V}_k\mathbf{y} + \mathbf{V}_{k+1}\mathbf{Hy}\|. \tag{3.7}$$

Since the columns of $\mathbf{V}_{k+1}$ are orthonormal, we have

$$\begin{aligned}\|\mathbf{V}_{k+1}\mathbf{e} - \mathbf{V}_k\mathbf{y} + \mathbf{V}_{k+1}\mathbf{H}\mathbf{y}\| &= \|\mathbf{V}_{k+1}\mathbf{e} - \mathbf{V}_{k+1}\begin{bmatrix}\mathbf{y}\\0\end{bmatrix} + \mathbf{V}_{k+1}\mathbf{H}\mathbf{y}\| \\ &= \|\mathbf{V}_{k+1}\mathbf{e} + \mathbf{V}_{k+1}\left(\mathbf{H} - \begin{bmatrix}\mathbf{I}\\\mathbf{0}\end{bmatrix}\right)\mathbf{y}\| \\ &= \|\mathbf{e} + \bar{\mathbf{H}}\mathbf{y}\|,\end{aligned}$$

where $\bar{\mathbf{H}}$ is the same as the Hessenberg matrix $\mathbf{H}$ generated by Algorithm 1 except that 1's have been subtracted from the diagonal elements. With this substitution, (3.7) simplifies to

$$\min_{\mathbf{y}} \|\mathbf{e} + \bar{\mathbf{H}}\mathbf{y}\|. \tag{3.8}$$

Finally, the vector $\mathbf{e} + \bar{\mathbf{H}}\mathbf{y}$ is multiplied on the left by a series of Givens rotations reducing $\bar{\mathbf{H}}$ to triangular form $\mathbf{R}$ and reducing $\mathbf{e}$ to a vector $\mathbf{f}$. Our approximation to the solution of $\mathbf{C}\mathbf{x} = \mathbf{b}$ is $\mathbf{x}_k = \mathbf{x}_0 + \mathbf{V}_k\mathbf{y}$, where $\mathbf{y}$ is the solution of the triangular system $(\mathbf{R}\mathbf{y})_{1:k} = \mathbf{f}_{1:k}$. The minimum norm in (3.8) is $|f_{k+1}|$.

In the following statement of TMRES, we overwrite $\mathbf{H}$ and $\mathbf{e}$ with $\mathbf{R}$ and $\mathbf{f}$, respectively. This algorithm is basically the same as preconditioned GMRES; in GMRES preconditioned by $\mathbf{S}^{-1}$, the multiplication by $\mathbf{S}^{-1}\mathbf{T}$ is replaced by $\mathbf{S}^{-1}\mathbf{C}$ and the diagonal of $\mathbf{H}$ is not modified. The function Givens($\mathbf{a}$) below generates a 2×2 rotation matrix $\mathbf{Q}$ with the property that $(\mathbf{Q}\mathbf{a})_2 = 0$.

ALGORITHM 2 (TMRES for $\mathbf{C}\mathbf{x} = \mathbf{b}$).

$\mathbf{C} = \mathbf{S} - \mathbf{T}$, $\mathbf{S}$ nonsingular, $\rho(\mathbf{S}^{-1}\mathbf{T}) \le 1$
$\mathbf{g} \leftarrow \mathbf{S}^{-1}(\mathbf{b} - \mathbf{C}\mathbf{x}_0)$, $\quad \mathbf{v}_1 \leftarrow \mathbf{g}/\|\mathbf{g}\|$, $\quad \mathbf{e} \leftarrow \mathbf{0}$, $\quad e_1 \leftarrow \|\mathbf{g}\|$
for $j = 1, 2 \ldots$ until convergence
  $\mathbf{s} \leftarrow \mathbf{S}^{-1}(\mathbf{T}\mathbf{v}_j)$
  for $i = 1 : j$
    $h_{ij} \leftarrow \mathbf{s}^{\mathsf{T}}\mathbf{v}_i$
    $\mathbf{s} \leftarrow \mathbf{s} - \mathbf{v}_i h_{ij}$
  end
  $h_{j+1,j} \leftarrow \|\mathbf{s}\|$
  $\mathbf{v}_{j+1} \leftarrow \mathbf{s}/h_{j+1,j}$
  $h_{ii} \leftarrow h_{ii} - 1$
  for $i = 1 : j - 1$
    $\mathbf{H}(i : i+1, j) \leftarrow \mathbf{Q}_i\mathbf{H}(i : i+1, j)$
  end
  $\mathbf{Q}_j \leftarrow \text{Givens}(\mathbf{H}(j : j+1, j))$
  $\mathbf{H}(j : j+1, j) \leftarrow \mathbf{Q}_j\mathbf{H}(j : j+1, j)$
  $\mathbf{e}(j : j+1) \leftarrow \mathbf{Q}_j\mathbf{e}(j : j+1)$
end
$\mathbf{x}_j \leftarrow \mathbf{x}_0 - \mathbf{V}_j\left(\mathbf{H}(1 : j, 1 : j)^{-1}\mathbf{e}(1 : j)\right)$
end Algorithm 2

Possible choices of $\mathbf{S}$ with the property that $\rho(\mathbf{S}^{-1}\mathbf{T}) < 1$ are the following: (i) if $\mathbf{C}$ is symmetric and positive definite, then $\mathbf{S} = \mathbf{L}$, where $\mathbf{L}$ is the lower triangular matrix whose lower triangle matches that of $\mathbf{C}$ (Gauss–Seidel choice); (ii) if $\mathbf{C}$ is row diagonally dominant, then $\mathbf{S} = \mathbf{L}$ or $\mathbf{S} = \mathbf{D}$, where $\mathbf{D}$ is the diagonal matrix whose diagonal matches that of $\mathbf{C}$ (Jacobi choice); (iii) if $\mathbf{C} = \mathbf{A}\mathbf{A}^{\mathsf{T}} + \sigma\mathbf{I}$ where the columns of $\mathbf{A}$ come from the columns of a larger matrix $\mathbf{B} = (\mathbf{A}|\mathbf{N})$, then $\mathbf{S} = \mathbf{B}\mathbf{B}^{\mathsf{T}} + \sigma\mathbf{I}$ (see Theorem 6.2 below).

4. Convergence analysis. The convergence of TMRES follows readily from previously established theory. Recall [29, p. 471] that the index of the matrix $\mathbf{M}$ with respect to the zero eigenvalue is the size of the largest singular Jordan block. If $\mathcal{R}(\mathbf{C})$ denotes the range of $\mathbf{C}$, then we have the following result.

THEOREM 4.1. *Let $\mathbf{C}$ be a square matrix with $\mathbf{C} = \mathbf{S} - \mathbf{T}$, where $\mathbf{S}$ is nonsingular, and define $\mathbf{M} = \mathbf{S}^{-1}\mathbf{T}$. Then $\mathbf{Cx} = \mathbf{b}$ has a solution in the Krylov space $\mathcal{K}(\mathbf{M}, \mathbf{S}^{-1}\mathbf{b}, k)$ for some k if and only if $\mathbf{S}^{-1}\mathbf{b} \in \mathcal{R}((\mathbf{I}-\mathbf{M})^i)$, where i is the index of the zero eigenvalue of $\mathbf{I} - \mathbf{M}$.*

Proof. By [30, Thm. 2] the linear system $(\mathbf{I} - \mathbf{M})\mathbf{x} = \mathbf{g}$ has a solution in the Krylov space $\mathcal{K}(\mathbf{I}-\mathbf{M}, \mathbf{g}, k)$ for k sufficiently large if and only if $\mathbf{g} \in \mathcal{R}((\mathbf{I}-\mathbf{M})^i)$, where i is the index of the zero eigenvalue of $\mathbf{I} - \mathbf{M}$. If P_k denotes the set of polynomials of degree at most k, it can be shown that

$$P_k = \{p(y) : y = 1 - x,\ p \in P_k\}.$$

As a result, we have $\mathcal{K}(\mathbf{M}, \mathbf{g}, k) = \mathcal{K}(\mathbf{I} - \mathbf{M}, \mathbf{g}, k)$. Hence, by [30, Thm. 2] the linear system $(\mathbf{I} - \mathbf{M})\mathbf{x} = \mathbf{g}$ has a solution in the Krylov space $\mathcal{K}(\mathbf{M}, \mathbf{g}, k)$ for k sufficiently large if and only if $\mathbf{g} \in \mathcal{R}((\mathbf{I} - \mathbf{M})^i)$. We take $\mathbf{g} = \mathbf{S}^{-1}\mathbf{b}$ to complete the proof. ∎

If one is not an eigenvalue of $\mathbf{M}$, then the index of the zero eigenvalue of $\mathbf{I} - \mathbf{M}$ is zero and the condition $\mathbf{S}^{-1}\mathbf{b} \in \mathcal{R}(\mathbf{I})$ holds trivially. If one is a nondefective eigenvalue of $\mathbf{M}$, then the index of the zero eigenvalue of $\mathbf{I} - \mathbf{M}$ is one and the condition $\mathbf{S}^{-1}\mathbf{b} \in \mathcal{R}(\mathbf{I} - \mathbf{M})$ is equivalent to $\mathbf{b} \in \mathcal{R}(\mathbf{C})$.

An analysis of the convergence of TMRES in terms of the spectrum of $\mathbf{M}$ can be given along the lines of the analysis for the conjugate gradient method in [48] or [14], and for GMRES in [43]. For example, we have the following result.

THEOREM 4.2. *Suppose that $\mathbf{C} = \mathbf{S} - \mathbf{T}$, where $\mathbf{S}$ is invertible, and that there exists a nonsingular matrix $\mathbf{F}$ such that $\mathbf{M} = \mathbf{S}^{-1}\mathbf{T} = \mathbf{F}\mathbf{\Phi}\mathbf{F}^{-1}$, where $\mathbf{\Phi}$ is diagonal. If $\mathbf{x}_k$ denotes a solution to (3.3), and μ_i is the ith diagonal element of $\mathbf{\Phi}$, then we have*

$$\|\mathbf{t}(\mathbf{x}_k)\| \le \|\mathbf{F}\|\ \|\mathbf{F}^{-1}\|\ \|\mathbf{t}(\mathbf{x}_0)\| \min_{\substack{p \in P_k \\ p(0)=1}} \max_i\ |p(1-\mu_i)|.$$

Proof. Simply apply [43, Prop. 4] to the transformed system $(\mathbf{I}-\mathbf{M})\mathbf{z} = \mathbf{g}$, where $\mathbf{g} = \mathbf{t}(\mathbf{x}_0)$. ∎

5. More examples. When $\mathbf{C}$ is symmetric and positive definite, the Gauss–Seidel splitting is convergent and, hence, yields an appropriate TMRES splitting. Figure 1.1 shows the iterates for the TMRES/GS algorithm applied to (1.1) where $\mathbf{A}$ is the matrix from beaconfd and $\sigma = 0$. After about 35 iterations, the norm of the residual has been reduced by 12 orders of magnitude, providing relatively rapid convergence.

Next, we examine the convergence for progressively larger problems from the netlib/lp directory. Figure 5.1 shows the convergence for the system (1.1) where $\mathbf{A}$ is the matrix from fffff800 ($m = 524$, $n = 1028$, $\sigma = 0$). The condition number for $\mathbf{A}\mathbf{A}^\mathsf{T}$ is of order 10^{19}. Also, in Figure 5.1 we show the convergence of preconditioned GMRES with the preconditioner $\mathbf{S} = \mathbf{L}$, the lower triangular part of $\mathbf{C}$. Theoretically, the TMRES and the preconditioned GMRES curves should coincide since the Krylov spaces generated by $\mathbf{M}$ and by $\mathbf{I} - \mathbf{M}$ are identical, algebraically. Note though that when a vector is multiplied by $\mathbf{I} - \mathbf{M}$, some subtraction of nearly equal numbers occurs: If $\mathbf{f}$ is an eigenvector of $\mathbf{M}$ whose associated eigenvalue μ is near one, then

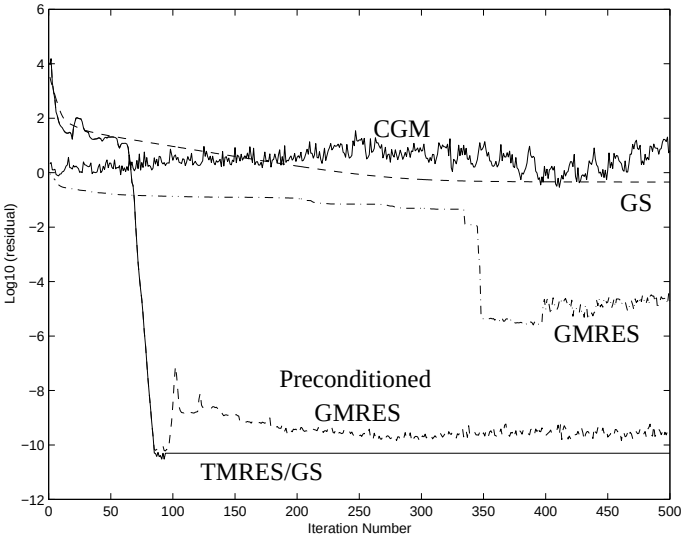


FIG. 5.1. *Convergence for the test problem* fffff800, $\sigma = 0$.

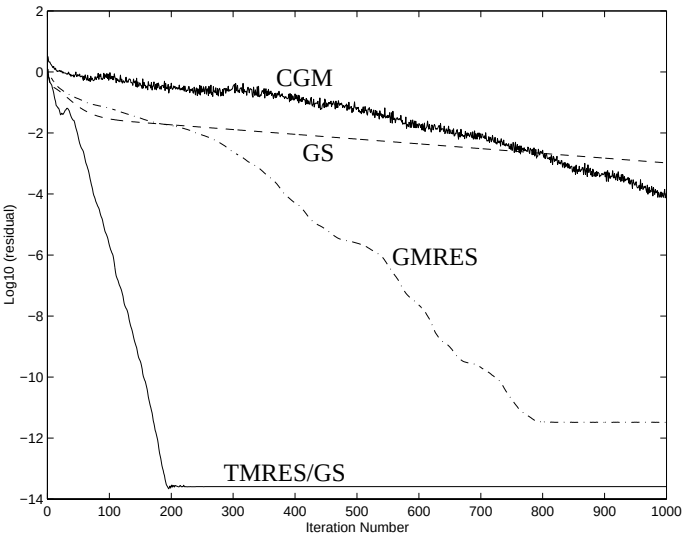


FIG. 5.2. *Convergence for the test problem greenbea, zero rows deleted,* $\sigma = 0$.

$(\mathbf{I} - \mathbf{M})\mathbf{f} = (1 - \mu)\mathbf{f} \approx \mathbf{0}$. As is well known (see [25, sects. 1–4]), the subtraction of nearly equal numbers can produce a large relative error and numerical differences between TMRES and GMRES.

Figures 5.2 and 5.3 show the convergence for greenbea ($m = 2392$, $n = 5598$). Since 3 rows in greenbea are completely zero, we ran two variations of the problem. In the variation of Figure 5.2, the zero rows are deleted, while in the variation of Figure 5.3, the zero rows are included, but $\sigma = 10^{-6}$ to keep the diagonal of $\mathbf{C}$ nonzero. In each case, the TMRES algorithm exhibits an attractive convergence rate. Since the TMRES algorithm minimizes the norm of the transformed residual in

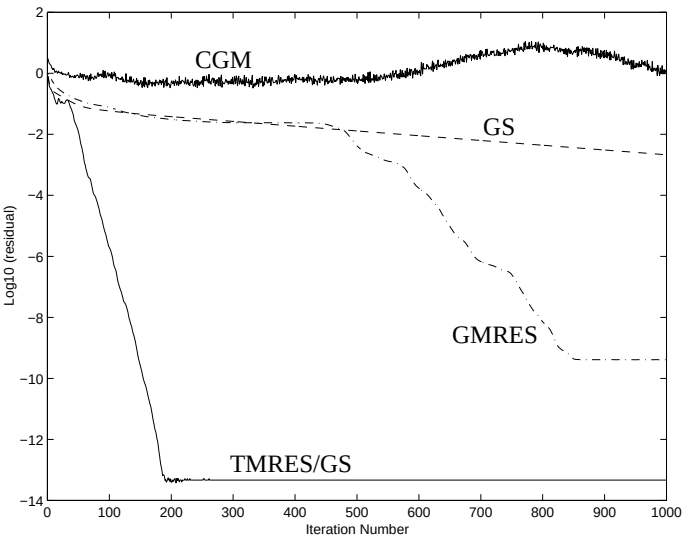


FIG. 5.3. *Convergence for the test problem greenbea, zero rows included,* $\sigma = 10^{-6}$.

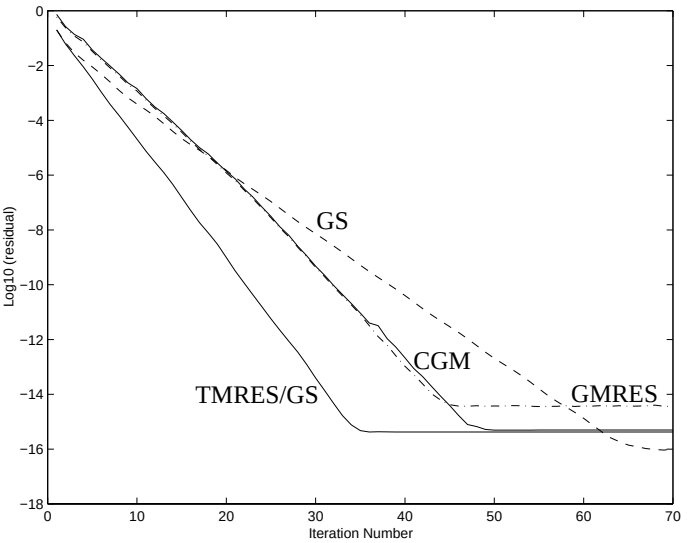


FIG. 5.4. *Convergence for the test problem greenbea,* $\sigma = 10$.

each step, the norm of the residual itself, plotted in Figures 1.1 and 5.1–5.3, is not guaranteed to decay in each iteration. In fact, GMRES is the only algorithm depicted in these figures that is guaranteed (in theory) to reduce the norm of the residual in each step.

Finally, we took $\sigma = 10$ in greenbea in order to examine the convergence speed when the eigenvalues of the matrix are strongly bounded away from zero and the matrix is well-conditioned. As seen in Figure 5.4, all the schemes converge quickly. The convergence of CGM and GMRES is almost identical since the ratio between the largest and smallest eigenvalue in (2.5) is of order 1, and hence, CGM and GMRES

minimize almost the same expression in each step. We emphasize that although TMRES with Gauss–Seidel splitting is effective for nearly singular linear systems such of those found in netlib/lp, its convergence when applied to a well-conditioned partial differential equation may be no better than either CGM or GMRES.

6. Symmetric TMRES. It is well known that for the Arnoldi process (Algorithm 1), the leading submatrices of the upper Hessenberg matrix $\mathbf{H}$ are tridiagonal if $\mathbf{M} = \mathbf{S}^{-1}\mathbf{T}$ is symmetric. Hence, in this symmetric case, we can skip the evaluation of h_{ij} when $i < j - 1$. In the spirit of Hageman and Young's terminology [22], we say that the iteration (3.2) is symmetrizable if there exists an invertible matrix $\mathbf{W}$ such that $\mathbf{WMW}^{-1}$ is symmetric. With the change of variables $\mathbf{x} = \mathbf{W}^{-1}\mathbf{w} + \mathbf{x}_0$, the iteration (3.2) can be written

$$\mathbf{w}_{k+1} = \mathbf{WMW}^{-1}\mathbf{w}_k + \mathbf{Wg}, \quad \mathbf{g} = \mathbf{S}^{-1}(\mathbf{b} - \mathbf{Cx}_0). \tag{6.1}$$

We now form a least-squares problem analogous to (3.4) to obtain an approximation $\mathbf{x}_k$ to a solution of $\mathbf{Cx} = \mathbf{b}$. In particular, $\mathbf{x}_k = \mathbf{W}^{-1}\mathbf{z}_k + \mathbf{x}_0$, where $\mathbf{z}_k$ is a solution to

$$\min \|\mathbf{Wg} - (\mathbf{I} - \mathbf{WMW}^{-1})\mathbf{z}\| \quad \text{subject to} \quad \mathbf{z} \in \mathcal{K}(\mathbf{WMW}^{-1}, \mathbf{Wg}, k). \tag{6.2}$$

This symmetric least squares problem can be solved using Algorithm 2. After taking into account the tridiagonal structure of $\mathbf{H}$, the resulting algorithm is closely connected with Paige and Saunders' MINRES algorithm [40]. Although the orthonormal vectors $\mathbf{v}_i$ for $i \le j$ are needed in Algorithm 2, the following idea of Paige and Saunders [40, 41] can be used to avoid their storage. Instead of evaluating

$$\mathbf{V}_j \left(\mathbf{H}(1:j, 1:j)^{-1}\mathbf{e}(1:j)\right)$$

as we do in Algorithm 2, Paige and Saunders evaluate

$$\left(\mathbf{V}_j\mathbf{H}(1:j, 1:j)^{-1}\right)\mathbf{e}(1:j).$$

Let $\mathbf{R}$ denote the upper triangular matrix $\mathbf{H}(1:j, 1:j)$ generated by Algorithm 2, and define $\mathbf{Z} = \mathbf{V}_j\mathbf{R}^{-1}$. Since the Hessenberg matrix used to generate $\mathbf{R}$ is tridiagonal, $\mathbf{R}$ has three bands, a diagonal band and two superdiagonal bands. Hence, the jth column $\mathbf{z}_j$ of $\mathbf{Z}$ is given by the recurrence

$$\mathbf{z}_j = (\mathbf{v}_j - \mathbf{z}_{j-1}r_{j-1,j} - \mathbf{z}_{j-2}r_{j-2,j})/r_{jj}.$$

The complete algorithm is the following.

ALGORITHM 3 (TMRES for $\mathbf{Cx} = \mathbf{b}$, $\mathbf{WS}^{-1}\mathbf{CW}^{-1}$ symmetric).

$\mathbf{g} \leftarrow \mathbf{S}^{-1}(\mathbf{b} - \mathbf{Cx}_0)$, $\mathbf{e} \leftarrow \mathbf{0}$, $e_1 \leftarrow \|\mathbf{Wg}\|$, $\mathbf{v}_1 \leftarrow \mathbf{Wg}/e_1$
$u \leftarrow 0$, $\mathbf{Q}_0 \leftarrow \mathbf{I}$, $\mathbf{Q}_{-1} \leftarrow \mathbf{0}$
for $j = 1, 2, \ldots$ until convergence
  $\mathbf{r} \leftarrow \mathbf{W}^{-1}\mathbf{v}_j$
  $\mathbf{s} \leftarrow \mathbf{W}(\mathbf{Mr})$ $(\mathbf{C} = \mathbf{S} - \mathbf{T}, \quad \mathbf{M} = \mathbf{S}^{-1}\mathbf{T})$
  $d \leftarrow \mathbf{s}^{\mathsf{T}}\mathbf{v}_j$, $\bar{u} \leftarrow u$
  $\mathbf{s} \leftarrow \mathbf{s} - d\mathbf{v}_j - u\mathbf{v}_{j-1}$
  $u \leftarrow \|\mathbf{s}\|$
  $\mathbf{v}_{j+1} \leftarrow \mathbf{s}/u$
  $d \leftarrow d - 1$

$[t\ \bar{u}] \leftarrow [0\ \bar{u}]\mathbf{Q}_{j-2}^{\mathsf{T}}$
$[\bar{u}\ d] \leftarrow [\bar{u}\ d]\mathbf{Q}_{j-1}^{\mathsf{T}}$
$\mathbf{Q}_j \leftarrow \text{Givens}(d, u)$
$\mathbf{e}(j : j+1) \leftarrow \mathbf{Q}_j \mathbf{e}(j : j+1)$
$d \leftarrow ([d\ u]\mathbf{Q}_j^{\mathsf{T}})_1$
$\mathbf{z}_j \leftarrow (\mathbf{r} - \bar{u}\mathbf{z}_{j-1} - t\mathbf{z}_{j-2})/d$
$\mathbf{x}_j \leftarrow \mathbf{x}_{j-1} - \mathbf{z}_j e_j$
end
end Algorithm 3

Since an iteration of Algorithm 3 requires only the vectors $\mathbf{v}_j$, $\mathbf{v}_{j-1}$, $\mathbf{z}_{j-1}$, $\mathbf{z}_{j-2}$, and $\mathbf{x}_{j-1}$, all the preceding vectors can be discarded. As in Algorithm 2, for a nearly singular linear system $\mathbf{Cx} = \mathbf{b}$, the iteration (6.1) should be devised in such a way that $\rho(\mathbf{M}) \leq 1$. If $\mathbf{z}$ lies in the null space of $\mathbf{C}$, then $\mathbf{Wz}$ is an eigenvector of $\mathbf{WMW}^{-1} = \mathbf{W}(\mathbf{S}^{-1}\mathbf{T})\mathbf{W}^{-1}$ whose eigenvalue is 1. Thus the eigenvalues of smallest modulus in the original problem are associated with eigenvalues of largest modulus in the transformed problem.

There are various ways to choose a symmetrizing matrix $\mathbf{W}$. If $\mathbf{C} = \mathbf{S} - \mathbf{T}$ is a splitting for which $\rho(\mathbf{M}) = \rho(\mathbf{S}^{-1}\mathbf{T}) < 1$ where $\mathbf{S}$ is symmetric and positive definite and $\mathbf{T}$ is symmetric, then we can take $\mathbf{W} = \mathbf{S}^{1/2}$. With this choice for $\mathbf{W}$, we can avoid squares roots in Algorithm 3 if vectors like $\mathbf{s}$ and $\mathbf{v}_j$ are replaced by new variables that are equal to $\mathbf{W}^{-1}$ times old variables. The resulting algorithm is the following.

ALGORITHM 3a (TMRES for $\mathbf{Cx} = \mathbf{b}$, $\mathbf{WS}^{-1}\mathbf{CW}^{-1}$ symmetric).
$\mathbf{g} \leftarrow \mathbf{S}^{-1}(\mathbf{b} - \mathbf{Cx}_0), \quad \mathbf{e} \leftarrow \mathbf{0}, \quad e_1 \leftarrow \|\mathbf{Wg}\|, \quad \mathbf{v}_1 \leftarrow \mathbf{g}/e_1$
$u \leftarrow 0, \quad \mathbf{Q}_0 \leftarrow \mathbf{I}, \quad \mathbf{Q}_{-1} \leftarrow \mathbf{0}$
for $j = 1, 2, \ldots$ until convergence
$\mathbf{r} \leftarrow \mathbf{v}_j$
$\mathbf{s} \leftarrow \mathbf{Mr} \quad (\mathbf{C} = \mathbf{S} - \mathbf{T}, \quad \mathbf{M} = \mathbf{S}^{-1}\mathbf{T})$
$d \leftarrow (\mathbf{Ws})^{\mathsf{T}}(\mathbf{Wv}_j), \quad \bar{u} \leftarrow u$
$\mathbf{s} \leftarrow \mathbf{s} - d\mathbf{v}_j - u\mathbf{v}_{j-1}$
$u \leftarrow \|\mathbf{Ws}\|$
Continue as in Algorithm 3
end
end Algorithm 3a

In this variation of Algorithm 3, $\mathbf{W}$ enters as a product $\mathbf{W}^{\mathsf{T}}\mathbf{W}$. Hence, when $\mathbf{W} = \mathbf{S}^{1/2}$, we have $\mathbf{W}^{\mathsf{T}}\mathbf{W} = \mathbf{S}$, and the square root is gone.

As an illustration of this symmetrization, suppose that $\mathbf{C}$ is symmetric and positive definite and $\mathbf{D}$ is a block diagonal matrix whose diagonal blocks match those of $\mathbf{C}$. The damped Jacobi iteration (see [22]) corresponding to the splitting $\mathbf{S} = \omega\mathbf{D}$ is convergent when the damping parameter ω is sufficiently large, and for this splitting, $\mathbf{S}$ is positive definite. To determine a suitable value for ω, observe that

$$\mathbf{S}^{-1}\mathbf{T} = \mathbf{I} - \frac{1}{\omega}\mathbf{D}^{-1}\mathbf{C},$$

where $\mathbf{D}^{-1}\mathbf{C}$ is similar to the symmetric, positive definite matrix $\mathbf{D}^{-1/2}\mathbf{C}\mathbf{D}^{-1/2}$. The eigenvalues of $\mathbf{S}^{-1}\mathbf{T}$ are real and less than 1 for any choice of $\omega > 0$, and for $\omega > \frac{1}{2}\rho(\mathbf{D}^{-1}\mathbf{C})$, the eigenvalues of $\mathbf{S}^{-1}\mathbf{T}$ are greater than -1. Since the spectral radius of a matrix is bounded by any matrix norm, we have $\rho(\mathbf{M}) \leq \|\mathbf{M}\|_1$. An estimate for the 1-norm of a matrix can be obtained using the algorithm developed

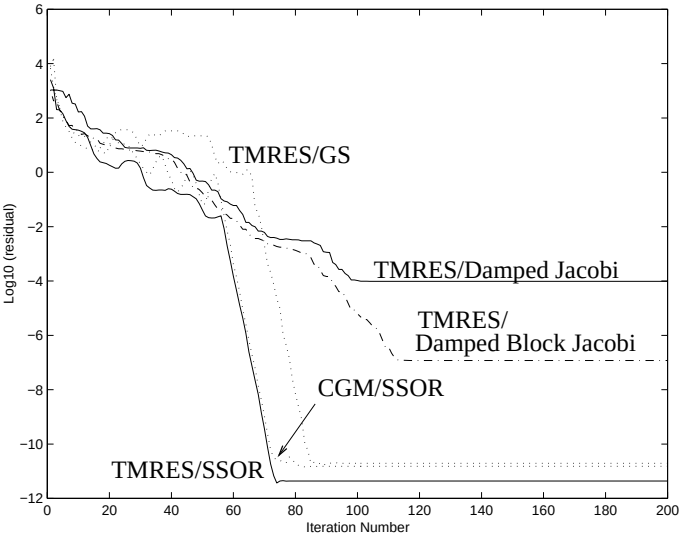


FIG. 6.1. *Convergence for* TMRES *using the test problem* fffff800 *and various symmetrization schemes.*

in [24]. Also see [25, p. 139] and [15, Alg. 7] for symbolic formulations of this algorithm. A Matlab version of this 1-norm estimation algorithm, due to Tim Davis, is available in Mathwork's contributed m-files ftp site, ftp.mathworks.com, as the file pub/contrib/v4/linalg/normest1.m, while Nick Higham gives a Fortran implementation in ACM TOMS Algorithm 674 [28].

In Figure 6.1, we show the convergence of TMRES using the damped Jacobi scheme, where $\mathbf{D}$ is the diagonal of $\mathbf{C}$ and ω is $\frac{2}{3}\|\mathbf{D}^{-1}\mathbf{C}\|_1$, for the test problem fffff800. We also show in Figure 6.1 convergence for TMRES/GS and for a damped Jacobi scheme associated with a block diagonal $\mathbf{D}$ gotten from the Chaco partitioning code [27] of Hendrickson and Rothberg by permuting the rows and columns in order to maximize the number of nonzero elements of $\mathbf{C}$ in diagonal blocks of size 8×8 and 9×9 (60 blocks altogether). Similar partitions are generated by the code Metis [34]. Observe that the TMRES/Block Jacobi scheme converges slightly faster than TMRES/Jacobi, but not as fast as TMRES/GS.

For schemes like Gauss–Seidel or successive overrelaxation (SOR), the $\mathbf{S}$ matrix is not symmetric, and the square root $\mathbf{S}^{1/2}$ may be complex or nonsymmetric. However, when $\mathbf{C} = \mathbf{S} - \mathbf{T}$ is symmetric, a two-step symmetrization is possible using the original iteration followed by its transpose:

$$\mathbf{S}\mathbf{x}_{k+1/2} = \mathbf{T}\mathbf{x}_k + \mathbf{b}, \quad \mathbf{S}^{\mathsf{T}}\mathbf{x}_{k+1} = \mathbf{T}^{\mathsf{T}}\mathbf{x}_{k+1/2} + \mathbf{b}.$$

If the two steps are combined into one step, then

$$\mathbf{x}_{k+1} = \mathbf{M}_2\mathbf{x}_k + \mathbf{g}, \quad \text{where} \quad \mathbf{M}_2 = \mathbf{S}^{-\mathsf{T}}\mathbf{T}^{\mathsf{T}}\mathbf{S}^{-1}\mathbf{T} \quad \text{and} \quad \mathbf{g} = \mathbf{S}^{-\mathsf{T}}(\mathbf{T}^{\mathsf{T}}\mathbf{S}^{-1} + \mathbf{I})\mathbf{b}.$$

In particular, for the SOR splitting of $\mathbf{C}$, this two-step scheme is the same as symmetric SOR (SSOR). As we now show, these two-step schemes are always symmetrizable when $\mathbf{C}$ is symmetric and positive definite.

PROPOSITION 6.1. *If* $\mathbf{C} = \mathbf{S} - \mathbf{T}$, *where* $\mathbf{S}$ *is nonsingular and* $\mathbf{C}$ *is symmetric and positive definite, then* $\mathbf{M}_2 = \mathbf{S}^{-\mathsf{T}}\mathbf{T}^{\mathsf{T}}\mathbf{S}^{-1}\mathbf{T}$ *is symmetrizable with* $\mathbf{W} = \mathbf{C}^{1/2}$.

Proof. Substituting $\mathbf{T} = \mathbf{S} - \mathbf{C}$, we obtain

$$\mathbf{M}_2 = \mathbf{I} - \mathbf{S}^{-\mathsf{T}}\mathbf{C} - \mathbf{S}^{-1}\mathbf{C} + \mathbf{S}^{-\mathsf{T}}\mathbf{C}\mathbf{S}^{-1}\mathbf{C}.$$

Hence, $\mathbf{C}^{1/2}\mathbf{M}_2\mathbf{C}^{-1/2}$ is symmetric. ∎

Let $\mathbf{D}$ be the diagonal of a symmetric, positive definite matrix $\mathbf{C}$. For the SOR splitting $\mathbf{S} = \mathbf{L} + \rho\mathbf{D}$, where $\rho = (1-\omega)/\omega$ and $0 < \omega < 2$, another possible symmetrization is $\mathbf{W} = \mathbf{D}^{-1/2}(\mathbf{L}+\rho\mathbf{D})^{\mathsf{T}}$ (see [22, p. 31]). In Figure 6.1, the curve labeled TMRES/SSOR, we show the convergence of symmetrized TMRES for the Gauss–Seidel/SOR splitting $\mathbf{S} = \mathbf{L}$ ($\rho = 0$). Also in Figure 6.1 we show the convergence of a preconditioned conjugate gradient method [6, 12, 13] with SSOR preconditioner ($\omega = 1$). Observe that the convergence of TMRES/SSOR and CGM/SSOR are very similar. Both schemes generate iterates in the same Krylov space, but they differ in the merit function used to select the approximate solution.

The following identities can be used to streamline the implementation of the SSOR preconditioned iteration:

$$\begin{aligned} \mathbf{M}_2 &= \mathbf{I} - (2\rho+1)(\mathbf{S}^{-\mathsf{T}}\mathbf{D}\mathbf{S}^{-1})\mathbf{C}, \\ \mathbf{W}\mathbf{M}_2\mathbf{W}^{-1} &= \mathbf{I} - (2\rho+1)\mathbf{D}^{1/2}\mathbf{S}^{-1}\mathbf{C}\mathbf{S}^{-\mathsf{T}}\mathbf{D}^{1/2}. \end{aligned} \tag{6.3}$$

From this relation, we see that if $\mathbf{z}$ lies in the null space of $\mathbf{C}$, then $\mathbf{W}\mathbf{z}$ is an eigenvector of $\mathbf{W}\mathbf{M}_2\mathbf{W}^{-1}$ with eigenvalue 1. Hence, eigenvalues of smallest modulus for $\mathbf{C}$ again correspond to eigenvalues of largest modulus for $\mathbf{W}\mathbf{M}_2\mathbf{W}^{-1}$.

In some optimization applications, such as the LP dual active set algorithm, we need to solve many different systems of the form (1.1) where the columns of $\mathbf{A}$ come from the columns of a larger matrix $\mathbf{B}$: $\mathbf{B} = (\mathbf{A}|\mathbf{N})$. As a consequence of the following result, which is basically a reformulation of the classical convergence theorem used for the SOR scheme, the splitting obtained by taking $\mathbf{S} = \mathbf{B}\mathbf{B}^{\mathsf{T}} + \sigma\mathbf{I}$ and $\mathbf{T} = \mathbf{N}\mathbf{N}^{\mathsf{T}}$ is convergent.

THEOREM 6.2. *Suppose that* $\mathbf{C}$ *is symmetric and positive definite and* $\mathbf{C} = \mathbf{S}-\mathbf{T}$, *where* $\mathbf{S}$ *is invertible. If* $\mathbf{S}+\mathbf{T}^{\mathsf{T}}$ *is positive definite, then* $\rho(\mathbf{M}) < 1 > \rho(\mathbf{M}_2)$.

Proof. Our proof that $\rho(\mathbf{M}) < 1$ is essentially that which appears in [49, p. 77] recast in terms of a matrix splitting. A proof that $\rho(\mathbf{M}_2) < 1$ for the SOR iteration appears in [21] (also see the classic reference [52]), while here we obtain a more general result.

Let $\mathbf{f}$ be any vector and define $\mathbf{g} = \mathbf{M}\mathbf{f}$ and $\boldsymbol{\delta} = \mathbf{f}-\mathbf{g}$. From the identity $\mathbf{S}\mathbf{g} = \mathbf{T}\mathbf{f}$, we obtain the following relations:

$$(\mathbf{S}-\mathbf{T})\mathbf{g} + \mathbf{T}\mathbf{g} = \mathbf{T}\mathbf{f} \quad \text{or } \mathbf{C}\mathbf{g} = \mathbf{T}\boldsymbol{\delta},$$

and

$$\mathbf{S}\mathbf{g} = (\mathbf{T}-\mathbf{S})\mathbf{f} + \mathbf{S}\mathbf{f} \quad \text{or } \mathbf{C}\mathbf{f} = \mathbf{S}\boldsymbol{\delta}.$$

Multiplying the first relation by $\mathbf{g}^*$ (the conjugate transpose of $\mathbf{g}$) and the second relation by $\mathbf{f}^*$, subtracting, and exploiting the identity $\mathbf{S}\mathbf{g} = \mathbf{T}\mathbf{f}$, we have

$$\begin{aligned} \mathbf{f}^*\mathbf{C}\mathbf{f} - \mathbf{g}^*\mathbf{C}\mathbf{g} &= (\mathbf{f}^*\mathbf{S} - \mathbf{g}^*\mathbf{T})\boldsymbol{\delta} \\ &= (\mathbf{f}^*\mathbf{S} - \mathbf{g}^*\mathbf{T})\boldsymbol{\delta} + (\mathbf{T}\mathbf{f} - \mathbf{S}\mathbf{g})^*\boldsymbol{\delta} \\ &= \mathbf{f}^*(\mathbf{S}+\mathbf{T}^{\mathsf{T}})\boldsymbol{\delta} - \mathbf{g}^*(\mathbf{T}+\mathbf{S}^{\mathsf{T}})\boldsymbol{\delta}. \end{aligned} \tag{6.4}$$

Since $\mathbf{C}$ is symmetric,

$$\mathbf{C}^{\mathsf{T}} = \mathbf{S}^{\mathsf{T}} - \mathbf{T}^{\mathsf{T}} = \mathbf{C} = \mathbf{S} - \mathbf{T}.$$

Hence, $\mathbf{S} + \mathbf{T}^{\mathsf{T}} = \mathbf{S}^{\mathsf{T}} + \mathbf{T}$, and substituting this in (6.4) gives

$$\mathbf{f}^*\mathbf{C}\mathbf{f} - \mathbf{g}^*\mathbf{C}\mathbf{g} = \boldsymbol{\delta}^*(\mathbf{S} + \mathbf{T}^{\mathsf{T}})\boldsymbol{\delta}. \tag{6.5}$$

If $\mathbf{f}$ is an eigenvector associated with an absolute largest eigenvalue λ of $\mathbf{M}$, then $\mathbf{g} = \mathbf{M}\mathbf{f} = \lambda\mathbf{f}$, and it follows from (6.5) that

$$(1 - |\lambda|^2)\mathbf{f}^*\mathbf{C}\mathbf{f} = \boldsymbol{\delta}^*(\mathbf{S} + \mathbf{T}^{\mathsf{T}})\boldsymbol{\delta} \geq 0$$

since $\mathbf{S} + \mathbf{T}^{\mathsf{T}}$ is positive definite. Since $\mathbf{C}$ is also positive definite, we conclude that $|\lambda| \leq 1$. If $|\lambda| = 1$, then $\boldsymbol{\delta} = \mathbf{0}$, or $\mathbf{g} = \mathbf{f} = \mathbf{M}\mathbf{f}$. Rearranging this last identity gives $\mathbf{C}\mathbf{f} = \mathbf{0}$, which is impossible since $\mathbf{C}$ is invertible. Hence $|\lambda| < 1$ and $\rho(\mathbf{M}) < 1$.

With $\mathbf{h}$ defined by $\mathbf{h} = \mathbf{S}^{-\mathsf{T}}\mathbf{T}^{\mathsf{T}}\mathbf{g} = \mathbf{S}^{-\mathsf{T}}\mathbf{T}^{\mathsf{T}}\mathbf{S}^{-1}\mathbf{T}\mathbf{f} = \mathbf{M}_2\mathbf{f}$, the same manipulations used to obtain (6.5), but with $\mathbf{S}$ and $\mathbf{T}$ replaced by $\mathbf{S}^{\mathsf{T}}$ and $\mathbf{T}^{\mathsf{T}}$, yield the relation

$$\mathbf{g}^*\mathbf{C}\mathbf{g} - \mathbf{h}^*\mathbf{C}\mathbf{h} = \boldsymbol{\epsilon}^*(\mathbf{S} + \mathbf{T}^{\mathsf{T}})\boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} = \mathbf{g} - \mathbf{h}.$$

Adding this relation to (6.5), we obtain

$$\mathbf{f}^*\mathbf{C}\mathbf{f} - \mathbf{h}^*\mathbf{C}\mathbf{h} = \boldsymbol{\delta}^*(\mathbf{S} + \mathbf{T}^{\mathsf{T}})\boldsymbol{\delta} + \boldsymbol{\epsilon}^*(\mathbf{S} + \mathbf{T}^{\mathsf{T}})\boldsymbol{\epsilon} \geq 0.$$

Now, if $\mathbf{f}$ is an eigenvector associated with the absolute largest eigenvalue μ of $\mathbf{M}_2$, then $|\mu| \leq 1$ and $|\mu| = 1$ if and only if $\boldsymbol{\delta} = \mathbf{0} = \mathbf{f} - \mathbf{g}$ and $\boldsymbol{\epsilon} = \mathbf{0} = \mathbf{g} - \mathbf{h}$. Again, the identity $\mathbf{g} = \mathbf{f} = \mathbf{M}\mathbf{f}$ implies that $\mathbf{C}$ is singular. Hence $|\mu| < 1$ and $\rho(\mathbf{M}_2) < 1$. ∎

7. Sparsity considerations. If TMRES is implemented using either a Gauss–Seidel or an SOR splitting, then for a linear system of the form (1.1), it would appear that the product $\mathbf{A}\mathbf{A}^{\mathsf{T}}$ must be evaluated. After forming this product, the number of multiplications and additions needed to perform the iteration is $\mathrm{nnz}(\mathbf{A}\mathbf{A}^{\mathsf{T}})$, where nnz is (Matlab) notation for the number of nonzero elements. When $\mathbf{A}$ is sparse, $\mathbf{A}\mathbf{A}^{\mathsf{T}}$ often has more nonzero entries than $\mathbf{A}$ itself. Hence, for some sparse matrices, it may be more efficient to express an iteration for (1.1) in terms of $\mathbf{A}$ itself rather than $\mathbf{A}\mathbf{A}^{\mathsf{T}}$. In [6] (also see [5, p. 284]) Björck and Elfving show that for SSOR preconditioned conjugate gradient iterations, the matrix-vector product $(\mathbf{W}\mathbf{M}_2\mathbf{W}^{-1})\mathbf{x}$ associated with the SSOR matrix (6.3) and $\mathbf{A}\mathbf{A}^{\mathsf{T}}$ can be evaluated in about $4\mathrm{nnz}(\mathbf{A})$ multiplications since products of the form $\mathbf{S}^{-\mathsf{T}}\mathbf{x}$ and $\mathbf{A}(\mathbf{S}^{-\mathsf{T}}\mathbf{x})$ can be simultaneously evaluated in $2\mathrm{nnz}(\mathbf{A})$ multiplications altogether. A similar approach applies to SOR splittings as we now observe.

If $\mathbf{y}$ and $\mathbf{z}$ denote the iterates $\mathbf{x}_k$ and $\mathbf{x}_{k+1}$, respectively, in (3.2) (so that $\mathbf{S}\mathbf{z} = \mathbf{T}\mathbf{y} + \mathbf{b}$), then for the SOR iteration with relaxation parameter $0 < \omega < 2$, we have

$$z_i - y_i = \frac{\omega\left(b_i - \sum_{j=1}^{i-1} c_{ij}z_j - \sum_{j=i}^{m} c_{ij}y_j\right)}{c_{ii}}, \quad i = 1 \text{ to } m.$$

Since $\mathbf{C} = \mathbf{A}\mathbf{A}^{\mathsf{T}} + \sigma\mathbf{I}$ for the system (1.1), it follows that

$$c_{ij} = \mathbf{a}_i^{\mathsf{T}}\mathbf{a}_j \text{ for } i \neq j, \quad c_{ii} = \sigma + \|\mathbf{a}_i\|^2,$$

TABLE 1
Method and approximate number of multiplications and additions.

Method	Operations
TMRES/SSOR	4nnz($\mathbf{A}$)
CGM/SSOR	4nnz($\mathbf{A}$)
TMRES/SOR	2nnz($\mathbf{A}$)
GS	2nnz($\mathbf{A}$)
GMRES	2nnz($\mathbf{A}$)
TMRES/Jacobi	2nnz($\mathbf{A}$)
CGM	2nnz($\mathbf{A}$)

where $\mathbf{a}_i$ denotes the ith column of $\mathbf{A}^\mathsf{T}$. Let us define $\mathbf{q}_i \in \mathbf{R}^n$ by

$$\mathbf{q}_i = \sum_{j=1}^{i-1} \mathbf{a}_j z_j + \sum_{j=i}^{m} \mathbf{a}_j y_j.$$

From this definition, it follows that

$$z_i - y_i = \omega(b_i - \sigma y_i - \mathbf{q}_i^\mathsf{T}\mathbf{a}_i)/c_{ii}, \quad \text{where } \mathbf{q}_{i+1} = \mathbf{q}_i + \mathbf{a}_i(z_i - y_i).$$

Hence, an iteration of SOR, overwriting the current iterate $\mathbf{x}_k$ by the new iterate $\mathbf{x}_{k+1}$, can be implemented in the following way.

ALGORITHM 4 (SOR for $(\mathbf{A}\mathbf{A}^\mathsf{T} + \sigma\mathbf{I})\mathbf{x} = \mathbf{b}, \mathbf{q} = \mathbf{A}^\mathsf{T}\mathbf{x}_k$).
for $i = 1 : m$
$d \leftarrow \omega(b_i - \sigma x_i - \mathbf{q}^\mathsf{T}\mathbf{a}_i)/(\sigma + \|\mathbf{a}_i\|^2)$
$\mathbf{q} \leftarrow \mathbf{q} + \mathbf{a}_i d$
$x_i \leftarrow x_i + d$
end
end Algorithm 4

After completing the loop in Algorithm 4, $\mathbf{q}$ contains $\mathbf{A}^\mathsf{T}\mathbf{x}_{k+1}$ and $\mathbf{x}$ contains $\mathbf{x}_{k+1}$, assuming $\mathbf{x}$ initially stores $\mathbf{x}_k$. Note that this SOR iteration requires 2nnz($\mathbf{A}$) multiplications and additions.

8. Conclusions. In Table 1 we give work estimates for various iterative schemes studied in this paper and for the prototype system (1.1). We assume that the product $\mathbf{A}\mathbf{A}^\mathsf{T}$ is not formed, and that the computations utilize $\mathbf{A}$ by itself. Table 1 only gives the matrix-vector product work; for Algorithm 3 and symmetric schemes such as CGM, TMRES/SSOR, CGM/SSOR, and TMRES/Jacobi, each iteration involves a small number (between 5 and 10) of additional vector-vector or scalar-vector products. For schemes based on the nonsymmetric Algorithm 2, there are about $(k-1)(k+2)$ additional vector-vector or scalar-vector products.

The numerical experiments in this paper show that iterative schemes may converge slowly when the matrix is nearly singular. In the TMRES approach, we start with a convergent splitting $\mathbf{C} = \mathbf{S} - \mathbf{T}$ and compute an orthonormal basis for a Krylov space generated by $\mathbf{M} = \mathbf{S}^{-1}\mathbf{T}$. Since TMRES is algebraically equivalent to GMRES preconditioned by $\mathbf{S}^{-1}$, TMRES converges in theory if and only if GMRES preconditioned by $\mathbf{S}^{-1}$ converges. We observed that when $\rho(\mathbf{S}^{-1}\mathbf{T}) < 1$, a small dimensional Krylov space often contains a good approximation to the solution of a nearly singular linear system. For the prototype system (1.1), TMRES/SSOR and CGM/SSOR exhibited similar (rapid) convergence and were more efficient than unpreconditioned

schemes. The efficiency of TMRES/Jacobi was comparable to that of the SSOR-based schemes when its lower operation count 2nnz(**A**) was taken into account.

Acknowledgment. The author gratefully acknowledges the many thoughtful and perceptive comments of the referees and their pointers to related work.

REFERENCES

[1] L. Adams and J. L. Nazareth, eds., *Linear and Nonlinear Conjugate Gradient-Related Methods*, SIAM, Philadelphia, PA, 1996.

[2] E. Allgower and K. Georg, *Simplicial and continuation methods for approximating fixed points and solutions to systems of equations*, SIAM Rev., 22 (1980), pp. 28–85.

[3] W. E. Arnoldi, *The principle of minimized iteration in the solution of the matrix eigenvalue problem*, Quart. Appl. Math., 9 (1951), pp. 17–29.

[4] J. Baglama, D. Calvetti, G. H. Golub, and L. Reichel, *Adaptively preconditioned GMRES algorithms*, SIAM J. Sci. Comput., 20 (1998), pp. 243–269.

[5] Å. Björck, *Numerical Methods for Least Squares Problems*, SIAM, Philadelphia, PA, 1996.

[6] Å. Björck and T. Elfving, *Accelerated projection methods for computing pseudoinverse solutions of systems of linear equations*, BIT, 19 (1979), pp. 145–163.

[7] P. N. Brown and H. F. Walker, *GMRES on (nearly) singular systems*, SIAM J. Matrix Anal. Appl., 18 (1997), pp. 37–51.

[8] R. H. Byrd, R. B. Schnabel, and G. A. Schultz, *A trust region algorithm for nonlinearly constrained optimization*, SIAM J. Numer. Anal., 24 (1987), pp. 1152–1170.

[9] S. L. Campbell, I. C. F. Ipsen, C. T. Kelley, and C. D. Meyer, *GMRES and the minimal polynomial*, BIT, 36 (1996), pp. 664–675.

[10] M. R. Celis, J. E. Dennis, and R. A. Tapia, *A trust region strategy for nonlinear equality constrained optimization*, in Numerical Optimization, SIAM, Philadelphia, PA, 1985, pp. 71–82.

[11] T. F. Chan, *Newton-like pseudo-arclength methods for computing simple turning points*, SIAM J. Sci. Statist. Comput., 5 (1984), pp. 135–148.

[12] P. Concus, G. H. Golub, and D. P. O'Leary, *A generalized conjugate gradient method for the numerical solution of elliptic partial differential equations*, in Sparse Matrix Computations, J. R. Bunch and D. J. Rose, eds., Academic Press, New York, 1976, pp. 309–332.

[13] P. Concus, G. H. Golub, and D. P. O'Leary, *Numerical solution of nonlinear elliptic partial differential equations by a generalized conjugate gradient method*, Computing, 19 (1978), pp. 321–339.

[14] J. W. Daniel, *The conjugate gradient method for linear and nonlinear operator equations*, SIAM J. Numer. Anal., 4 (1967), pp. 10–26.

[15] T. A. Davis and W. W. Hager, *Modifying a sparse Cholesky factorization*, SIAM J. Matrix Anal. Appl., 20 (1999), pp. 606–627.

[16] M. El-Alem, *A global covergence theory for the Celis-Dennis-Tapia trust-region algorithm for constrained optimization*, SIAM J. Numer. Anal., 28 (1991), pp. 266–290.

[17] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 2nd ed., Johns Hopkins University Press, Baltimore, MD, 1989.

[18] A. Greenbaum *Comparison of splittings used with the conjugate gradient algorithm*, Numer. Math., 33 (1979), pp. 181–194.

[19] A. Greenbaum, V. Pták, and Z. Strakoš, *Any nonincreasing convergence curve is possible for GMRES*, SIAM J. Matrix Anal. Appl., 17 (1996), pp. 465–469.

[20] A. Greenbaum and Z. Strakoš, *Predicting the behavior of finite precision Lanczos and conjugate gradient computations*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 121–137.

[21] G. J. Habetler and E. L. Wachspress, *Symmetric successive overrelaxation in solving diffusion difference equations*, Math. Comp., 15 (1961), pp. 356–362.

[22] L. A. Hageman and D. M. Young, *Applied Iterative Methods*, Academic Press, New York, 1981.

[23] W. W. Hager, *The LP dual active set algorithm*, in High Performance Algorithms and Software in Nonlinear Optimization, R. D. Leone, A. Murli, P. M. Pardalos, and G. Toraldo, eds., Kluwer, Dordrecht, the Netherlands, 1998, pp. 243–254.

[24] W. W. Hager, *Condition estimates*, SIAM J. Sci. Statist. Comput., 5 (1984), pp. 311–316.

[25] W. W. Hager, *Applied Numerical Linear Algebra*, Prentice–Hall, Englewood Cliffs, NJ, 1988 (available from W. W. Hager, Department of Mathematics, University of Florida, Gainesville).

[26] W. W. Hager, *Minimizing a Quadratic over a Sphere*, Department of Mathematics, University of Florida, Gainesville, http://www.math.ufl.edu/∼hager/papers/sphere.ps (May 1999).
[27] B. Hendrickson and E. Rothberg, *Improving the Runtime and Quality of Nested Dissection Ordering*, Technical report, Sandia National Laboratories, Albuquerque, NM, 1997.
[28] N. J. Higham, *Fortran codes for estimating the one-norm of a real or complex matrix with applications to condition estimation*, ACM Trans. Math. Software, 14 (1988), pp. 381–396.
[29] R. A. Horn and C. R. Johnson, *Topics in Matrix Analysis*, Cambridge University Press, Cambridge, UK, 1991.
[30] I. C. F. Ipsen and C. D. Meyer, *The idea behind Krylov methods*, Amer. Math. Monthly, 105 (1998), pp. 889–899.
[31] K. C. Jea and D. M. Young, *Generalized conjugate gradient acceleration of nonsymmetrizable iterative methods*, Linear Algebra Appl., 34 (1980), pp. 159–194.
[32] A. Jennings, *Influence of the eigenvalue spectrum on the convergence rate of the conjugate gradient method*, J. Inst. Math. Appl., 20 (1977), pp. 61–72.
[33] N. Karmarkar, *A new polynomial time algorithm for linear programming*, Combinatorica, 4 (1984), pp. 373–395.
[34] G. Karypis and V. Kumar, *METIS: Unstructured Graph Partitioning and Sparse Matrix Ordering System*, Technical report, Department of Computer Science, University of Minnesota, Minneapolis, 1995.
[35] D. G. Luenberger, *Convergence rate of a penalty-function scheme*, J. Optim. Theory Appl., 7 (1971), pp. 39–51.
[36] D. G. Luenberger, *Linear and Nonlinear Programming*, Addison-Wesley, Reading, MA, 1984.
[37] J. C. Meza and W. W. Symes, *Deflated Krylov subspace methods for nearly singular linear systems*, J. Optim. Theory Appl., 72 (1992), pp. 441–457.
[38] J. J. Moré, *Recent developments in algorithms and software for trust region methods*, in A. Bachem, M. Grotschel, and B. Korte, eds., Mathematical Programming: State of the Art, Springer-Verlag, Berlin, 1983, pp. 258–287.
[39] N. M. Nachtigal, S. C. Reddy, and L. N. Trefethen, *How fast are nonsymmetric matrix iterations?*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 778–795.
[40] C. C. Paige and M. A. Saunders, *Solution of sparse indefinite systems of linear equations*, SIAM J. Numer. Anal., 12 (1975), pp. 617–629.
[41] C. C. Paige and M. A. Saunders, *LSQR: An algorithm for sparse linear equations and sparse least squares*, ACM Trans. Math. Software, 8 (1982), pp. 43–71.
[42] M. J. D. Powell and Y. Yuan, *A trust region algorithm for equality constrained optimization*, Math. Programming, 49 (1991), pp. 189–211.
[43] Y. Saad and M. H. Schultz, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 856–869.
[44] M. A. Saunders, *Solution of sparse rectangular systems using LSQR and CRAIG*, BIT, 35 (1995), pp. 588–604.
[45] M. A. Saunders, *Computing projections with LSQR*, BIT, 37 (1997), pp. 96–104.
[46] D. C. Sorensen, *Implicit application of polynomial filters in a k-step Arnoldi method*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 357–385.
[47] G. W. Stewart, *The convergence of the method of conjugate gradients at isolated extreme points in the spectrum*, Numer. Math., 24 (1975), pp. 85–93.
[48] E. L. Stiefel, *Kernal polynomials in linear algebra and their numerical applications*, Nat. Bur. Standards Appl. Math. Ser., 49 (1958), pp. 1–22.
[49] R. S. Varga, *Matrix Iterative Analysis*, Prentice–Hall, Englewood Cliffs, NJ, 1962.
[50] P. K. W. Vinsome, *ORTHOMIN, an iterative method for solving sparse sets of simultaneous linear equations*, in Proceedings of the Fourth Symposium on Reservoir Simulation, Society of Petroleum Engineers of AIME, 1976, pp. 149–159.
[51] S. J. Wright, *Primal-Dual Interior-Point Methods*, SIAM, Philadelphia, PA, 1997.
[52] D. M. Young, *Iterative Solution of Large Linear Systems*, Academic Press, New York, 1971.

SIAM J. SCI. COMPUT.
Vol. 22, No. 3, pp. 767–786
© 2000 Society for Industrial and Applied Mathematics

A CIRCULANT PRECONDITIONER FOR THE SYSTEMS OF LMF-BASED ODE CODES*

D. BERTACCINI†

Abstract. In this paper, a recently introduced block circulant preconditioner for the linear systems of the codes for ordinary differential equations (ODEs) is investigated. Most ODE codes based on implicit formulas, at each integration step, need the solution of one or more unsymmetric linear systems that are often large and sparse. Here, the boundary value methods, a class of implicit methods for the numerical integration of ODEs based on linear multistep formulas, are considered more in detail for initial value problems.

Theoretical and practical arguments are given to show that the block circulant preconditioner can give fast preconditioned iterations for various classes of differential problems. Moreover, the *P-circulants*, a recently introduced circulant approximation for unsymmetric Toeplitz matrices, are shown to be more suitable sometimes than other circulant matrices for the underlying block preconditioner.

Key words. circulant preconditioning, unsymmetric block (almost) Toeplitz linear systems, numerical solution of differential equations, boundary value methods, implicit linear multistep formulas

AMS subject classifications. 65F10, 65N22, 65L05, 15A18

PII. S1064827599353476

1. Introduction.

The aim of this paper is to study a new class of block preconditioners for the linear systems

$$Mx = b \tag{1.1}$$

arising in the codes for the numerical integration of ordinary differential equations (ODEs) based on linear multistep formulas (LMFs) (see, e.g., [18]). The solution of the large and sparse system of equations, arising at each integration step, is one of the crucial parts in a numerical integrator based on implicit formulas; see, e.g., [12, 4]. Here will be considered the linear systems arising in boundary value methods (BVMs) (see [3] and references therein) or that can be reduced to those with some transformation on (1.1). BVMs are a class of numerical methods based on LMFs solving initial and boundary value problems for ordinary differential equations.

The underlying block preconditioner is based on the circulant approximation of the (small rank perturbation of) band Toeplitz matrices arising in (1.1). To this end, a new type of circulant approximation for Toeplitz matrices introduced in [1], the *P-circulant matrices*, or *P-circulants*, can be effective. T. F. Chan's *optimal* [5] and Strang's circulant [23] will also be considered. Other techniques for band Toeplitz linear systems can be found, e.g., in [11, 6, 16], but they are effective only in the symmetric case while, in general, matrices for (1.1) are not symmetric.

An $n \times n$ matrix T is said to be *Toeplitz* if its entries are constant along its diagonals; an $n \times n$ matrix C is called *circulant* if it is a Toeplitz matrix with the

*Received by the editors March 12, 1999; accepted for publication (in revised form) April 15, 2000; published electronically August 31, 2000.
http://www.siam.org/journals/sisc/22-3/35347.html

†Università di Roma "La Sapienza," Istituto Matematico "G. Castelnuovo," P. le A. Moro 2, 00185 Roma, Italy (bertaccini@na-net.ornl.gov). This research was partially supported by the Italian Ministry of Scientific Research.

following pattern:

$$C = circ(c_0, \ldots, c_{n-1}) = \begin{pmatrix} c_0 & c_1 & \ldots & c_{n-1} \\ c_{n-1} & c_0 & & c_{n-2} \\ \vdots & \vdots & \ddots & \vdots \\ c_1 & c_2 & \ldots & c_0 \end{pmatrix}. \tag{1.2}$$

In the last decade, there has been intensive work on preconditioners for Toeplitz matrices and their spectral properties; see, for instance, the survey [8], and the references therein, and [11, 5, 6, 7, 14, 16, 21, 23, 24].

Consider for simplicity the linear initial value problem (IVP)

$$\begin{cases} y'(t) &= f(t, y(t)) := J\, y(t) + g(t), \quad t \in (t_0, T], \\ y(t_0) &= \eta, \end{cases} \tag{1.3}$$

where $y(t), g(t) : \mathbb{R} \to \mathbb{R}^m$, $J \in \mathbb{R}^{m \times m}$, $\eta \in \mathbb{R}^m$. The matrix M (1.1) related to the underlying LMF-based code, in general, can be written as

$$M = A \otimes I_m - hB \otimes J, \tag{1.4}$$

where A, B are the matrices of the integration method, h is the integration stepsize, and $\otimes$ is the Kronecker product (see, e.g., [19, Chapter 12]). A and B are usually reducible to Toeplitz plus small rank perturbation pattern. M turns out to be large and sparse when the Jacobian matrix of the underlying system of ODEs (and/or A and B) is so. In that case, the solution of the linear system (1.1) via a direct method is often computationally expensive and not easily parallelizable; therefore the use of a preconditioned iterative method is preferable.

We stress that the proposed iterative technique can be naturally generalized to the case of d-level structures arising in d-dimensional partial differential equations (PDEs). In that case, the direct solvers become very expensive since they do not exploit properly the multiple band structure (see section 5, Example 2).

It is interesting to observe that P-circulants preserve important properties occurring in the matrices A, B in (1.4), such as

- invertibility,
- eigenvalues in the right half plane,
- almost the same sparsity pattern.

Moreover, our block preconditioner has interesting implementation potentialities, in both a scalar and a parallel computing environment.

Under appropriate hypotheses, the results obtained here can be extended to the case where the given ODE is nonlinear. Indeed, we can observe that the discrete nonlinear problem corresponding to the approximation by an LMF can be solved iteratively by considering, at each step, discrete problems such as (1.1). Notice that there are several important additional topics to take into account in the nonlinear case, such as the convergence of the modified Newton approach (see, e.g., [13]) and the mesh selection strategy.

The paper is organized as follows. In section 2 we recall some classes of BVMs. In section 3 we introduce the block circulant preconditioner and the circulant approximations. Section 4 contains some notes on the computational cost and on the convergence of preconditioned iterations. Finally, section 5 contains some numerical tests.

2. BVMs and their matrix form. Recently, the class of BVMs for differential equations has been introduced (see [3] and references therein). Such methods are based on LMFs. In order to briefly describe them, suppose for simplicity that we have the linear IVP (1.3). A BVM approximates the solution of (1.3) by means of a discrete boundary value problem (BVP). The latter is obtained by using a k-step linear multistep formula of order p over a uniform mesh $t_j = t_0 + j\,h,\ j = 0, \ldots, s$, $h = (T - t_0)/s$:

$$\sum_{i=-\nu}^{k-\nu} \alpha_{i+\nu} y_{n+i} = h \sum_{i=-\nu}^{k-\nu} \beta_{i+\nu} f_{n+i}, \quad n = \nu, \ldots, s-k+\nu. \tag{2.1}$$

As usual, y_n is the discrete approximation to $y(t_n)$, $f_n = f(t_n, y_n) \equiv J\,y_n + g_n$, $g_n = g(t_n)$, and the values

$$y_0, \ldots, y_{\nu-1}, \quad y_{s-k+\nu+1}, \ldots, y_s \tag{2.2}$$

are given. We observe that the IVP (1.3) provides only the initial value y_0. It is possible to avoid supplying the other conditions in (2.2) by coupling the *main method* (2.1) with other difference schemes of order p, called *additional methods*, which provide the set of equations

$$\sum_{i=0}^{k} \alpha_i^{(j)} y_i = h \sum_{i=0}^{k} \beta_i^{(j)} f_i, \quad j = 1, \ldots, \nu-1, \tag{2.3}$$

$$\sum_{i=0}^{k} \alpha_{k-i}^{(j)} y_{s-i} = h \sum_{i=0}^{k} \beta_{k-i}^{(j)} f_{s-i}, \quad j = s-k+\nu+1, \ldots, s, \tag{2.4}$$

independent of those in (2.1). For simplicity, such formulas are assumed to have the same number of steps as the main method. The equations (2.1), (2.3), and (2.4) define the use of a BVM on problem (1.3). The advantage in using BVMs, over the known LMFs requiring only initial conditions, derives from their stability properties.

It is useful to cast BVMs in matrix form. This is done by introducing the matrices $A, B \in \mathbb{R}^{(s+1)\times(s+1)}$

$$A = \begin{pmatrix} 1 & \cdots & 0 & & & & \\ \alpha_0^{(1)} & \cdots & \alpha_k^{(1)} & & & & \\ \vdots & \vdots & \vdots & & & & \\ \alpha_0^{(\nu-1)} & \cdots & \alpha_k^{(\nu-1)} & & & & \\ \alpha_0 & \cdots & \alpha_k & & & & \\ & \alpha_0 & \cdots & \alpha_k & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & \alpha_0 & \cdots & \alpha_k & \\ & & & \alpha_0^{(s-k+\nu+1)} & \cdots & \alpha_k^{(s-k+\nu+1)} & \\ & & & \vdots & \vdots & \vdots & \\ & & & \alpha_0^{(s)} & \cdots & \alpha_k^{(s)} & \end{pmatrix}, \tag{2.5}$$

and B similarly but with β_j's instead of α_j's and all zeros in its first row. The discrete problem generated by the application of the BVM (2.1)–(2.4) to problem (1.3) is then

given by

$$M\,Y = e_1 \otimes \eta + h\,(B \otimes I)g, \tag{2.6}$$
$$e_1 = (1, 0, \ldots, 0)^T \in \mathbb{R}^{s+1}, \quad Y = (y_0, \ldots, y_s)^T, \quad g = (g_0, \ldots, g_s)^T,$$

$$M = A \otimes I_m - h\,B \otimes J.$$

The matrix M in (2.6) turns out to be large and sparse when $s \gg k$ and/or J is large and sparse.

2.1. Some families of BVMs. Here we give the definitions of some families of BVMs (see [3] for details). All considered methods are consistent, i.e., they satisfy the conditions

$$\rho(1) = 0, \qquad \rho'(1) = \sigma(1),$$

where $\rho(z)$ and $\sigma(z)$ denote, as usual, the two characteristic polynomials associated with the given method, i.e.,

$$\rho(z) = z^\nu \sum_{j=-\nu}^{k-\nu} \alpha_{j+\nu}\, z^j, \quad \sigma(z) = z^\nu \sum_{j=-\nu}^{k-\nu} \beta_{j+\nu}\, z^j. \tag{2.7}$$

The generalized BDF, or GBDF, are a generalization of the backward differentiation formulas (BDF) (see [13]). They can be written in the form

$$\sum_{i=-\nu}^{k-\nu} \alpha_{i+\nu}\, y_{n+i} = h\, f_n, \quad n = \nu, \ldots, s-k+\nu, \tag{2.8}$$

where the coefficients $\{\alpha_i\}$ are uniquely determined by imposing that the method has maximum order, i.e., k, for all $k \geq 1$, with $\nu = (k+2)/2$ if k is even and $\nu = (k+1)/2$ if k is odd. Such methods are well suited for stiff problems. (See [3, Chapter 5] for details.)

The generalized Adams methods (GAM) are a generalization of the Adams–Moulton methods (see [13]). They can be written in the form

$$y_{n+\nu} - y_{n+\nu-1} = h \sum_{i=-\nu}^{k-\nu} \beta_{i+\nu}\, f_{n+i}, \tag{2.9}$$

where the coefficients $\{\beta_i\}$ are uniquely determined by imposing that the method has maximum order, i.e., $k+1$, for all $k \geq 1$, with $\nu = k/2$ if k is even and $(k+1)/2$ if k is odd. When k is odd, they are called extended trapezoidal rules (ETR), because they share the same stability properties of the trapezoidal rule. Such methods turn out to be well suited for approximating either Hamiltonian problems or continuous BVPs. When k is even, GAM are well suited for stiff problems (see [3, Chapters 6 and 7] for details).

ETR_2 are another generalization of the trapezoidal rule belonging to the class of symmetric schemes:

$$\sum_{i=-\nu}^{\nu-1} \alpha_{i+\nu}\, y_{n+i} = \frac{h}{2}(f_n + f_{n-1}), \tag{2.10}$$

where $k = 2\nu - 1$ is odd and the coefficients $\{\alpha_i\}$ are uniquely determined by imposing that the method has maximum order, i.e., $k+1$, $k = 1, 3, 5, \ldots$. When $\nu = 1$, then $k = 1$ and the formulas (2.10), (2.9) become the trapezoidal rule. Indeed, all such formulas can be regarded as generalizations of this method, sharing the same stability properties. Such methods turn out to be well suited for approximating both Hamiltonian problems and continuous BVPs (see [3, Chapter 7] for details).

3. Circulant approximations for the block preconditioner. Let M in (2.6) be the small rank perturbation of a block Toeplitz matrix generated by a BVM as in the previous section. A way to solve such large, sparse linear systems is through an iterative method; see, e.g., [12, 4]. To accelerate convergence, a preconditioner P should be chosen to approximate the matrix M while keeping the system

$$P\,x = c$$

cheap enough to solve with respect to the unpreconditioned iterations.

In order to obtain the preconditioner, let us consider the following approximation of the matrix M:

$$P = \breve{A} \otimes I_m - h\,\breve{B} \otimes \hat{J}, \tag{3.1}$$

where $\hat{J}$ can be a suitable approximation of the Jacobian matrix of the ODE, or the Jacobian itself. $\breve{A}$, $\breve{B}$ are circulant matrices the entries of which are derived from the coefficients of the main method (2.1) as follows:

$$\begin{aligned} &\breve{A} = circ(\tilde{\alpha}_j), \qquad \tilde{\alpha}_j = c_{j,1}(s)\alpha_{j+\nu} + c_{j,2}(s)\alpha_{j+\nu-(s+1)}, \\ &\breve{B} = circ(\tilde{\beta}_j), \qquad \tilde{\beta}_j = c_{j,3}(s)\beta_{j+\nu} + c_{j,4}(s)\beta_{j+\nu-(s+1)}, \qquad j = 0, \ldots, s, \end{aligned} \tag{3.2}$$

where the $c_{j,i}(s)$, $i = 1, \ldots, 4$, $j = 0, \ldots, s$ are linear in j. It is understood that α_j (β_j) is zero for $j < 0$ or $j > k$ in (3.2). The coefficients $c_{i,j}(s)$ in (3.2) are chosen in such a way that $\breve{A}$, $\breve{B}$ are suitable approximations of A, B in (2.5), respectively.

The approximation of A, B with Chan's optimal circulant (see [5]) requires that

$$c_{j,1}(s) = c_{j,3}(s) = 1 - \frac{j}{s+1}, \qquad c_{j,2}(s) = c_{j,4}(s) = \frac{j}{s+1}, \qquad j = 0, \ldots, s, \tag{3.3}$$

while for Strang's natural circulant (see [23])

$$c_{j,1}(s) = c_{j,3}(s) = 1, \qquad j = 0, \ldots, \left\lfloor \frac{s+1}{2} \right\rfloor,$$

$$c_{j,2}(s) = c_{j,4}(s) = 1, \qquad j = \left\lfloor \frac{s+1}{2} \right\rfloor + 1, \ldots, s, \qquad c_{r,j}(s) = 0 \ \text{ otherwise.} \tag{3.4}$$

Let us observe that, for an $(s+1) \times (s+1)$ Toeplitz matrix T, Chan's circulant $C = C(T)$ is defined to be the minimizer of

$$\|T - C\|_F \tag{3.5}$$

over all $(s+1) \times (s+1)$ circulant matrices C; see [5]. ($\|\cdot\|_F$ is the Frobenius norm.) Consider, instead of (3.3), the following definition of the coefficients $c_{j,i}(s)$:

$$c_{j,1}(s) = c_{j,3}(s) = 1 + \frac{j}{s+1}, \qquad c_{j,2}(s) = c_{j,4}(s) = \frac{j}{s+1}, \qquad j = 0, \ldots, s. \tag{3.6}$$

We will call P-circulants the circulant matrices defined in (3.2), (3.6). The definitions (3.2), (3.3), (3.2), (3.4) and (3.2), (3.6) differ for a quantity that vanishes as s increases for banded matrices, and (3.5) is not minimized for P-circulants. Despite this, the P-circulant matrices $\breve{A}$, $\breve{B}$, defined as (3.2), (3.6), are nonsingular and appropriate approximations of A, B given by (2.5), as observed in [2].

3.1. Spectral properties of the block preconditioner. We observe that positive stability is a sufficient condition for the invertibility of a matrix. Recall that a square matrix A is said to be positive stable if its eigenvalues have positive real part (see, e.g., [15]). The matrices A, B of the BVMs we consider here are assumed to be positive stable. This assumption is a consequence of the stability properties of the formulas; see [3]. It is interesting to observe that the P-circulant approximations $\breve{A}$, $\breve{B}$ given by (3.2), (3.6) preserve positive stability for the methods described in section 2.1 (see [2]). Let $\breve{A} = circ(\tilde{\alpha}_j)$ be a $(s+1) \times (s+1)$ circulant matrix defined in (3.2), (3.6) from the first characteristic polynomial of the main method (2.1). The eigenvalues ϕ_j, $j = 0, \ldots, s$, of $\breve{A}$ can be written as linear combinations of the entries of its first row (see [10]):

$$\phi_l = \sum_{j=0}^{s} \tilde{\alpha}_j \epsilon^{jl}, \quad l = 0, \ldots, s, \quad \epsilon = e^{2\pi \mathbf{i}/(s+1)}, \tag{3.7}$$

where $\mathbf{i}$ is the imaginary unit. From (3.2), (3.6) one obtains

$$\phi_l = \sum_{j=0}^{s} \alpha_{j+\nu} \left(1 + \frac{j}{s+1}\right) \epsilon^{jl} + \sum_{j=0}^{s} \left(\frac{j}{s+1} \alpha_{j+\nu-(s+1)}\right) \epsilon^{jl},$$

which can be restated as

$$\phi_l = \sum_{j=-\nu}^{k-\nu} \alpha_{j+\nu} \left(1 + \frac{j}{s+1}\right) \epsilon^{jl}, \qquad l = 0, \ldots, s. \tag{3.8}$$

A similar expression holds for the eigenvalues of $\breve{B}$:

$$\psi_l = \sum_{j=-\nu}^{k-\nu} \beta_{j+\nu} \left(1 + \frac{j}{s+1}\right) \epsilon^{jl}, \qquad l = 0, \ldots, s. \tag{3.9}$$

In [2] we have observed that, for some A-stable ($A_{\nu,k-\nu}$-stable, a generalization of A-stability for (2.1) if $k > \nu$; see [3]) BVMs, the $l2$-norms of the P-circulant matrices $\breve{A}$, $\breve{B}$ are uniformly bounded and the $l2$-norms of their inverses are bounded with respect to the number of steps k of the method (2.1), i.e.,

$$\|\breve{A}\|_2 \leq c_1, \qquad \|\breve{B}\|_2 \leq c_2, \qquad \|\breve{A}^{-1}\|_2 \leq c_3 \cdot (s+1), \qquad \|\breve{B}^{-1}\|_2 \leq c_4 \cdot (s+1), \tag{3.10}$$

where c_j, $j = 1, \ldots, 4$, are constants of the order of unity. Exceptions are the GBDF (they have $\breve{B} \equiv I$) and the ETR_2.

Notice that the circulant matrix $\breve{A}$ defined as the optimal circulant (3.2), (3.3), may become severely ill-conditioned as k increases (see [2]).

We can give a sample of sufficient conditions for the block preconditioner (3.1) to be positive stable. From here on, for simplicity, we will assume the Jacobian matrix to be diagonalizable.

PROPOSITION 3.1. *Suppose that the eigenvalues μ_r, $r = 1, \ldots, m$, of the approximation of the Jacobian matrix J of the given ODE are in the left half plane, i.e., they have nonpositive real part. Each of the following conditions is sufficient for the positive stability of the block preconditioner P given by* (3.1):

1. *$\breve{A}$, $\breve{B}$ are P-circulants and* (2.1) *belongs to GBDF;*
2. *$\breve{A}$, $\breve{B}$ are positive stable and $\breve{B}$ is Hermitian;*
3. *$\breve{A}$, $\breve{B}$ are positive stable and, for $j \neq 0, (s+1)/2$, we have*

$$|\mathrm{Im}(\mu_r)| < \left(\frac{\mathrm{Re}(\phi_j) - h\mathrm{Re}(\psi_j)\mathrm{Re}(\mu_r)}{h\,|\mathrm{Im}(\psi_j)|} \right), \ r = 1, \ldots, m, \tag{3.11}$$

where ϕ_j, ψ_j, $j = 0, \ldots, s$, are the eigenvalues of $\breve{A}$, $\breve{B}$, respectively.

Proof. If J is the Jacobian matrix,

$$V^{-1} J V = D = \mathrm{diag}(\mu_1, \ldots, \mu_m). \tag{3.12}$$

The circulant matrices $\breve{A}$ and $\breve{B}$ are simultaneously diagonalized by the unitary Fourier matrix F (see [10])

$$F = (F)_{j,r}\,, \qquad (F)_{j,r} = \frac{1}{\sqrt{s+1}} \epsilon^{jr}, \qquad \epsilon = e^{2\pi\,\mathbf{i}/(s+1)}, \qquad 0 \leq j, r \leq s, \tag{3.13}$$

i.e.,

$$F\,\breve{A}\,F^* = \Lambda_A = \mathrm{diag}(\phi_0, \ldots, \phi_s), \quad F\,\breve{B}\,F^* = \Lambda_B = \mathrm{diag}(\psi_0, \ldots, \psi_s).$$

We can write

$$\begin{aligned} P &= (F^* \otimes I_m)\ (\Lambda_A \otimes I_m - \Lambda_B \otimes h\,J)\,(F \otimes I_m) \\ &= (F^* \otimes V)\ (\Lambda_A \otimes I_m - \Lambda_B \otimes h\,D)\left(F \otimes V^{-1}\right), \end{aligned} \tag{3.14}$$

where Λ_A, Λ_B are nonsingular diagonal matrices of size $s+1$. Let

$$\breve{\Lambda} = \Lambda_A \otimes I_m - \Lambda_B \otimes h\,D, \tag{3.15}$$

$$\begin{aligned} \breve{\Lambda} = \mathrm{diag}\,(&\phi_0 - h\psi_0\mu_1, \ldots, \phi_0 - h\psi_0\mu_m, \ldots, \\ &\phi_s - h\psi_s\mu_1, \ldots, \phi_s - h\psi_s\mu_m)\,. \end{aligned} \tag{3.16}$$

$\breve{\Lambda}$ is diagonal and, if

$$\mathrm{Re}(\phi_j - h\psi_j\mu_r) > 0, \quad j = 0, \ldots, s, \quad r = 1, \ldots, m, \tag{3.17}$$

then $\breve{\Lambda}$ (and P) is also positive stable. If $\breve{A}$, $\breve{B}$ are P-circulants, or, more generally, for all block preconditioners whose matrices $\breve{A}$, $\breve{B}$ are positive stable and $\breve{B}$ is Hermitian, we have that (3.17) holds true. Indeed, by the hypotheses, we have

$$\mathrm{Re}(\mu_r) \leq 0, \qquad \mathrm{Re}(\phi_j), \mathrm{Re}(\psi_j) > 0$$

and $\mathrm{Im}(\psi_j) = 0$, $j = 0, \ldots, s$, $r = 1, \ldots, m$.

If $\breve{B}$ is not Hermitian, the condition (3.17) is equivalent to

$$\mathrm{Re}(\phi_j) + h(\mathrm{Im}(\psi_j)\mathrm{Im}(\mu_r) - \mathrm{Re}(\psi_j)\mathrm{Re}(\mu_r)) > 0, \tag{3.18}$$

which can be restated as in (3.11). ∎

Notice that P may not be positive stable for certain formulas, even if the Jacobian matrix J has the spectrum in the left half plane.

Let us observe that, if the eigenvalues μ_r, $r = 1, \ldots, m$, of the matrix J of the IVP are in the left half plane, then the block preconditioner's matrix P is nonsingular for all schemes whose matrices $\breve{A}$, $\breve{B}$ in (3.2) are such that

$$\mathrm{Re}\left(\frac{\phi_j}{\psi_j}\right), \quad j = 0, \ldots, s, \tag{3.19}$$

is positive. Indeed, P is singular if and only if there exists at least a couple j, r, $j \in \{0, \ldots, s\}$, $r \in \{1, \ldots, m\}$, such that (see Proposition 3.1 and (3.16))

$$\phi_j - h\psi_j \mu_r = 0.$$

This cannot happen if $\mathrm{Re}(\phi_j/\psi_j)$ is positive.

We have verified that this is the case of P-circulant matrices for the BVMs introduced in section 2.1. Notice that for GBDF, this is a consequence of positive stability of P-circulants.

PROPOSITION 3.2. *If* $\mathrm{Re}(\mu_r) < -\delta$, $r = 1, \ldots, m$, *and* $\delta > 0$, *the block preconditioner* P *given by* (3.1) *which uses Strang's approximation* (3.2), (3.4) *is invertible, regardless of the stepsize* $h > 0$, *for all* $A_{\nu,k-\nu}$*-stable methods such that their boundary locus* $\rho(z)/\sigma(z)$, $|z| = 1$, *is a regular Jordan curve.*

Proof. Indeed, under the above hypotheses, we have

$$\mathrm{Re}(\rho(z)/\sigma(z)) \geq 0, \qquad |z| = 1$$

(see [3, Theorem 4.7.2]). By observing that $\mathrm{Re}(\rho(\epsilon^j)/\sigma(\epsilon^j))$, $j = 0, \ldots, s$, $\epsilon = e^{2\pi \mathbf{i}/(s+1)}$, is the ratio (3.19) when Strang's approximation is in use, it follows that $\phi_j - h\psi_j\mu_r$ cannot be zero. ∎

For more details on the boundary locus of an LMF, see [18] or [3].

Strang's approximation (3.4) for our block preconditioner was considered also in [9].

4. Using the preconditioner.

4.1. The computational cost. We have that the typical computational cost of the block circulant preconditioner (3.1) for an iterative Krylov subspace method is of the order of

$$(c_1 ms \log s + c_2\, c_3 \chi_1(J))\, n + c_3 \chi_2(J) \tag{4.1}$$

floating point operations. The c_i, $i = 1, 2$, are constants of moderate size, $c_3 \leq (s+1)$, n is the number of iterations of the iterative method, and $\chi_i(J)$, $i = 1, 2$, are suitable functions of the size and the structure of J, as explained in the sequel.

From (3.14), we can write P as

$$P = (F^* \otimes I_m) G (F \otimes I_m). \tag{4.2}$$

G is a $(s+1)$-block diagonal matrix with $m \times m$ diagonal blocks,

$$G_j = \phi_j I_m - h\psi_j\, J, \quad j = 0, \ldots, s, \tag{4.3}$$

that, under suitable assumptions (see, e.g., the previous section) are nonsingular matrices. Thus, to solve the linear systems whose matrices are given by (3.1), we need to apply FFTs of length $s+1$ to suitable permutations of the vector $\tilde{v}$ (asymptotic cost: $O(ms\log(s))$ operations). After this, we have to solve the $s+1$ linear systems whose matrices are given by (4.3). Denote with $\chi_2(J)$ the number of flops required for the factorization of G_j and with $\chi_1(J)$ the cost of the related back-substitution. We have, as an example, $\chi_i(J) = O(m)$, $i = 1, 2$, if J has a few nonzero diagonal entries (this is the case of the examples of section 5), or $\chi_2(J) = O(m^3)$, $\chi_1(J) = O(m^2)$, if J is dense and unstructured, etc. The decompositions should be computed only once and then stored.

If J is cheap enough to be diagonalized, and the condition number of V is moderate, where $D = V^{-1}JV$, the solution of the linear systems (2.6) can be found by direct inversion. Unfortunately, such hypotheses are relatively infrequent, especially for nonlinear problems. This is why we have not used the direct inversion for Examples 1 and 2 in section 5 even if possible (Example 3 has a nondiagonalizable Jacobian).

Notice that sometimes we do not need to solve all $s+1$ linear systems (4.3). Indeed, *after* the stepsize h has been chosen, we can check if (for those $j = 0, \ldots, s$ such that $\phi_j \neq 0$), the following condition is satisfied:

$$h \left| \frac{\psi_j}{\phi_j} \right| \|J\| < 1, \tag{4.4}$$

where ψ_j, ϕ_j are the eigenvalues of $\breve{B}$, $\breve{A}$, respectively. For those j such that (4.4) holds true, it can be observed that

$$G_j^{-1} \approx \phi_j^{-1} \left(I_m + h\frac{\psi_j}{\phi_j} J + h^2 \frac{\psi_j^2}{\phi_j^2} J^2 + \cdots + h^r \frac{\psi_j^r}{\phi_j^r} J^r \right), \tag{4.5}$$

where $r \geq 1$ is an integer and $\|\cdot\|$ is any norm such that $\|I\| = 1$. We have experienced that, truncating (4.5) to the linear term, we have a cheap and effective approximation for G_j^{-1}.

The condition (4.4) has proved useful also for stiff (nonlinear) problems. Indeed, often there are several subintervals of integration where the condition (4.4) is satisfied for some values of $j \in \{0, \ldots, s\}$. As an example, this is the case of van der Pol's equation

$$\begin{cases} y_1' = y_2, \\ y_2' = -y_1 + \mu y_2(1 - y_1^2), \quad t \in [0, \mu], \\ y_1(0) = 2, \qquad y_2(0) = 0. \end{cases} \tag{4.6}$$

If μ is large, (4.6) is stiff (see, e.g., [13]). However, we have experienced (up to $\mu = 1000$) that the condition (4.4) is satisfied in several mesh intervals for most of the indexes j (except in the layer regions when μ is large). Notice that $R_j = |\psi_j/\phi_j|$ is less than 1 for the BVMs considered here for several j, $j \in \{0, \ldots, s\}$. See Figure 4.1.

The matrices G_j occur in conjugate pairs, and several other ideas are possible to exploit their special structure; see [13, chapter IV.8].

The computational cost of the underlying block circulant preconditioners can be reduced defining ad hoc methods for differential equations. Indeed, the coefficients α_j, β_j, $j = 0, \ldots, k$, for the LMF (2.1) can be chosen such that the following ratio is

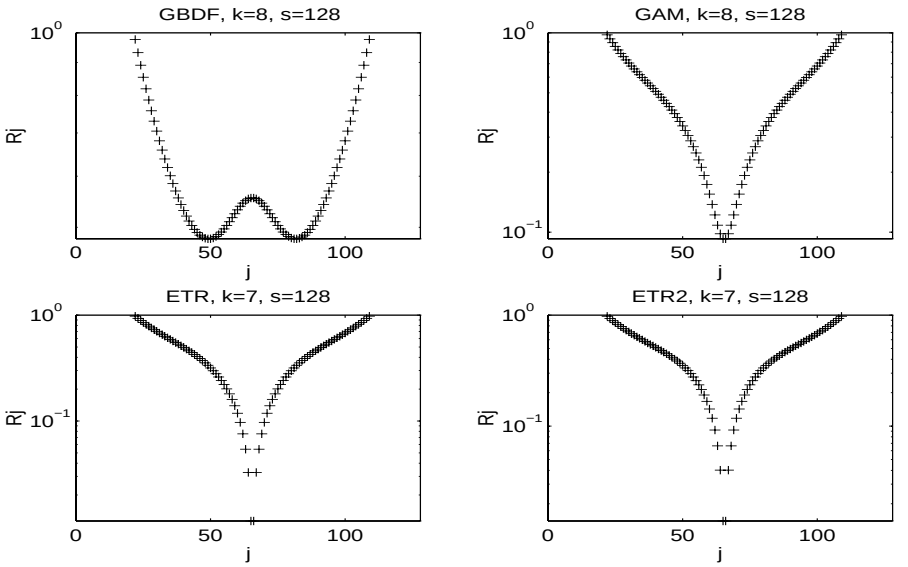


FIG. 4.1. $R_j = |\psi_j/\phi_j|$, $j = 0, \ldots, s$, *for* $k = 7, 8$ *(GBDF, GAM, ETR, ETR$_2$ formulas).*

kept constant:

$$\frac{\psi_j}{\phi_j} = \lambda, \ j = 0, \ldots, s,$$

where ϕ_j, ψ_j are, respectively, the eigenvalues of the P-circulant matrices $\breve{A}$, $\breve{B}$. Using those methods, we need only one factorization of an $m \times m$ matrix which has (almost) the same sparsity pattern of the Jacobian matrix of the underlying ODE. Notice that this process has some analogy with the diagonally implicit Runge–Kutta methods (see [13]).

An alternative approach to the direct solution of the linear systems (4.3) can be effective for particular problems. This is the case of the systems of time-dependent partial differential equations; see [14] and references therein. We will pursue a similar approach for our block preconditioner in a future work.

Finally, the block preconditioner is inherently parallel. The $s + 1$ linear systems (4.3) are independent and can be solved in parallel.

4.2. Convergence of the preconditioned iterations. We expect fast convergence of preconditioned iterations if the spectrum of the block preconditioned matrix is clustered around $(1, 0) \in \mathbb{C}$. To this end, notice that $P^{-1}M$, where P is the block P-circulant or the block Strang's preconditioner, can be written as the sum of the identity, a low rank and a small norm matrix.

THEOREM 4.1. *Let M be the matrix of the linear system* (2.6) *for the schemes described in section* 2.1, *and let P be its block P-circulant preconditioner as* (3.1). *Then, for fixed $\delta > 0$, there exist $C_\delta \geq 0$, $s_\delta \geq k$ such that, for all $s \geq s_\delta$ ($s+1$ is the size of A, B),*

$$P^{-1} M = I + M_\delta^{(1)} + M_\delta^{(2)}, \tag{4.7}$$

where $\operatorname{rank}(M_\delta^{(2)}) \le m\,[2(k+1)+C_\delta]$ *and* $\|M_\delta^{(1)}\|_2 \le \delta\, c_J$, *where* c_J *does not depend on* s.

If P *is defined as Strang's circulant,* $C_\delta = \|M_\delta^{(1)}\| = 0$.

Proof. Let $E = M - P$,

$$\begin{aligned} E &= (A - \breve{A}) \otimes I_m - h(B - \breve{B}) \otimes J \\ &= E_A \otimes I_m - h\, E_B \otimes J \\ &= (E_A^{(1)} + E_A^{(2)}) \otimes I_m - h(E_B^{(1)} + E_B^{(2)}) \otimes J. \end{aligned} \tag{4.8}$$

Here $E_A^{(1)}$ is an $(s+1) \times (s+1)$ Toeplitz matrix with bandwidth $m(k+1)$ defined from the coefficients α_j of the main method (2.1)

$$E_A^{(1)} = (e_{A,rl}^{(1)}), \quad e_{A,rl}^{(1)} = e_{A,r-l}^{(1)} = e_{A,j}^{(1)}, \quad j = -s, \dots, s,$$

and, for P-circulants and optimal circulants,

$$e_{A,j}^{(1)} = \begin{cases} \dfrac{-j}{s+1}\alpha_{j+\nu}, & j = -\nu, \dots, k-\nu \ \text{ (P-circulant)}, \\ \dfrac{|j|}{s+1}\alpha_{j+\nu}, & j = -\nu, \dots, k-\nu \ \text{ (optimal circulant)}, \\ 0 & \text{otherwise.} \end{cases} \tag{4.9}$$

The entries of $E_B^{(1)}$ are defined similarly to (4.9), but with β_j instead of α_j. For Strang's natural circulant it is easy to check that $E_A^{(1)}$, $E_B^{(1)}$ are the null matrix; see (3.2), (3.4).

From the previous equations, we have

$$\begin{aligned} P^{-1}M = I + P^{-1}E &= I + P^{-1}(E_A^{(1)} \otimes I_m - h\, E_B^{(1)} \otimes J) \\ &\quad + P^{-1}(E_A^{(2)} \otimes I_m - h\, E_B^{(2)} \otimes J) = I + \tilde{M}^{(1)} + \tilde{M}^{(2)}, \end{aligned} \tag{4.10}$$

where $E_A^{(2)}$, $E_B^{(2)}$ are $(s+1) \times (s+1)$ matrices whose entries are nonzero at most in the following four corners:

1. $\nu \times (k+1)$ in the upper left;
2. $\nu \times \nu$, upper right;
3. $(k-\nu) \times (k+1)$, lower right;
4. $(k-\nu) \times (k-\nu)$, lower left.

From the above arguments, we have that

$$\operatorname{rank}(E_A^{(2)}) \le k+1, \qquad \operatorname{rank}(E_B^{(2)}) \le k+1,$$

and, if P^{-1} is well defined,

$$\operatorname{rank}(\tilde{M}^{(2)}) \le 2m\,(k+1),$$

independently on the size of $\breve{A}$, $\breve{B}$. Thus, if $\breve{A}$, $\breve{B}$ are the Strang's circulants as (3.2), (3.4), the thesis follows by setting $M_\delta^{(2)} = \tilde{M}^{(2)}$ and by observing that $C_\delta = \|M_\delta^{(1)}\| = 0$.

Let us consider $\tilde{M}^{(1)}$ when $\breve{A}$, $\breve{B}$ are P-circulant matrices. From (4.10) we have

$$\tilde{M}^{(1)} = \frac{1}{s+1} P^{-1} \left(\hat{E}_A^{(1)} \otimes I_m - h\, \hat{E}_B^{(1)} \otimes J \right), \tag{4.11}$$

where $\hat{E}_A^{(1)} = (s+1)E_A^{(1)}$, $\hat{E}_B^{(1)} = (s+1)E_B^{(1)}$ are banded Toeplitz matrices whose infinity norm is constant with respect to their dimension $s+1$.

We are interested in the behavior of $P^{-1}M$ for $s \to \infty$ (and then for $h \to 0$, $h = (T - t_0)/s$; see section 2.1). To this end, for the spectral properties of the P-circulant matrices $\breve{A}$, $\breve{B}$ (see [2] and section 3.1), $\tilde{M}^{(1)}$ in (4.10) can be written as the sum of a small rank and a small $l2$-norm matrix. Indeed, as in the proofs of [21, Theorems 3.2, 3.4], let us fix $\delta > 0$ and consider the cardinality C_δ of the set

$$\left\{ l \in \{0, \ldots, s\} : |\lambda_l(\breve{A})| \leq \frac{1}{\delta} \frac{1}{s+1} \right\},$$

where $\lambda_l(\breve{A})$ are the $s+1$ distinct eigenvalues of $\breve{A}$. For (3.8), $\lambda_l(\breve{A}) = \phi_k(x_l)$, $x_l = 2\pi l/(s+1)$, $l = 0, \ldots, s$, where

$$\phi_k(x) = \sum_{j=-\nu}^{k-\nu} \alpha_{j+\nu} \left(1 + \frac{j}{s+1} \right) e^{\mathbf{i} j x}. \tag{4.12}$$

The expression $\phi_k(x)$ is a trigonometric polynomial and, as observed in [2], for the formulas in (2.1),

$$\lim_{s\to\infty} \phi_k(x_0) = \lim_{s\to\infty} \phi_k(0) = 0,\ \phi_k'(x_0) \neq 0, \qquad \lim_{s\to\infty} \phi_k'(x_0) \neq 0, \tag{4.13}$$

i.e., x_0 is a simple zero for $\lim_{s\to\infty} \phi_k(x)$ and

$$\frac{c_1}{s+1} \leq |\phi_k(x)| \leq c_2, \qquad \mathrm{Re}\,(\phi_k(x)) > 0, \qquad x \in \mathbb{R}, \tag{4.14}$$

where c_1, c_2 are constants of the order of unity. The matrix $\breve{A}$ is normal (because it is circulant), thus $|\lambda_l(\breve{A})| = |\phi_k(x_l)|$ is a singular value. Moreover, for (4.13), (4.14), and (4.12), if l is small with respect to s,

$$|\phi_k(x_l)| \geq \hat{c} \frac{l}{s+1},$$

and, as a consequence,

$$C_\delta \leq \# \left\{ l : \hat{c} \frac{l}{s+1} \leq \frac{1}{\delta} \frac{1}{s+1} \right\} \leq \left\lceil \frac{1}{\delta\, \hat{c}} \right\rceil, \tag{4.15}$$

which does not depend on s since $\hat{c}$ is a constant.

Recalling that the circulant matrices are simultaneously diagonalized by the matrix F in (3.13) (see [10]), consider the following splitting:

$$\frac{1}{s+1} \breve{A}^{-1} = \hat{\Theta}_1 + \hat{\Theta}_2, \tag{4.16}$$

where the $s+1$ singular values of the circulant matrices $\hat{\Theta}_2$, $\hat{\Theta}_1$ are

$$\{a_0, \ldots, a_{C_\delta - 1},\ 0, \ldots, 0\}, \qquad \{0, \ldots, 0, a_{C_\delta}, \ldots, a_s\}, \tag{4.17}$$

respectively, and $\delta < a_j$, $j = 0, \ldots, C_\delta - 1$; $a_j < \delta$, $j = C_\delta, \ldots, s$, $a_j \geq a_i \geq 0$ if $0 \leq j < i \leq s$.

Thus, from (4.11) and (4.16) we can set $\tilde{M}^{(1)}$ as

$$\tilde{M}^{(1)} = \left(I_{s+1} \otimes I_m - h\breve{A}^{-1}\breve{B} \otimes J\right)^{-1} \left((\hat{\Theta}_1 + \hat{\Theta}_2) \otimes I_m\right) \left(\hat{E}_A^{(1)} \otimes I_m - h\hat{E}_B^{(1)} \otimes J\right),$$

i.e., we can split the above matrix as

$$\tilde{M}^{(1)} = M_\delta^{(1)} + \tilde{M}_\delta^{(2)},$$

where the products containing $\hat{\Theta}_2$ are collected in $\tilde{M}_\delta^{(2)}$, while those containing $\hat{\Theta}_1$ are in $M_\delta^{(1)}$. As a consequence of the above arguments and of (4.10), (4.11), fixed $\delta > 0$, we can find s_δ such that, for all $s \geq s_\delta$,

$$\tilde{M}^{(1)} = M_\delta^{(1)} + \tilde{M}_\delta^{(2)}, \quad \operatorname{rank}(\tilde{M}_\delta^{(2)}) \leq m\, C_\delta, \quad \|\hat{\Theta}_1 \hat{E}_A^{(1)}\|_2 \leq \delta\, c, \quad \|\hat{\Theta}_1 \hat{E}_B^{(1)}\|_2 \leq \delta\, c,$$

where c does not depend on s and usually is of the order of unity. If we define $M_\delta^{(2)} = \tilde{M}^{(2)} + \tilde{M}_\delta^{(2)}$, the first half of the thesis follows. To check that $M_\delta^{(1)}$ is a small norm matrix, we can transform $M_\delta^{(1)}$ by the matrices $(F \otimes V^{-1})$, $(F^* \otimes V)$, $J = VDV^{-1}$. (We suppose for simplicity that the Jacobian matrix J is diagonalizable.) Thus, if $N = (F \otimes V^{-1}) M_\delta^{(1)} (F^* \otimes V)$, we have

$$\begin{aligned} N &= \left(I_{s+1} \otimes I_m - h\Lambda_A^{-1}\Lambda_B \otimes D\right)^{-1} (F \otimes I_m) \\ &\quad \cdot (\hat{\Theta}_1 \otimes I_m) \left(\hat{E}_A^{(1)} \otimes I_m - h\hat{E}_B^{(1)} \otimes D\right) (F^* \otimes I_m) \\ &= \left(I_{s+1} \otimes I_m - h\Lambda_A^{-1}\Lambda_B \otimes D\right)^{-1} (\Lambda_1\, F \otimes I_m) \\ &\quad \cdot \left(\hat{E}_A^{(1)} \otimes I_m - h\hat{E}_B^{(1)} \otimes D\right) (F^* \otimes I_m), \end{aligned} \tag{4.18}$$

where $\hat{\Theta}_1 = F^*\Lambda_1 F$, $\breve{A} = F^*\Lambda_A F$, $\breve{B} = F^*\Lambda_B F$, and Λ_1, Λ_A, Λ_B are diagonal matrices. From (4.18) and the above arguments we have the following bound:

$$\begin{aligned} \|N\|_2 &\leq \|\left(I_{s+1} \otimes I_m - h\Lambda_A^{-1}\Lambda_B \otimes D\right)^{-1}\|_2 \|F\hat{\Theta}_1 E_A^{(1)} F^* \otimes I_m\|_2 \\ &\quad + \max_r \|(I_{s+1} \otimes I_m - h\mu_r \Lambda_A^{-1}\Lambda_B)^{-1}(h\mu_r \Lambda_1\, F\, \hat{E}_B^{(1)} F^*)\|_2 \\ &\leq \frac{\|F\hat{\Theta}_1 \hat{E}_A^{(1)} F^*\|}{\min_{j,r}\left|1 + h(-\mu_r)\dfrac{\psi_j}{\phi_j}\right|} + \frac{\|\Lambda_1 F \hat{E}_B^{(1)} F^*\|_2}{\min_{j,r}\left|\dfrac{1}{h(-\mu_r)} + \dfrac{\psi_j}{\phi_j}\right|} \\ &\leq \frac{\|\hat{\Theta}_1 \hat{E}_A^{(1)}\|_2}{c_{J,1}} + \frac{\|\hat{\Theta}_1 \hat{E}_B^{(1)}\|_2}{c_{J,2}} \leq \delta\, \hat{c}_J. \end{aligned} \tag{4.19}$$

Excluding the trivial case $\mu_r = 0$, we have that $\operatorname{Re}(1/(h(-\mu_r))) \geq 0$ if the Jacobian matrix J has eigenvalues whose real parts are nonpositive, and $c_{J,1}$, $c_{J,2}$ in the above expression can be bounded uniformly in s if $\operatorname{Re}(\psi_j/\phi_j) \geq \epsilon > 0$.We have verified that

this holds, e.g., for the methods of section 2.1. Thus, $\hat{c}_J$, c_J and then the bound for $\|M_\delta^{(1)}\|_2$ are independent from s.

Notice that when $h\,\mu_r$ is small, such as in the nonstiff case, we have simply that

$$\|M_\delta^{(1)}\|_2 \le c_2 \|\hat{\Theta}_1 \hat{E}_A^{(1)}\|_2 \le \delta\, c\, c_2,$$

where c_2 is a constant. □

Unfortunately, we cannot state a similar result for the block preconditioner (3.1) based on the optimal circulants because of the ill-conditioning arising when k is large (see [2]). Nonetheless, if (4.14) hold true, for fixed k suitably small, similar arguments to those used for P-circulants can be used to prove convergence.

Despite the fact that $\|M_\delta^{(1)}\|$ and C_δ are zero for P given by (3.1) using Strang's approximations, we have observed that such a preconditioner is often less suitable than others. Indeed, as an example, the Strang's circulants can be severely ill-conditioned or even singular, e.g., for problems such that some of the eigenvalues of their Jacobian matrix have zero or small modulus real part (or imaginary part large in absolute value with respect to the real part). Unfortunately, this is not infrequent, e.g., for stiff problems; see Example 1 and the remarks in the next section.

5. Numerical results. To show the effectiveness of our block preconditioner, we will integrate some test problems with the formulas (2.9), a generalization of Adams–Moulton methods. Such formulas have been found to be quite effective (see [17]).

Further numerical examples using other formulas can be found in [1].

We will compare the number of matrix-vector products needed to converge for Bi-CGSTAB [25] (Bi-CGSTAB(2) [22] for Example 1 because Bi-CGSTAB has shown much more erratic unpreconditioned convergence) and GMRES [20].

The initial guess for the iterative solvers is always zero. The stopping criterion is $\|r_k\| < 10^{-6}\|b\|$, r_k true residual. The stability properties of the LMF considered here (see [3]) allow an application without stepsize restriction. Hence, we will use a constant stepsize h. All calculations are done in Matlab.

In the columns labeled I_s, P_s, C_s and S_s in the tables below, we will give the number of matrix-vector products needed to the convergence of the unpreconditioned iterations, preconditioned iterations using P-circulants, and Chan's and Strang's approximations, respectively. A "−" means that convergence was not attained.

The column labeled "rfp" gives a rough estimate of the ratio of the floating points operations required for, respectively, the P-circulant preconditioned and the unpreconditioned GMRES.

Example 1. *Wave equation.*

Let us consider the wave equation

$$\text{(5.1)} \qquad \begin{cases} u_{tt} - c u_{xx} = 0, & \\ u(x,0) = g_1(x),\ u_t(x,0) = g_2(x), & x \in [0,\pi], \\ u(0,t) = u(\pi,t) = 0, & t \in [0,T]. \end{cases}$$

Approximating the operator $\partial^2/\partial x^2$ in (5.1) with centered differences and reducing the obtained IVP to the first order gives the system of $2N$ ODEs:

$$\text{(5.2)} \qquad \begin{cases} y'(t) = H_{2N} y(t), & t \in [0,T], \\ y(0) = \eta, & \eta = (g_1(x_1)\ \cdots\ g_1(x_N) g_2(x_{N+1})\ \cdots\ g_2(x_{2N}))^T. \end{cases}$$

TABLE 5.1
Wave equation (5.1). *Number of matrix-vector products, GAM with* $k = 3$.

N	s	h	GMRES				BiCGstab(2)				rfp
			I_s	P_s	C_s	S_s	I_s	P_s	C_s	S_s	
20	8	$\pi/4$	298	31	32	31	>1000	40	44	64	.4e-1
50	8	$\pi/4$	776	34	35	45	-	45	52	81	.7e-3
100	8	$\pi/4$	>1000	34	37	71	-	48	52	109	.2e-3
20	16	$\pi/8$	311	36	38	40	609	45	49	104	.5e-1
50	16	$\pi/8$	>1000	42	46	62	-	52	60	206	.6e-4
100	16	$\pi/8$	>1000	42	45	60	-	52	56	197	-
20	32	$\pi/16$	203	35	37	42	744	44	49	125	.1
50	32	$\pi/16$	764	44	48	66	-	49	60	333	.1e-1
100	32	$\pi/16$	>1000	45	50	80	-	52	60	588	-

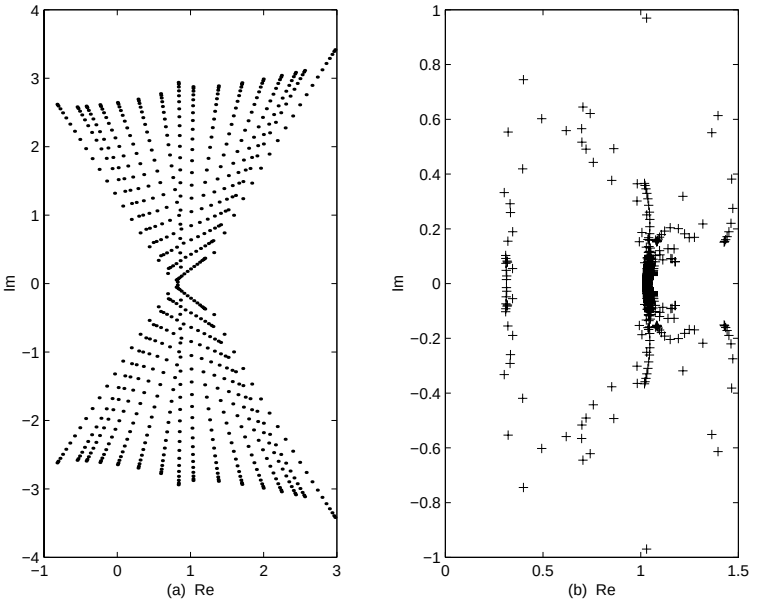


FIG. 5.1. *Wave equation* (5.1). *Spectrum of eigenvalues of the matrix* M (2.6) *before and after P-circulant preconditioning.* ($c = 1$, *GAM with* $k = 3$, $s = 16$, $N = 20$, $T = 2\pi$.)

The matrix H_{2N} is a Hamiltonian definite one:

$$H_{2N} = \begin{pmatrix} \mathbf{0} & I_N \\ T_N & \mathbf{0} \end{pmatrix}_{2N \times 2N}. \tag{5.3}$$

The matrix H_{2N} has the spectrum of eigenvalues on the imaginary axis. In Table 5.1, we can see the effect of the block preconditioner on the number of iterations needed to solve the IVP (5.2) using the fourth order GAM ($k = 3$), $T = 2\pi$, $g_1(x) = 0$, $g_2(x) = x$. In Figures 5.1 and 5.2 we can see the effect on the spectrum of eigenvalues of the matrix M.

Notice that the block preconditioner based on the Strang's circulants does not show the best preconditioned behavior, despite the fact that $M_\delta^{(1)}$, the small norm matrix in (4.7), is zero; see also Table 5.2. To this end, see also the remarks at the end of this section.

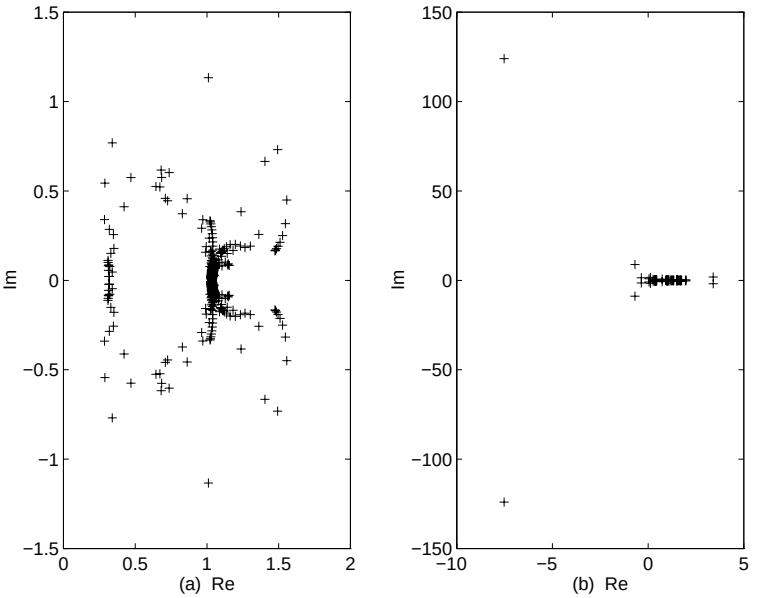


FIG. 5.2. *Wave equation* (5.1). *Spectrum of eigenvalues of the matrix* $P^{-1}M$ *as in Figure* 5.1(b) *but after block preconditioning using* (a) *Chan's and* (b) *Strang's circulants.*

TABLE 5.2
Wave equation (5.1)*, same parameters as Table* 5.1. *Comparison of the computational costs related to the block preconditioner using Strang's approximations.*

			GMRES			BiCGstab(2)		
N	s	size	P_s	C_s	S_s	P_s	C_s	S_s
20	8	320	1	1.04	1	.63	.7	1
50	8	800	.71	.73	1	.57	.66	1
100	8	1600	.4	.43	1	.45	.49	1
20	16	640	.87	.94	1	.43	.48	1
50	16	1600	.6	.68	1	.24	.28	1
100	16	3200	.27	.29	1	.63	.69	1
20	32	1280	.8	.85	1	.35	.39	1
50	32	3200	.15	.188	1	.59	.66	1
100	32	6400	.089	.1	1	.47	.57	1

Example 2. *Heat equation in a rectangle.*

Consider the two-dimensional heat equation defined in a rectangular domain:

$$\text{(5.4)} \qquad \begin{cases} u_t - (u_{xx} + u_{yy}) = 0, & (x,y) \in \Omega = [0,\pi] \times [0,\pi], \\ u((x,y),0) = g(x,y), & (x,y) \in \Omega, \\ u((x,y),t) = 0, & (x,y) \in \partial\Omega,\ 0 \le t \le T. \end{cases}$$

Using centered differences to approximate the Laplacian operator in the rectangle Ω with a uniform grid

$$\Delta x = \Delta y = \Delta, \quad \Delta = \pi/(N+1)$$

gives the IVP

$$\text{(5.5)} \qquad \begin{cases} y'(t) = \dfrac{1}{\Delta^2} \hat{L}_N y(t), & t \in [0,T], \\ y(0) = \eta, & \eta = (g(x_1,y_1)), \ldots, g(x_N,y_N)^T, \end{cases}$$

TABLE 5.3
Heat equation (5.4). *Number of matrix-vector products, GAM with* $k = 4$.

			GMRES				BiCGstab				
N	s	h	I_s	P_s	C_s	S_s	I_s	P_s	C_s	S_s	rfp
4	8	$\pi/4$	51	8	8	7	74	14	14	12	.3
8	8	$\pi/4$	121	8	8	7	216	14	14	12	.4
20	8	$\pi/4$	368	7	7	6	631	14	16	12	.5e-1
4	16	$\pi/8$	53	7	7	6	72	12	10	10	.3
8	16	$\pi/8$	126	7	7	6	175	12	10	10	.5
20	16	$\pi/8$	376	6	6	6	648	12	10	10	.5e-1
4	24	$\pi/12$	45	7	7	7	61	10	10	10	.6
8	24	$\pi/12$	107	7	7	7	154	12	12	10	.2
20	24	$\pi/12$	320	6	6	6	518	12	12	10	.7e-1

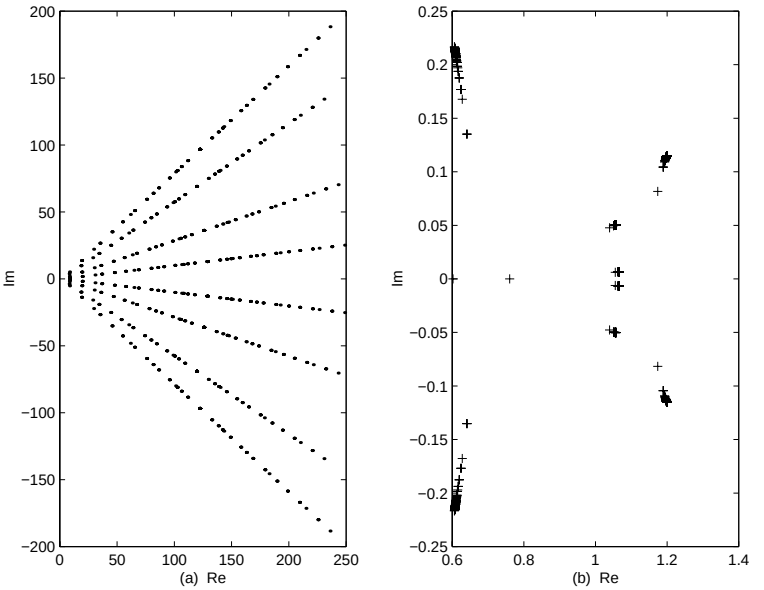


FIG. 5.3. *Heat equation* (5.4). *Spectrum of the eigenvalues of the matrix* M (2.6) *before and after P-circulant block preconditioning. (GAM,* $k = 4$, $s = 16$, $N = 20$.*)*

where $\hat{L}_N$ is a $N^2 \times N^2$ block tridiagonal matrix:

$$\hat{L}_N = \begin{pmatrix} \hat{T}_N & I_N & & \\ I_N & \ddots & \ddots & \\ & \ddots & \ddots & I_N \\ & & I_N & \hat{T}_N \end{pmatrix}, \qquad \hat{T}_N = \begin{pmatrix} -4 & 1 & & \\ 1 & \ddots & \ddots & \\ & \ddots & \ddots & 1 \\ & & 1 & -4 \end{pmatrix} \tag{5.6}$$

and x_i, y_j, $i, j = 1, \ldots, N$, in (5.5) are defined accordingly. In Table 5.3, we can see the effect of the block preconditioner on the number of iterations needed to solve the IVP (5.5) using fifth order GAM ($k = 4$), $T = 2\pi$, $g(x, y) = x\,y$. In Figure 5.3 we can see the effect on the spectrum of eigenvalues of the matrix M.

As an example, notice that the built-in Matlab sparse direct solver for the linear system (2.6) used for Example 2 with $N = 20$, $s = 8$ ($N = 20$, $s = 16$) needs 6

(18) times more flops than the P-circulant block preconditioned GMRES. In general, we can save flops provided that M in (2.6) is large enough and the convergence of preconditioned iterations is achieved in a moderate number of iterations.

Example 3. *Wave equation of first order.*

$$\left\{\begin{array}{ll} u_t - u_x = 0, & \\ u(x,0) = g(x), & x \in [0,\pi], \\ u(\pi,t) = 0, & t \in [0,2\pi]. \end{array}\right. \tag{5.7}$$

We discretize the partial derivative $\partial/\partial x$ with the first order forward difference and stepsize $\Delta x = \pi/N$, $x_j = j\Delta x$ (upwind discretization). We obtain the system of N ODEs

$$\left\{\begin{array}{ll} y'(t) = L_N y(t), & t \in [0,2\pi], \\ y(0) = \eta, & \eta = (g(x_0) \ \cdots \ g(x_{N-1}))^T, \end{array}\right. \tag{5.8}$$

$$L_N = \frac{1}{\Delta x}\begin{pmatrix} -1 & 1 & & \\ & \ddots & \ddots & \\ & & \ddots & 1 \\ & & & -1 \end{pmatrix}_{N\times N}.$$

In Table 5.4, we can see the effect of the block preconditioner on the number of iterations needed to solve the IVP (5.7) using fifth order GAM ($k = 4$), $T = 2\pi$, $g(x) = \sqrt{x(\pi - x)}$. In Figure 5.4 we can see the effect on the spectrum of eigenvalues of the matrix M.

As expected, the block preconditioner (3.1) is effective for those classes of differential problems for which the preconditioned matrix $P^{-1}M$, M in (2.6), has a clustered spectrum. For Theorem 4.1, this is true if $M_\delta^{(1)}$ in (4.7) either is zero (as is the case of Strang's circulants) or is a suitable perturbation of the null matrix (e.g., has eigenvalues clustered around the origin of the complex plane, as is the case of P-circulants) *and* P is not ill-conditioned. Moreover, the small rank matrix $M_\delta^{(2)}$ should have few and possibly clustered outliers (i.e., the eigenvalues outside the multiple eigenvalue in the origin of $\mathbb{C}$).

Notice that the above properties may not hold if the preconditioner is ill-conditioned, e.g., because $\breve{A}$ or $\breve{B}$ are ill-conditioned or even singular. As an example, see Example 1 for the block preconditioner using Strang's or Chan's approximations. A similar behavior can be observed for Example 2 if a fast decaying diffusion coefficient is considered. The effect of the ill-conditioning on the convergence of preconditioned iterations can be reduced significantly if special initial conditions for the underlying partial differential equations are considered; see the numerical tests in [9]. More details and analysis of the rate of convergence of preconditioned iterations will be given in a forthcoming paper.

In this section, we have considered methods based on the formula (2.9). In general, the same behavior can be observed also for the other formulas of section 2.1, e.g., (2.10) for Example 1 and (2.8) for Examples 2 and 3.

6. Concluding remarks. In this work we have considered a block circulant preconditioner we introduced in [1] for the linear systems of certain codes for ordinary differential equations. Such preconditioners are used here implicitly with iterative Krylov subspace methods for nonsymmetric systems such as Bi-CGSTAB, Bi-CGSTAB(2) and GMRES.

TABLE 5.4
Wave equation of first order (5.7). *Number of matrix-vector products, GAM with* $k = 4$.

			GMRES				BiCGstab				
N	s	h	I_s	P_s	C_s	S_s	I_s	P_s	C_s	S_s	rfp
20	8	$\pi/4$	38	10	9	10	88	10	10	10	.2
50	8	$\pi/4$	100	12	11	12	-	12	12	13	.7e-1
100	8	$\pi/4$	239	13	12	13	-	18	16	18	.1e-1
20	16	$\pi/8$	28	9	8	8	49	10	10	10	.7
50	16	$\pi/8$	87	10	9	9	-	12	12	12	.7e-1
100	16	$\pi/8$	222	10	10	10	-	12	14	12	.1e-1
20	32	$\pi/16$	36	7	7	7	41	8	8	9	.3
50	32	$\pi/16$	70	8	8	8	179	9	9	9	.1
100	32	$\pi/16$	176	9	9	9	-	10	12	10	.2e-1

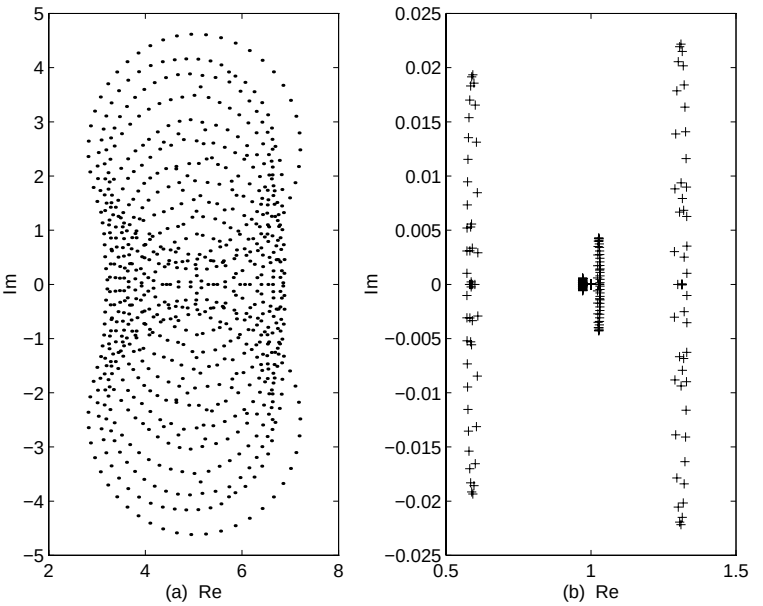


FIG. 5.4. *Wave equation of first order* (5.7). *Spectrum of the eigenvalues of the matrix* M (2.6) *before and after P-circulant preconditioning. (GAM,* $k = 3$, $s = 16$, $N = 20$.*)*

It has been observed that, for some classes of differential problems, the preconditioned iterations are almost independent from the discretization.

Moreover, a recently introduced circulant approximation, called P-circulant, has been found to be promising in comparison to optimal approximations, in the sense of the norm (see, e.g., [8, p. 432–434]), such as Strang's and Chan's. This has been confirmed by the analysis in [2].

Despite the fast convergence, the above block preconditioner has a moderate theoretical parallel complexity and an interesting serial computational cost for the classes of differential problems such that the spectrum of the block preconditioned matrix $P^{-1}M$ is clustered.

Acknowledgments. The author would like to thank both the referees and Michele Benzi of the Los Alamos National Laboratory for very helpful comments that have improved this presentation. The author is grateful to Luigi Brugnano and Donato Trigiante of the University of Firenze for their suggestions and constructive criticism.

REFERENCES

[1] D. Bertaccini, *P-circulant preconditioners and the systems of ODE codes*, in Iterative Methods in Scientific Computation IV, IMACS Ser. Comput. Appl. Math., D. R. Kincaid and A. C. Elster, eds, New Brunswick, NJ, 1999, pp. 179–193.

[2] D. Bertaccini, *An Analysis of Circulant Preconditioners for Linear Multistep Integrators*, Math. Comp., submitted.

[3] L. Brugnano and D. Trigiante, *Solving ODE by Linear Multistep Methods: Initial and Boundary Value Methods*, Gordon and Breach, Reading, UK, 1998.

[4] T. F. Chan and K. R. Jackson, *The use of iterative linear-equation solvers in codes for large systems of stiff IVPs for ODEs*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 378–417.

[5] T. F. Chan, *An optimal circulant preconditioner for Toeplitz systems*, SIAM J. Sci. Statist. Comput., 9 (1988), pp. 766–771.

[6] R. H. Chan, *Toeplitz preconditioners for Toeplitz systems with nonnegative generating functions*, IMA J. Numer. Anal., 11 (1991), pp. 333–345.

[7] R. H. Chan and M.-C. Yeung, *Circulant preconditioners for complex Toeplitz matrices*, SIAM J. Numer. Anal., 30 (1993), pp. 1193–1207.

[8] R. H. Chan and M. K. Ng, *Conjugate gradient methods for Toeplitz systems*, SIAM Rev., 38 (1996), pp. 427–482.

[9] R. H. Chan, M. K. Ng, and X. Jin, *Circulant preconditioners for solving ordinary differential equations*, in Structured Matrices, D. Bini, E. Tyrtyshnikov, and P. Yalamov, eds., Nova Science, to appear.

[10] P. J. Davis, *Circulant Matrices*, John Wiley, New York, 1979.

[11] F. Di Benedetto, G. Fiorentino, and S. Serra, *C.G. preconditioning for Toeplitz matrices*, Comput. Math. Appl., 25 (1993), pp. 33–45.

[12] C. W. Gear and Y. Saad, *Iterative solution of linear equations in ODE codes*, SIAM J. Sci. Statist. Comput., 4 (1983), pp. 583–601.

[13] E. Hairer and G. Wanner, *Solving Ordinary Differential Equations* II. *Stiff and Differential-Algebraic Problems*, Springer-Verlag, Berlin, Heidelberg, 1991.

[14] S. Holmgreen and K. Otto, *A framework for polynomial preconditioners based on fast transforms* I: *Theory*, BIT, 38 (1998), pp. 544–559.

[15] R. A. Horn and C. R. Johnson, *Topics in Matrix Analysis*, Cambridge University Press, Cambridge, UK, 1994.

[16] T. Huckle, *Some aspects of circulant preconditioners*, SIAM J. Sci. Comput., 14 (1993), pp. 531–541.

[17] F. Iavernaro and F. Mazzia, *Solving ordinary differential equations by generalized Adams methods: Properties and implementations techniques*, Appl. Numer. Math. 28 (1198), pp. 107–126.

[18] J. D. Lambert, *Numerical Methods for Ordinary Differential Systems*, John Wiley, Chichester, 1991.

[19] P. Lancaster and M. Tismenetsky, *The Theory of Matrices*, Comput. Sci. Appl. Math., Academic Press, New York, 1985.

[20] Y. Saad and M. H. Schultz, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 856–869.

[21] S. Serra, *The rate of convergence of Toeplitz based PCG methods for second order nonlinear boundary value problems*, Numer. Math., 81 (1999), pp. 461–495.

[22] G. L. G. Sleijpen, H. A. van der Vorst, and D. R. Fokkema, *BiCGstab(l) and other hybrid Bi-CG methods*, Numer. Algor., 7 (1994), pp. 75–109.

[23] G. Strang, *A proposal for Toeplitz matrix calculations*, Stud. Appl. Math., 74, (1986), pp. 171–176.

[24] V. V. Strela and E. E. Tyrtyshnikov, *Which circulant preconditioner is better?*, Math. Comp., 65 (1996), pp. 137–150.

[25] H. A. van der Vorst, *Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 10 (1992), pp. 631–644.

SIAM J. SCI. COMPUT.
Vol. 22, No. 3, pp. 787–801
© 2000 Society for Industrial and Applied Mathematics

NUMERICAL SHADOWING USING COMPONENTWISE BOUNDS AND A SHARPER FIXED POINT RESULT*

ERIK S. VAN VLECK†

Abstract. Shadowing provides a means of obtaining global error bounds for approximate solutions of nonlinear differential equations with interesting dynamics, in particular, for initial value problems with positive Lyapunov exponents. Shadowing breaks down in the presence of zero Lyapunov exponents, although some results such as shadowing with rescaling of time have been obtained. Using a reformulation of the original differential equations and an improved fixed point result we take advantage of componentwise local error bounds to use relatively smaller error tolerances in nonhyperbolic and contractive directions (i.e., in directions corresponding to zero and negative Lyapunov exponents). The result is a decrease in the shadowing global error.

Key words. shadowing lemma, hyperbolicity, global error estimation

AMS subject classification. 65L

PII. S1064827599353452

1. Introduction.

Global error analysis using a shadowing lemma approach involves treating the error equation as a boundary value problem (BVP) as opposed to an initial value problem (IVP). This allows for shadowing-type error analysis results for problems with expansive and contractive directions. The difficulty occurs when there are directions that are neither exponentially expansive nor contractive, and we call these nonhyperbolic directions. Most shadowing lemma techniques rely on a fixed point result that is similar to using the numerically computed solution as an initial guess for a modified Newton's method to find a true trajectory to the given differential equation. This leads to the need to find and bound an inverse (in our case a right inverse) of the Jacobian for the modified Newton's method.

In this paper, we consider an improved fixed point result and find that what is important is not so much the norm of a right inverse of the Jacobian, but the action of the right inverse on several vectors. This produces sharper results and the ability to gain a better understanding of the breakdown of shadowing in the presence of nonhyperbolic directions. We reformulate the original differential equation in terms of a new coordinate system that corresponds to the directions associated with the Lyapunov exponents of the linearized problem that we denote by the columns of an orthogonal matrix function $Q \equiv Q(t)$. This allows us to write the dependent variable x of the nonlinear differential equation in terms of these directions of different growth and decay rates and write $x = Qy$ and solve for Q and y instead of x. With this reformulation we derive componentwise bounds on the magnitude of the local error in directions corresponding to different rates of growth and decay, and with the improved fixed point result we are able to derive sharper shadowing bounds by employing smaller local error tolerances in the nonhyperbolic and contractive directions.

Our focus numerically is in employing existing IVP solvers, including the extrapolation code `ODEX` from [17] and the Runge–Kutta suite `RKSUITE` from [3]. The

*Received by the editors March 22, 1999; accepted for publication (in revised form) February 21, 2000; published electronically August 31, 2000.
http://www.siam.org/journals/sisc/22-3/35345.html
†Department of Mathematical and Computer Sciences, Colorado School of Mines, Golden, CO 80401-1887 (evanvlec@mines.edu). The work of this author was supported by NSF grants DMS-9505049 and DMS-9973393.

numerical shadowing methods presented here provide accurate results, due to the componentwise error bounds and the improved fixed point result, and are obtained with relatively small cost due to the use of efficient numerical linear algebra techniques.

Recent work on shadowing has involved extending shadowing to allow for rescaling of time, i.e., allowing for perturbations in the time step (see the work of Coomes [6], Coomes, Kocak, and Palmer [7], Palmer [22], Van Vleck [26], and Franke and Selgrade [12], [13]). Rescaling of time can be thought of as allowing for a special perturbation of the differential equation. Work on numerical shadowing includes the ground breaking work of Hammel, Yorke, and Grebogi [18], [19], the shadowing-type results of Beyn [2], the work of Chow, Lin, and Palmer (e.g., [4]), Chow and Palmer (e.g., [5]), the numerical work of Sauer and Yorke [24], and the initial work on the breakdown of shadowing of Dawson et al. [8]. We also mention here a related concept known as backward error analysis in which one considers a modified differential equation and provides error statements with respect to this modified equation. This has proved to be a very useful concept of error, especially in the case of Hamiltonian systems (see, e.g., [1], [15], [16], and [23]).

This paper is organized as follows. In section 2 we present the shadowing setup that we will employ throughout this paper, and in section 3 we present an improved fixed point result that has two fewer norm bounds than the previous result. In section 4 we consider a reformulation of the given differential equation and its linear variational equation by writing the dependent variable $x(t)$ as $x(t) = Q(t)y(t)$ and a fundamental matrix solution $X(t) = Q(t)R(t)$, respectively, where $Q(t)$ is a square orthogonal matrix and $R(t)$ is an upper triangular matrix, and by solving for y, Q, and R instead of x and X. Based upon this reformulation, we consider in section 5 the form of a right inverse, in our case the pseudoinverse, of a linear operator that approximates the Jacobian, and we provide bounds in section 6 on the elements of the vectors multiplying the right inverse. Section 7 contains algorithmic formulations and details, section 8 contains our numerical results, and section 9 contains our conclusions.

2. Shadowing setup. Consider the autonomous IVP where f is a C^2 function,

$$\left\{ \begin{array}{rl} \dot{x} & = f(x), \\ x(0) & = x_0, \end{array} \right. \tag{2.1}$$

and $x \equiv x(t) : \mathbb{R} \to \mathbb{R}^N$ for a positive integer N.

We assume that a numerical orbit $x \equiv \{x_n\}_0^M$ using time steps $h \equiv \{h_n\}_0^{M-1}$, for a positive integer M, has been produced, and let $t_{n+1} = t_n + h_n$ with $t_0 = 0$. Consider the nonlinear operator $F(x,h)$ with nth block component given by

$$(F(x,h))_n = x_{n+1} - \phi(x_n, h_n) =: \delta_{n+1}, \tag{2.2}$$

for $n = 0, \ldots, M-1$, where ϕ is the local solution operator and δ_{n+1} is the local error at the nth step. The linearization of $F(x,h)$ is given by $(D_{x_n}\phi(x_n,h_n) \equiv \frac{\partial \phi}{\partial x_n}(x_n,h_n)$ and $D_{h_n}\phi(x_n,h_n) \equiv \frac{\partial \phi}{\partial h_n}(x_n,h_n))$,

$$(DF(x,h)(\Delta x, \Delta h))_n = \Delta x_{n+1} - D_{x_n}\phi(x_n,h_n)\Delta x_n - \theta D_{h_n}\phi(x_n,h_n)\Delta h_n, \tag{2.3}$$

for $n = 0, \ldots, M-1$, where $\theta \geq 0$ is a scaling factor (see [26]). We assume that $DF(x,h)$ is approximated by a linear operator,

$$(L(\Delta x, \Delta h))_n = \Delta x_{n+1} - Y_n \Delta x_n - \theta f_{n+1} \Delta h_n, \tag{2.4}$$

for $n = 0, \ldots, M-1$. Throughout the paper we consider $z = x \in \mathbb{R}^{(M+1)N}$ for $\theta = 0$ and $z = (x, h) \in \mathbb{R}^{(M+1)N} \times \mathbb{R}^M$ for $\theta > 0$, with norms $||z|| = \sup_n ||x_n||$ for $\theta = 0$ and $||z|| = ||(x,h)|| = \max\{\sup_n ||x_n||, \sup_n |\theta^{-1} h_n|\}$ for $\theta > 0$. For $n = 0, \ldots, M-1$ the nth $N \times N$ block column of $DF(z)$ and of L is nonzero only in the nth block component. In addition, for $\theta > 0$ the remaining M columns of $DF(z) - L$ contain a nonzero block component of the form $\theta[f_{n+1} - D_{h_n}\phi(x_n, h_n)]$. The nth block component of $F(z)$ is given by δ_{n+1}.

3. Fixed point result. The following is an improved version of a fixed point result that has been used in many instances to prove the existence of a nearby solution in a shadowing context. The proof is basically the same, but it allows for the use of more precise componentwise information. The statement of the fixed point theorem is very similar to the statement of Theorem 1, p. 536, in Kantorovich and Akilov [20], and the proof is a minor modification of Proposition 4.1 in [4].

THEOREM 3.1. *Suppose that $\mathcal{X}, \mathcal{Y}$ are Banach spaces, $F : \mathcal{X} \to \mathcal{Y}$ is C^1, and that there exist positive constants ϵ_0, $\alpha < 1$, β, a point $z \in \mathcal{X}$, and a linear operator $L : \mathcal{X} \to \mathcal{Y}$ with right inverse $L^\dagger$ such that*

$$||L^\dagger(DF(w) - L)|| \le \alpha \quad \text{for} \quad ||w - z|| \le \epsilon_0. \tag{3.1}$$

If, for $0 < \epsilon \le \epsilon_0$,

$$||L^\dagger F(z)|| \le (1 - \alpha)\epsilon =: \beta, \tag{3.2}$$

then the equation $F(w) = 0$ has a solution with

$$||w - z|| \le \epsilon. \tag{3.3}$$

Our goal now is to find a bound $\alpha < 1$ in (3.1) and a bound on β in (3.2) so that we can set $\epsilon = \beta/(1 - \alpha)$ provided $\epsilon \le \epsilon_0$.

4. Reformulation of the differential equation. In order to decrease the shadowing distance when there are nonhyperbolic directions, for instance, in the neighborhood of a nontrivial attracting set or in the presence of first integrals, we rewrite the differential equation (2.1) to take advantage of componentwise local error tolerances. We introduce an auxiliary matrix differential equation that will also be useful in solving the linear variational equation. To make the idea clear, suppose for each time t we have an orthogonal matrix $Q(t) \in \mathbb{R}^{N \times N}$ whose columns correspond to directions of different rates of growth. Then we can write $x(t)$, the solution to (2.1), as a linear combination of the columns of $Q(t)$, i.e., $x(t) = Q(t)y(t)$. The components of $y(t)$ correspond to the growth rates in the directions given by the columns of $Q(t)$. If we write the solution to the linear variational equation, $X(t)$, as $X(t) = Q(t)R(t)$ (see [9]), where $Q(t)$ is orthogonal and $R(t)$ is upper triangular, then $Q(t)$ satisfies the following matrix differential equation, where $S(Q) \equiv S(Q, A(t))$ is a skew-symmetric matrix and $A(t) \equiv Df(x(t))$,

$$\dot{Q} = QS(Q) \quad \text{with} \quad S(Q)_{ij} = \begin{cases} (Q^T A Q)_{ij}, & i > j, \\ 0, & i = j, \\ -(Q^T A Q)_{ji}, & i < j. \end{cases} \tag{4.1}$$

Since $x(t) = Q(t)y(t)$, we have $\dot{x}(t) = \dot{Q}(t)y(t) + Q(t)\dot{y}(t)$, so $y(t)$ satisfies the differential equation

$$\dot{y} = Q^T f(Qy) - S(Q)y. \tag{4.2}$$

The equation for $R(t)$ is given by

$$\dot{R} = \tilde{A}R \quad \text{with} \quad \tilde{A} = Q^T AQ - S(Q); \tag{4.3}$$

the coefficient matrix $\tilde{A}(t)$ is thus upper triangular.

By solving numerically (4.1), (4.2), and (4.3), we obtain approximations to the original nonlinear problem and the linear variational equation. If we write $x_{n+1} = Q_{n+1}y_{n+1}$ and $\phi(x_n, h_n) = \tilde{Q}_{n+1}\psi(y_n, h_n)$, where ψ is the local solution operator for (4.2), then by (2.2), we have

$$Q_{n+1}^T\delta_{n+1} = Q_{n+1}^T[x_{n+1} - \phi(x_n, h_n)] = y_{n+1} - \psi(y_n, h_n) + (\tilde{Q}_{n+1}^T - Q_{n+1}^T)\phi(x_n, h_n). \tag{4.4}$$

We solve using (4.2) as opposed to (2.1) so that componentwise local error control may be obtained with respect to the moving coordinate system given by $Q(t)$. This could be accomplished by solving with (2.1) but would require that the error control for the x variable depend on our approximation to Q. Solving by using (4.1) and (4.3) is a numerically stable way of approximating the linear variational equation.

5. Form of the pseudoinverse. We choose for the right inverse, $L^\dagger$, in Theorem 3.1, the pseudoinverse. This allows us to find $L^\dagger$ explicitly, which is useful when determining the bounds for α and β in (3.1) and (3.2). The choice of the pseudoinverse also has the advantage that it is independent of the dynamics of the problem, i.e., does not depend on the number of stable, unstable, or neutral modes, and it is the optimal choice with respect to the two norm.

We solve (4.1) and (4.2) numerically to produce sequences $\{Q_n\}_0^M$ and $\{y_n\}_0^M$. Simultaneously, for $n = 0, \dots, M-1$, we solve

$$\begin{cases} \dot{\tilde{R}}_n(t) = & \tilde{A}(t + t_n)\tilde{R}_n(t), \\ \tilde{R}_n(0) = & I \end{cases} \tag{5.1}$$

for $0 \le t \le h_n$. If we let R_n denote the approximation to $\tilde{R}(h_n)$, then in (2.4) we have $Y_n = Q_{n+1}R_nQ_n^T$. Notice that L in (2.4) can be written as $L = \overline{Q}U\hat{Q}^T$, where $\overline{Q}, \hat{Q}$ are block diagonal matrices given by $\overline{Q} = \text{diag}(Q_1, \dots, Q_M)$, $\hat{Q} = \text{diag}(Q_0, \dots, Q_M, I_M)$, where I_M is the $M \times M$ identity matrix, and in block indices,

$$(U(z, \tau))_n = z_{n+1} - R_nz_n - \theta g_{n+1}\tau_n, \tag{5.2}$$

where $f_{n+1} = Q_{n+1}g_{n+1}$ and $f_{n+1} = f(x_{n+1})$. Thus, the pseudoinverse of L, $L^\dagger$, can be written as

$$L^\dagger = \hat{Q}U^\dagger\overline{Q}^T, \tag{5.3}$$

where $U^\dagger$ is the pseudoinverse of U. We write $U^\dagger = U^T(UU^T)^{-1}$ and note that UU^T is symmetric and block tridiagonal with entries $(UU^T)_{n,n} = I + R_nR_n^T + \theta^2 g_{n+1}g_{n+1}^T$ for $n = 0, \dots, M-1$ and $(UU^T)_{n-1,n} = -R_n^T$ for $n = 1, \dots, M-1$.

We decompose $L^\dagger$ as in (5.3) and determine componentwise bounds on $\overline{Q}^T[DF(w) - L]$ and $\overline{Q}^TF(z)$. Once componentwise bounds are obtained we find the bounds α and β in (3.1) and (3.2), respectively, using the following simple idea. Let u denote

an arbitrary column of $\overline{Q}^T[DF(w) - L]$ or let u denote $\overline{Q}^T F(z)$. Assume that the ith component of u, u_i, satisfies $|u_i| \leq \gamma_i$ for some $\gamma_i > 0$. If $y = \hat{Q}U^\dagger u$, then we have

$$|y_i| \leq \sum_j |(\hat{Q}U^\dagger)_{ij}| \cdot \gamma_j. \tag{5.4}$$

Using the results in [21] and the references therein we have that the (i, j) block of $(UU^T)^{-1}$ has the form

$$\Gamma_{ij} \equiv (UU^T)^{-1}_{ij} = \begin{cases} \Psi_i \Omega_j^T & \text{for} \quad i \leq j, \\ \Omega_i \Psi_j^T & \text{for} \quad i > j, \end{cases} \tag{5.5}$$

where Ψ_i and Ω_j are $N \times N$ matrices. In addition, the block vectors $\Psi = (\Psi_0^T, \ldots, \Psi_{M-1}^T)^T$ and $\Omega = (\Omega_0^T, \ldots, \Omega_{M-1}^T)^T$ can be determined by solving

$$(UU^T)\Omega = \Xi_0 \quad \text{and} \quad (UU^T)\Psi = \Xi_{M-1}\Omega_{M-1}^{-T}, \tag{5.6}$$

where Ξ_0 denotes the block vector with the identity matrix in the 0th block and Ξ_{M-1} denotes the block vector with the identity in the $(M-1)$th block (with all other blocks zero). In general, (5.6) is not a numerically stable way to form $(UU^T)^{-1}$.

In the next section we will obtain componentwise bounds on $\overline{Q}^T F(z)$ and $\overline{Q}^T[DF(w) - L]$ to form the vectors that will multiply the matrix $\hat{Q}U^\dagger$ which has, using the form of $(UU^T)^{-1}$, the block entries,

$$(\hat{Q}U^\dagger)_{ij} = \begin{cases} -Q_0 R_0^T \Gamma_{0,j}, & i = 0, \\ Q_i[\Gamma_{i-1,j} - R_i^T \Gamma_{i,j}], & i = 1, \ldots, M-1, \\ Q_M \Gamma_{M-1,j}, & i = M \end{cases} \tag{5.7}$$

for $j = 0, \ldots, M-1$, and for $\theta > 0$, the remaining M rows of $\hat{Q}U^T(UU^T)^{-1}$ have the form

$$-\theta(g_i^T \Gamma_{i-1,0}, \ldots, g_i^T \Gamma_{i-1,M-1}) \quad \text{for} \quad i = 1, \ldots, M. \tag{5.8}$$

Note that, by using the form of L, we have (for $\theta = 0$) that (2.2) is of the form

$$Q_{n+1}^T \Delta x_{n+1} - R_n Q_n^T \Delta x_n = Q_{n+1}^T \delta_{n+1}, \tag{5.9}$$

which after a change of variables ($z_n = Q_n^T \Delta x_n$ and $\rho_n = Q_n^T \delta_n$) has the form

$$z_{n+1} - R_n z_n = \rho_{n+1}. \tag{5.10}$$

Consider the following simplified example of (5.10) with

$$R_n = \begin{pmatrix} R_{11} & R_{12} & R_{13} \\ 0 & R_{22} & R_{23} \\ 0 & 0 & R_{33} \end{pmatrix}, \tag{5.11}$$

where $R_{11} > 1$, $0 < R_{33} < 1$, $R_{22} = 1$, and $\rho_n \equiv \rho$ for all n. If we solve (5.10) forward in time for the second and third components of z with initial conditions of 0 and backward in the first component of z with terminal condition 0, then we obtain linear growth in the solution z unless we set the second and third components of ρ to $O(M^{-1})$ times the first component of ρ. This suggests having, after considering

(4.4), a relatively smaller local error tolerance for the components of y and columns of Q that correspond to nonhyperbolic and contractive directions (assuming that the directions are ordered as in our simple example). Similarly, we will use a relatively smaller local error tolerance for the rows of R corresponding to nonhyperbolic and contractive directions. It is important to point out that, although our plan is to require smaller error tolerances when computing the coupled set of equations for y, Q, and R numerically, this should really only restrict our time step in the integration of Q since the time step for y and R should be based on the unstable directions.

6. Bounds on $\overline{Q}^T F(z)$ and $\overline{Q}^T[DF(w) - L]$. In what follows, let $\mathtt{U}$ denote the set of indices corresponding to unstable directions and let $\mathtt{CS}$ denote the set indices corresponding to nonhyperbolic and stable directions based upon some criteria, for instance, the average magnitude of the diagonal elements of the upper triangular coefficient matrix $\tilde{A}(t)$ from (4.3).

We first consider the case in which $\theta = 0$ and wish to determine componentwise bounds on $Q_{n+1}^T \delta_{n+1}$ and on $Q_{n+1}^T[D_{w_n}\phi(w_n, h_n) - Y_n]$, which we rewrite as

$$\begin{aligned} (6.1)\quad & Q_{n+1}^T (DF(w) - L)_{n,n} \\ & := Q_{n+1}^T [D_{w_n}\phi(w_n, h_n) - D_{x_n}\phi(x_n, h_n)] + Q_{n+1}^T [D_{x_n}\phi(x_n, h_n) - Y_n]. \end{aligned}$$

For $0 \le t \le h_n$ define $y_n(t)$ and $Y_n(t)$ with $y_n(0) = x_n$ and $Y_n(0) = I$ as the numerical solution to the original nonlinear problem and the linear variational equation, respectively, using dense or continuous output (see [17, p. 176]). Let $E_n(t) := D_{x_n}\phi(x_n, t) - Y_n(t)$ for $0 \le t \le h_n$, where $E_n(0) = 0$. In the following lemma we consider componentwise local errors in $Q_{n+1}^T \delta_{n+1}$ and in $Q_{n+1}^T E_n(h_n)$ when componentwise local errors are satisfied in y, Q, and R.

LEMMA 6.1. *Let $\eta_{\mathtt{U}}$ and $\eta_{\mathtt{CS}}$ denote the local errors in the components of y, the columns of Q, and the rows of R_n corresponding to the indices* $\mathtt{U}$ *and* $\mathtt{CS}$*, respectively, with $\eta_{\mathtt{U}} \ge \eta_{\mathtt{CS}}$. Then we have*

$$(6.2)\qquad |(Q_{n+1}^T \delta_{n+1})_i| \le (1 + ||x_{n+1}||_1)\eta_\iota + O(\eta_{\mathtt{U}}^2) =: \mu_n \eta_\iota + O(\eta_{\mathtt{U}}^2),$$

(6.3)

$$|(Q_{n+1}^T E_n(h_n))_{ij}| \le \left(\chi_{\{i\le j\}} + \sum_{k=1}^{j} |(R_n)_{kj}|\right)\eta_\iota + O(\eta_{\mathtt{U}}^2) =: (\chi_{\{i\le j\}} + \sigma_{nj})\eta_\iota + O(\eta_{\mathtt{U}}^2),$$

where $\iota := \mathtt{U}$ for $i \in \mathtt{U}$, $\iota := \mathtt{CS}$ for $i \in \mathtt{CS}$, and $\chi_{\{i\le j\}}$ is 1 for $i \le j$ and 0 otherwise.

Proof. The inequality (6.2) follows using (4.4) and the fact that we have $x_{n+1} - \phi(x_n, h_n) = \delta_{n+1}$. The inequality (6.3) follows similarly. ∎

Remark 6.1. Note that if we bound the local error for the ith column of Q by $\eta_\iota/(1+\max\{||R_n||_1, ||x_{n+1}||_1\})$, then the bounds (6.2) and (6.3) in Lemma 6.1 become $2\eta_\iota + O(\eta_{\mathtt{U}}^2)$ and $(\chi_{\{i\le j\}}+1)\eta_\iota + O(\eta_{\mathtt{U}}^2)$, respectively, where $\iota := \mathtt{U}$ for $i \in \mathtt{U}$, and $\iota := \mathtt{CS}$ for $i \in \mathtt{CS}$.

Given the bound (6.3) for $Q_{n+1}^T E_n(t)$ we focus on determining componentwise bounds on $Q_{n+1}^T[D_{w_n}\phi(w_n, h_n) - D_{x_n}\phi(x_n, h_n)]$. Consider the linear equations

$$(6.4)\qquad Q_{n+1}^T \dot{W} = [Q_{n+1}^T Df(\phi(w_n, t))Q_{n+1}]Q_{n+1}^T W \equiv B(t)Q_{n+1}^T W,\ W(0) = I,$$

and

$$(6.5)\qquad Q_{n+1}^T \dot{X} = [Q_{n+1}^T Df(\phi(x_n, t))Q_{n+1}]Q_{n+1}^T X \equiv C(t)Q_{n+1}^T X,\ X(0) = I.$$

Let $V(t) := Q_{n+1}^T[W(t) - X(t)]$ and observe that $V(0) = 0$. Then we have

$$\dot{V} = B(t)V + [B(t) - C(t)](Q_{n+1}^T[Y_n(t) + E_n(t)]). \tag{6.6}$$

DEFINITION 6.1. *Define a and b_j as constants that satisfy $a \geq \sup_{0\leq t\leq h_n} ||B(t)||_\infty$ and $b_j \geq \sup_{0\leq t\leq h_n} ||\Pi_j||_\infty$, where Π_j denotes the jth column of*

$$[B(t) - C(t)](Q_{n+1}^T[Y_n(t) + E_n(t)]). \tag{6.7}$$

LEMMA 6.2. *With the assumptions of Lemma* 6.1 *we have*

$$|(Q_{n+1}^T[D_{w_n}\phi(w_n, h_n) - D_{x_n}\phi(x_n, h_n)])_{ij}| \leq \frac{b_j}{a}(\exp(ah_n) - 1). \tag{6.8}$$

Proof. Let $z_j(t)$ denote the supremum norm of the jth column of $V(t)$. Then we have

$$\frac{d}{dt}z_j(t) \leq az_j(t) + b_j, \tag{6.9}$$

so by Gronwall's inequality (see [25, p. 108]) and using the fact that $z_j(0) = 0$, we have

$$z_j(h_n) \leq \frac{b_j}{a}(\exp(ah_n) - 1). \qquad \square \tag{6.10}$$

Using (6.3) and (6.8), we obtain the following bounds on the magnitude of the components of $Q_{n+1}^T[D_{w_n}\phi(w_n, h_n) - Y_n]$.

COROLLARY 6.1. *With the assumptions of Lemma* 6.1 *we have*

$$\begin{aligned} |(Q_{n+1}^T[D_{w_n}\ \phi(w_n, h_n) - Y_n])_{ij}| &\leq \left(\chi_{\{i\leq j\}} + \sum_{k=1}^{j} |(R_n)_{kj}|\right)\eta_\iota + O(\eta_{\mathtt{U}}^2) \\ &+ \frac{b_j}{a}(\exp(ah_n) - 1) =: (\chi_{\{i\leq j\}} + \sigma_{nj})\eta_\iota + \rho_{nj} + O(\eta_{\mathtt{U}}^2), \end{aligned} \tag{6.11}$$

where $\iota := \mathtt{U}$ for $i \in \mathtt{U}$, and $\iota := \mathtt{CS}$ for $i \in \mathtt{CS}$.

Our task now is to determine computable bounds on a and b_j. The difficulty occurs because the matrix functions $B(t)$ and $C(t)$ are not explicitly known.

We assume that either f satisfies a one-sided Lipschitz condition (see [25, p. 173]) so there exists a constant L_{os} such that

$$\langle f(w_n(t)) - f(x_n(t)), w_n(t) - x_n(t)\rangle \leq L_{\text{os}}||w_n(t) - x_n(t)||_2^2, \tag{6.12}$$

or f is locally Lipschitz so that there exists a nonnegative constant L_{loc} such that

$$||f(w_n(t)) - f(x_n(t))||_\infty \leq L_{\text{loc}}||w_n(t) - x_n(t)||_\infty. \tag{6.13}$$

Using Gronwall's inequality for $\Lambda = L_{\text{os}}$ (and $\varepsilon_0 = \sqrt{N}\epsilon_0$) or $\Lambda = L_{\text{loc}}$ (and $\varepsilon_0 = \epsilon_0$), respectively, we have

$$||w_n(t) - x_n(t)||_\infty \leq \exp(\Lambda t) \cdot \varepsilon_0. \tag{6.14}$$

Let L_{Df} denote the Lipschitz constant for Df so that

$$\begin{aligned} ||Df(w_n(t)) - Df(x_n(t))||_\infty &\leq L_{Df}||w_n(t) - x_n(t)||_\infty, \\ ||Df(x_n(t)) - Df(y_n(t))||_\infty &\leq L_{Df}||x_n(t) - y_n(t)||_\infty. \end{aligned} \tag{6.15}$$

Thus,

$$\begin{aligned} |Df(w_n(t))_{ij} - Df(x_n(t))_{ij}| &\le \varepsilon_0 \exp(\Lambda t) L_{Df} =: G_1(t), \\ |Df(x_n(t))_{ij} - Df(y_n(t))_{ij}| &\le L_{Df} ||\delta_{n+1}(t)||_\infty =: G_2(t), \end{aligned} \tag{6.16}$$

where $\delta_{n+1}(t)$ is the local error at the nth step at time t for $0 \le t \le h_n$. Then we have

$$\begin{aligned} |Df(w_n(t))_{ij}| &\le |Df(w_n(t))_{ij} - Df(x_n(t))_{ij}| \\ &\quad + |Df(x_n(t))_{ij} - Df(y_n(t))_{ij}| + |Df(y_n(t))_{ij}| \\ &\le G_1(t) + G_2(t) + |Df(y_n(t))_{ij}| =: H_{ij}(t). \end{aligned} \tag{6.17}$$

We have the following for a:

$$a := \sup_{0\le t\le h_n} \max_{1\le i\le N} \sum_{j=1}^{N}\sum_{k=1}^{N}\sum_{l=1}^{N} |(Q_{n+1})_{ki}| \cdot H_{kl}(t) \cdot |(Q_{n+1})_{lj}|. \tag{6.18}$$

For b_j we obtain

$$\begin{aligned} b_j := \sup_{0\le t\le h_n} \sum_{i=1}^{N}\sum_{k=1}^{N}\sum_{l=1}^{N} |(Q_{n+1})_{ki}| \cdot |G_1(t)(Q_{n+1})_{kl}| \cdot (|(Q_{n+1}^T Y_n(t))_{lj}| \\ +|(Q_{n+1}^T E_n(t))_{lj}|). \end{aligned} \tag{6.19}$$

The bounds for a and b_j given in (6.18) and (6.19), respectively, combined with Lemma 6.1 and Corollary 6.1 provide bounds on the components of $\overline{Q}^T F(z)$ and $\overline{Q}^T(DF(w) - L)$ for $||w - z|| \le \epsilon_0$ when $\theta = 0$.

When $\theta > 0$, we need to determine bounds on

$$Q_{n+1}^T[D_{w_n}\phi(w_n, t) - D_{w_n}\phi(w_n, h_n)] \quad \text{and} \quad Q_{n+1}^T[f(\phi(w_n, t)) - f_{n+1}] \tag{6.20}$$

for $|t - h_n| \le \theta\epsilon$ and $||w_n - x_n||_\infty \le \epsilon_0$.

DEFINITION 6.2. *Define constants such that a_θ satisfies $a_\theta \ge \sup_{|t-h_n|\le\theta\epsilon} ||B(t)||_\infty$, $b_{\theta,j}$ satisfies $b_{\theta,j} \ge \sup_{|t-h_n|\le\theta\epsilon} ||\Lambda_j||_\infty$, where Λ_j denotes the jth column of*

$$B(t)Q_{n+1}^T D_{w_n}\phi(w_n, h_n), \tag{6.21}$$

and c_θ and d satisfy

$$c_\theta \ge \sup_{\{t:0\le t\le s, |s-h_n|\le\theta\epsilon\}} ||B(t)Q_{n+1}^T f(\phi(w_n, h_n))||_\infty$$

and

$$d \ge \sup_{0\le t\le h_n} ||[B(t) - C(t)]Q_{n+1}^T f(\phi(x_n, t))||_\infty.$$

LEMMA 6.3. *With the assumptions of Lemma 6.1 we have for $|t - h_n| \le \theta\epsilon$,*

$$|(Q_{n+1}^T[D_{w_n}\phi(w_n, t) - D_{w_n}\phi(w_n, h_n)])_{ij}| \le \frac{b_{\theta,j}}{a_\theta}(\exp(a_\theta\theta\epsilon) - 1). =: \gamma_{j,n}. \tag{6.22}$$

Proof. First derive an expression similar to (6.6), and then the proof follows using the same technique as the proof of Lemma 6.2. ∎

We have

$$
\begin{aligned}
Q_{n+1}^T[f(\phi(w_n,t)) - f_{n+1}] &= Q_{n+1}^T[\{f(\phi(w_n,t)) - f(\phi(w_n,h_n))\} \\
&\quad +\{f(\phi(w_n,h_n)) - f(\phi(x_n,h_n))\} + \{f(\phi(x_n,h_n)) - f_{n+1}\}]
\end{aligned} \tag{6.23}
$$

for $|t-h_n| \le \theta\epsilon$. Using the fact that $Q_{n+1}^T f(\phi(w_n,t))$ satisfies (6.4) and $Q_{n+1}^T f(\phi(x_n,t))$ satisfies (6.5), we may obtain bounds on the left-hand side of (6.23) in a similar fashion to those previously obtained.

LEMMA 6.4. *With the assumptions of Lemma* 6.1 *we have, for* $|t-h_n| \le \theta\epsilon$,

$$
\begin{aligned}
||Q_{n+1}^T[f(\phi(w_n,t)) - f_{n+1}]||_\infty &\le \tfrac{c_\theta}{a_\theta}(\exp(a_\theta\theta\epsilon) - 1) + \tfrac{d}{a}(\exp(ah_n) - 1) \\
&\quad +||Q_{n+1}^T[f(\phi(x_n,h_n)) - f_{n+1}]||_\infty := \zeta_n.
\end{aligned} \tag{6.24}
$$

We now determine computable bounds on a_θ, $b_{\theta,j}$, c_θ, and d_θ in a similar spirit to those for a and b_j. For a_θ we use (6.17) with the limits of the supremum changed to $|t-h_n| \le \theta\epsilon$ in (6.18). To find the bound for $b_{\theta,j}$, we bound $B(t)$ as for a_θ and then bound $Q_{n+1}^T D_{w_n}\phi(w_n,h_n)$ using (6.3) and (6.8). We bound c_θ and d in a fashion similar to that of $b_{\theta,j}$ and b_j, respectively, and employ the bound

$$
||Q_{n+1}^T[f(\phi(x_n,h_n)) - f_{n+1}]||_\infty \le ||Q_{n+1}^T||_\infty L_f||\delta_{n+1}||_\infty. \tag{6.25}
$$

Putting all of these bounds together we have the following theorem that summarizes the bounds for $\theta = 0$ and $\theta > 0$.

THEOREM 6.5. *With the assumption of Lemma* 6.1, *for* $\theta = 0$ *the bounds on the nonzero components of* $\overline{Q}^T F(z)$ *are given by inequality* (6.2) *in Lemma* 6.1, *while the bounds on the nonzero components of* $\overline{Q}^T(DF(w) - L)$ *are given by inequality* (6.11) *in Corollary* 6.1. *Bounds on the quantities* a *and* b_j *in Corollary* 6.1 *may be obtained using inequalities* (6.12)–(6.19).

For $\theta > 0$ *the bounds for the nonzero components of* $\overline{Q}^T(DF(w) - L)$ *are given by combining the bound* (6.11) *in Corollary* 6.1 *with the bound* (6.22) *in Lemma* 6.3 *and the bound* (6.24) *in Lemma* 6.4. *In addition, bounds on* a_θ, $b_{\theta,j}$, c_θ, *and* d *may be obtained using the inequalities* (6.12)–(6.25).

7. Implementation. The basic outline of our algorithm for solving and producing a shadowing distance estimate for an approximate solution of the IVP (2.1) is as follows.

ALGORITHM. Given constants $\epsilon_0 > 0$ and $\theta \ge 0$ and componentwise tolerances, $\eta_{\mathtt{U}} \ge \eta_{\mathtt{CS}} > 0$ with respect to the indices `U` and `CS`.

1. Simultaneously approximate solutions to (4.1), (4.2), and (4.3).
2. Update the elements of the matrix UU^T, and store y_n, Q_n, R_n, and f_{n+1}.
3. Bound componentwise the quantities $\overline{Q}^T F(z)$ and $\overline{Q}^T(DF(w) - L)$ using Theorem 6.5.
4. Decompose the matrix UU^T and form $\hat{Q}U^\dagger$ (see (5.3)).
5. Using the results from steps 3 and 4, estimate α in (3.1) and β in (3.2) using (5.4).
6. If $\alpha < 1$ and $\beta/(1-\alpha) \le \epsilon_0$, then set the shadowing distance $\epsilon := \beta/(1-\alpha)$.

Implementation Details.

1. We employ ODEX [17] or RKSUITE [3] to approximate the differential equations and use a projected integration scheme to maintain orthogonality of Q (see [9]) using a modified Gram–Schmidt procedure [14]. We modified the error control of ODEX to enforce componentwise error tolerances. The use of componentwise tolerances is a standard feature in RKSUITE. We bound the local error for the columns of Q as described in Remark 6.1 and set $\mu_n = 2$ and $\sigma_{nj} = 1$ for all n and j. For RKSUITE we employ the $(7, 8)$ Runge–Kutta pair.

2. We use symmetric block tridiagonal storage when forming UU^T.

3. We estimate $\overline{Q}^T F(z)$ and $\overline{Q}^T(DF(w) - L)$ by sampling at the mesh points $\{t_n\}_0^M$ so that $\sup_{0\le t\le h_n}$ is replaced by $\max_{t\in\{0,h_n\}}$ and we neglect the $O(\eta_{\mathtt{U}}^2)$ terms in (6.2) and (6.3).

4. We solve for some of the rows of $\hat{Q}U^\dagger$ and then use (5.4) to estimate α and β. This provides lower bounds on α and β that will be sharp if the "correct" rows are used. We use the first and last block rows of $\hat{Q}U^\dagger$ and several picked at random to obtain an estimate on α without the full cost of solving for all of $U^\dagger$. To determine block rows of $U^\dagger$ we use the block tridiagonal solver TRDBLK [11].

8. Numerical results. In this section we present numerical results of our shadowing algorithm applied to several nonlinear IVPs. We obtain approximate solutions and local error estimates from the IVP solvers ODEX and RKSUITE. In the description of our numerical results we denote by "M" the number of time steps taken; "U" denotes the set of indices for which the local error tolerance "$\eta_{\mathtt{U}}$" is employed, while the local error tolerance "$\eta_{\mathtt{CS}}$" is used for the remaining indices. The value "T" is the final time reached during the numerical computation; "QhUdnorm" is the maximum row sum of the rows of $\hat{Q}U^\dagger$ that were computed. For the numerical experiments recorded here we computed six block rows: the first two, last two, and middle two block rows. When $\theta > 0$ we compute the corresponding additional rows of $\hat{Q}U^\dagger$. We found some variation in QhUdnorm when using other rows of $\hat{Q}U^\dagger$, but always within the same order of magnitude and, in fact, never greater than twice the value of QhUdnorm reported here. We recommend randomly choosing the rows of $\hat{Q}U^\dagger$ to be computed, but have chosen to compute these particular six block rows for definiteness. We also report on the value "α" in (3.1) and the value of the shadowing distance "ϵ" that was obtained. We will indicate with "–" if there was no value of ϵ_0 for which we were able to satisfy $\alpha < 1$ and $\epsilon \le \epsilon_0$. All computations where performed on a Linux workstation with 512MB memory using a 400MHz processor. We found that the computation of y, Q, and R comprised approximately 85 percent of the computation time when using ODEX and approximately 70 percent of the computation time when using RKSUITE.

To compare the numerical shadowing technique presented here with other shadowing techniques we consider two categories of techniques: (i) "BVP techniques"—these are based upon forming a right inverse by specifying boundary conditions using the stability properties of the system and then bounding the norm of a solution to this BVP; (ii) "pseudoinverse techniques"—this is the approach taken here and involves finding and bounding the norm of a particular right inverse, the pseudoinverse. In general, the BVP technique (see, e.g., [5], [7]) provide somewhat conservative values for the shadowing distance, but they are obtained rather inexpensively. The pseudoinverse techniques (see, e.g., [26] and this paper) are expensive, but sharper. The expense is due to the requirement that we find $L^\dagger$ explicitly so the norm of the pseudoinverse is obtained essentially exactly. By obtaining only selected rows of $L^\dagger$

TABLE 8.1
Forced vdP ($M = 1E+5$, $\mathtt{U} = \{1\}$, $\eta_{\mathtt{U}} = 1.E-6$).

Meth	$\eta_{\mathtt{CS}}/\eta_{\mathtt{U}}$	θ	T	QhUdnorm	α	ϵ
RKSUITE	$1.E0$	$0.E0$	$1.56E+4$	$2.1E+2$	$1.3E-1$	$4.9E-4$
RKSUITE	$1.E0$	$1.E-1$	$1.56E+4$	$1.9E+1$	$1.9E-2$	$4.0E-5$
RKSUITE	$1.E-1$	$1.E-1$	$1.23E+4$	$2.4E+1$	$1.9E-2$	$2.8E-5$
RKSUITE	$1.E-2$	$1.E-1$	$9.80E+3$	$2.8E+1$	$1.9E-2$	$3.4E-5$
ODEX	$1.E0$	$0.E0$	$2.58E+4$	$1.9E+2$	–	–
ODEX	$1.E-1$	$0.E0$	$2.27E+4$	$1.8E+2$	$6.0E-1$	$9.0E-4$
ODEX	$1.E-2$	$0.E0$	$1.50E+4$	$2.0E+2$	$3.4E-1$	$5.9E-4$
ODEX	$1.E0$	$1.E-1$	$2.58E+4$	$1.5E+1$	$1.0E-1$	$3.3E-5$
ODEX	$1.E-1$	$1.E-1$	$2.27E+4$	$1.3E+1$	$8.3E-2$	$1.4E-5$
ODEX	$1.E-2$	$1.E-1$	$1.50E+4$	$3.6E+1$	$6.3E-2$	$2.2E-5$

we have decreased the expense but at the cost of perhaps not having a bound on the norm of the entire pseudoinverse. Regardless of how the right inverse has been chosen and bounded, a result similar to Proposition 4.1 in [4] has been used (see [5], [7], and [26]) to determine the shadowing distance ϵ. With the standard fixed point result, which takes $\alpha = 1/2$ and has two additional norm bounds, the shadowing distance is approximately $2||L^{\dagger}||\delta$ where δ is a uniform bound on the norm of the local error. With the formulation and fixed point result presented here, for $\eta_{\mathtt{CS}}/\eta_{\mathtt{U}} = 1$ the best possible shadowing distance is $\epsilon = \mathtt{QhUdnorm} \cdot 2\eta_{\mathtt{U}}$, where $2\eta_{\mathtt{U}}$ is our bound on the supremum norm of the local error tolerance (see Remark 6.1), and the results can only improve for $\eta_{\mathtt{CS}}/\eta_{\mathtt{U}} < 1$.

Example 8.1. As our first example we consider a forced van der Pol oscillator

$$\ddot{y} + \alpha(y^2 - 1)\dot{y} + y = \beta\cos(\omega t) \tag{8.1}$$

with $\alpha = \frac{2}{5}$, $\beta = 2\alpha^2$, and $\omega = \sqrt{1-\alpha^2}$ and the initial conditions $(0,0)$. In our numerical experiments, we set $\epsilon_0 = 5.E-4$ when using RKSUITE and set $\epsilon_0 = 1.E-3$ when using ODEX.

In Table 8.1, we see a decrease by an order of magnitude in QhUdnorm for $\theta > 0$ since one of the Lyapunov exponents is close to zero. In addition, there is some improvement in the shadowing distance ϵ when $\eta_{\mathtt{CS}}/\eta_{\mathtt{U}} < 1$. There is a trade-off here in that there is some decrease in the average stepsize when $\eta_{\mathtt{CS}}/\eta_{\mathtt{U}} < 1$, which tends to increase QhUdnorm, but in spite of this there is a decrease in ϵ especially for $\eta_{\mathtt{CS}}/\eta_{\mathtt{U}} = 1.E-1$. This is not a particularly good example for our method since both of the Lyapunov exponents are negative (we are using $\mathtt{U} = \{1\}$); nonetheless we do obtain some improvement with $\eta_{\mathtt{CS}}/\eta_{\mathtt{U}} < 1$.

Example 8.2. The Lorenz equation is given by

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{pmatrix} = \begin{pmatrix} \sigma(y-x) \\ \rho x - xz - y \\ xy - \beta z \end{pmatrix}. \tag{8.2}$$

We consider the parameter values $\sigma = 10$, $\rho = 28$, and $\beta = 8/3$, and the initial condition is chosen to be $(x_0, y_0, z_0) = (0, 1, 0)$. In our numerical experiments, we set $\epsilon_0 = 5.E-5$ when using RKSUITE and set $\epsilon_0 = 2.E-6$ when using ODEX. In Table 8.2, observe the decrease by an order of magnitude in the shadowing distance approximation ϵ with RKSUITE when changing $\eta_{\mathtt{CS}}/\eta_{\mathtt{U}}$ from $1.E0$ to $1.E-1$. In general, we tend to observe smaller shadowing distances for ODEX than for RKSUITE, ostensibly

TABLE 8.2
Lorenz equation ($M = 1E+5$, $\mathtt{U} = \{1\}$, $\eta_{\mathtt{U}} = 1.E-8$, $\theta = 1.E-2$).

Meth	$\eta_{\mathtt{CS}}/\eta_{\mathtt{U}}$	T	QhUdnorm	α	ϵ
RKSUITE	$1.E0$	$1.31E+3$	$1.4E+3$	$1.3E-1$	$3.3E-5$
RKSUITE	$1.E-1$	$9.90E+2$	$1.9E+3$	$1.6E-1$	$4.9E-6$
RKSUITE	$1.E-2$	$7.42E+2$	$2.5E+3$	$2.1E-1$	$7.3E-6$
ODEX	$1.E0$	$6.10E+3$	$2.4E+2$	$1.2E-1$	$1.9E-6$
ODEX	$1.E-1$	$6.29E+3$	$2.6E+2$	$1.2E-1$	$6.7E-7$
ODEX	$1.E-2$	$5.79E+3$	$2.8E+2$	$2.6E-2$	$4.3E-7$
ODEX	$1.E-3$	$5.54E+3$	$3.1E+2$	$1.3E-2$	$3.3E-7$

TABLE 8.3
Lorenz equation ($M = 4E+5$, $\mathtt{U} = \{1\}$, $\eta_{\mathtt{U}} = 1.E-8$, $\eta_{\mathtt{CS}} = 1.E-9$, $\theta = 1.E-2$).

Meth	ϵ_0	T	QhUdnorm	α	ϵ
RKSUITE	$5.E-6$	$3.97E+3$	$1.4E+3$	$1.2E-2$	$3.0E-6$
ODEX	$1.E-6$	$2.52E+4$	$1.0E+2$	$1.2E-1$	$6.7E-7$

TABLE 8.4
Coupled oscillators ($\mathtt{U} = \{1,2\}$, $\eta_{\mathtt{U}} = 1.E-10$, $\epsilon_0 = 2.E-7$).

Meth	$\eta_{\mathtt{CS}}/\eta_{\mathtt{U}}$	θ	T	QhUdnorm	M	ϵ
RKSUITE	$1.E0$	$0.E0$	$1.07E+2$	$2.9E+5$	$1E+4$	–
RKSUITE	$1.E0$	$1.E-2$	$1.07E+2$	$9.1E+2$	$1E+4$	$1.8E-7$
RKSUITE	$1.E-1$	$1.E-2$	$8.41E+1$	$1.2E+3$	$1E+4$	$1.1E-7$
RKSUITE	$1.E-2$	$1.E-2$	$6.40E+1$	$1.7E+3$	$1E+4$	$1.3E-7$
RKSUITE	$1.E-1$	$1.E-2$	$8.49E+2$	$1.2E+3$	$1E+5$	$1.1E-7$
ODEX	$1.E0$	$0.E0$	$4.50E+2$	$4.1E+5$	$1E+4$	–
ODEX	$1.E0$	$1.E-2$	$4.50E+2$	$3.9E+2$	$1E+4$	$7.9E-8$
ODEX	$1.E-1$	$1.E-2$	$4.29E+2$	$2.7E+2$	$1E+4$	$2.4E-8$
ODEX	$1.E-2$	$1.E-2$	$4.05E+2$	$2.9E+2$	$1E+4$	$3.9E-8$
ODEX	$1.E-1$	$1.E-2$	$4.25E+3$	$2.7E+2$	$1E+5$	$2.4E-8$

due to the larger average stepsize achieved by the higher order of the two methods, ODEX. In Table 8.3, we report on longer time simulations performed with $M = 4E+5$ time steps.

Example 8.3. The next example we consider is a ring of oscillators with an external force proportional to the position component of the limit cycle of a van der Pol oscillator. In particular, we consider the following system, similar to that considered in [10],

$$\begin{aligned} &\ddot{y} + \alpha(y^2-1)\dot{y} + \omega^2 y = 0, \\ &\ddot{x}_i + d_i\dot{x}_i + \gamma[\Phi'(x_i - x_{i-1}) - \Phi'(x_{i+1} - x_i)] = \sigma y \delta_{i1}\,, \quad i = 1,\ldots,\hat{N}\,. \end{aligned} \tag{8.3}$$

The function $\Phi(x) = (x^2/2) + (x^4/4)$ is the single well Duffing potential, $\alpha, \omega, \gamma, \sigma$ are scalar parameters, x_i is the displacement of the ith particle, d_i is the damping coefficient, and we have periodic boundary conditions ($x_0 = x_{\hat{N}}$ and $x_{\hat{N}+1} = x_1$). For the experiments we set $\hat{N} = 2$, $\alpha = 1$, $\omega = 1.82$, $\gamma = 1$, $\sigma = 4$, and $d_1 = d_2 = 3.E-2$ and employ the initial conditions $(y, \dot{y}, x_1, \dot{x}_1, x_2, \dot{x}_2) = (1,0,0,0,0,0)$. In Table 8.4 we report on simulations for the forced coupled oscillators. The importance of rescaling of time is seen in the decrease in QhUdnorm as we increase θ from $0.E0$ to $1.E-2$. In addition, we see the benefits of $\eta_{\mathtt{CS}}/\eta_{\mathtt{U}} < 1$ especially when using ODEX, in particular for $\theta = 1.E-2$ we see a nontrivial decrease in the shadowing distance when $\eta_{\mathtt{CS}}/\eta_{\mathtt{U}}$ is changed from $1.E0$ to $1.E-1$. In Figure 1 we exhibit plots

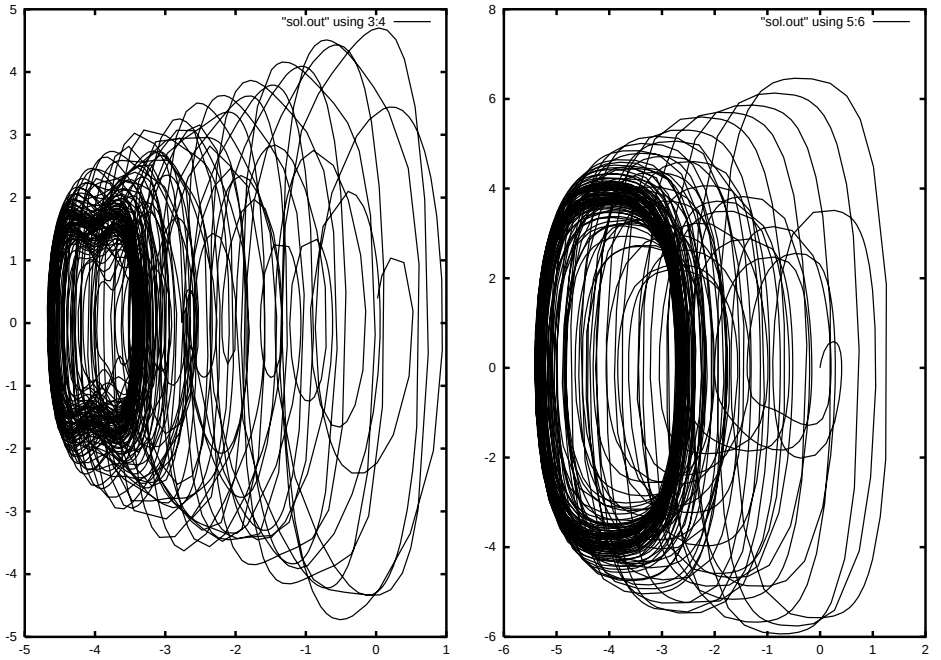


FIG. 1. *Plots of solution components* $(x_1, \dot{x}_1)$ *and* $(x_2, \dot{x}_2)$.

of the solution components $(x_1, \dot{x}_1)$ and $(x_2, \dot{x}_2)$. In Figure 2 we plot the four largest instantaneous Lyapunov exponents versus time. These instantaneous exponents are simply the diagonal elements of $\tilde{A}$ (see (4.3)), the coefficient matrix for R. Notice the variation of the exponents about zero which shows a lack of hyperbolicity and makes this a difficult problem for obtaining good shadowing distances. In fact, for $\theta = 0$ we were not able to shadow the approximate solution.

9. Conclusions. In this paper we devised a new approach to posteriori error analysis for approximate solutions to IVP ODEs using a shadowing lemma-type approach. Using an improved fixed point theorem and a reformulation of the differential equations we were able to obtain improved shadowing results. The cost of the numerical method for providing these global error estimates is small relative to the cost of approximating the differential equations. In particular, the method is $O(M)$ in both time and memory, and we found that the cost of providing the global error estimate is a fraction of the computation time. Any improvements to the estimates in section 6 would translate into improvements in the global error estimates. The use of different local error tolerances for different components of the differential equations, made possible by the reformation of the differential and taken advantage of with the improved fixed point result, allow for improved shadowing distances without severe degradation in the average time step. In all three examples there was improvement in the shadowing distance with the use of componentwise error control. The most substantial improvement was in Example 8.2, the Lorenz equation (one positive, one zero, and one negative Lyapunov exponent), which is a good fit to our prototype example (see (5.9)–(5.11) and the discussion thereafter), while in Example 8.1 (two negative exponents) and Example 8.3 (several finite time exponents oscillating about zero) we had some improvement.

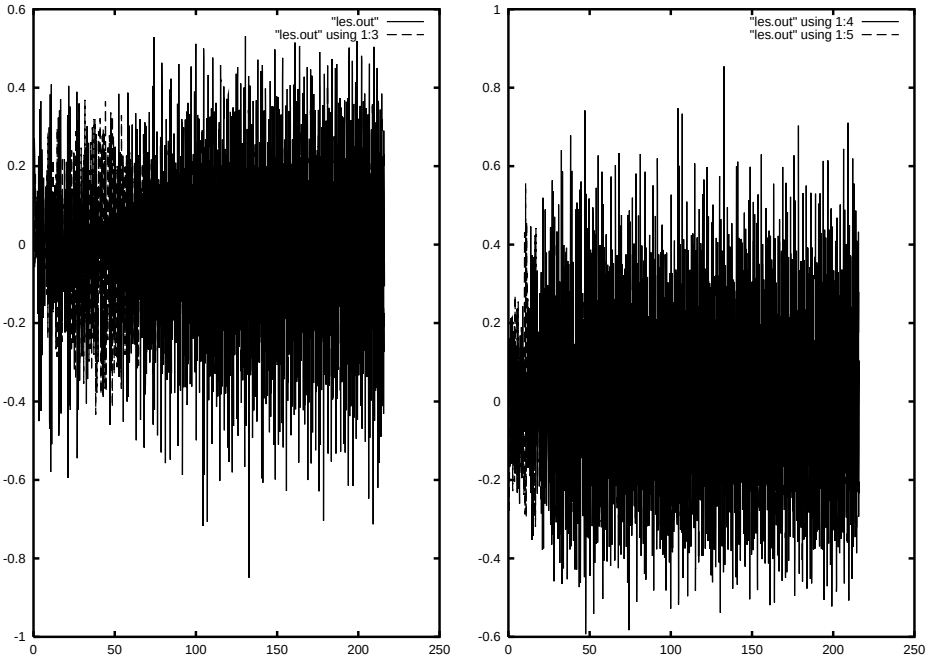


FIG. 2. *Plots of first two and second two instantaneous Lyapunov exponents versus time.*

Acknowledgments. We are grateful to Sebastian Reich for interesting discussions and to the referees for several helpful remarks.

REFERENCES

[1] B. BENETTIN AND A. GIORGILLI, *On the Hamiltonian interpolation of near the identity symplectic mappings with application to symplectic integration algorithms*, J. Statist. Phys., 74 (1994), pp. 1117–1143.

[2] W.-J. BEYN, *On the numerical approximation of phase portraits near stationary points*, SIAM J. Numer. Anal., 24 (1987), pp. 1095–1113.

[3] R. W. BRANKIN, I. GLADWELL, AND L. F. SHAMPINE, *RKSUITE: A Suite of Runge-Kutta Codes for the Initial Value Problem for ODEs*, Softreport 91-1, Mathematics Department, Southern Methodist University, Dallas, TX, 1991.

[4] S.-N. CHOW, X.-B. LIN, AND K. J. PALMER, *A shadowing lemma with applications to semilinear parabolic equations*, SIAM J. Math. Anal., 20 (1989), pp. 547–557.

[5] S. N. CHOW AND K. J. PALMER, *On the numerical computation of orbits of dynamical systems: The higher dimensional case*, J. Complexity, 8 (1992), pp. 398–423.

[6] B. A. COOMES, *Shadowing orbits of ordinary differential equations on invariant submanifolds*, Trans. Amer. Math. Soc., 349 (1997), pp. 203–216.

[7] B. A. COOMES, H. KOCAK, AND K. J. PALMER, *Rigorous computational shadowing of orbits of ordinary differential equations*, Numer. Math., 69 (1995), pp. 401–421.

[8] S. DAWSON, C. GREBOGI, T. SAUER, AND J. A. YORKE, *Obstructions to shadowing when a Lyapunov exponent fluctuates about zero*, Phys. Rev. Lett., 73 (1994), pp. 1927–1930.

[9] L. DIECI AND E. S. VAN VLECK, *Computation of orthonormal factors for fundamental solution matrices*, Numer. Math., 83 (1999), pp. 599–620.

[10] U. DRESSLER, *Symmetry property of the Lyapunov spectra of a class of dissipative dynamical systems with viscous damping*, Phys. Rev. A, 38-4 (1988), pp. 2103–2109.

[11] G. FAIRWEATHER, private communication, 1999.

[12] J. E. FRANKE AND J. F. SELGRADE, *Hyperbolicity and chain recurrence*, J. Differential Equation, 26 (1977), pp. 27–36.

[13] J. E. FRANKE AND J. F. SELGRADE, *A computer method for verification of asymptotically stable*

periodic orbits, SIAM J. Math. Anal., 10 (1979), pp. 614–628.
[14] G. H. Golub and C. F. Van Loan, *Matrix Computations*, The Johns Hopkins University Press, Baltimore, MD, 1989.
[15] E. Hairer, *Backward error analysis of numerical integrators and symplectic methods*, Ann. Numer. Math., 1 (1994), pp. 107–132.
[16] E. Hairer and C. Lubich, *The life-span of backward error analysis for numerical integrators*, Numer. Math., 76 (1997), pp. 441–462.
[17] E. Hairer, S. P. Norsett, and G. Wanner, *Solving Ordinary Differential Equations* I: *Non-stiff Problems*, Springer-Verlag, Berlin, 1987.
[18] S. Hammel, J. A. Yorke, and C. Grebogi, *Do numerical orbits of chaotic dynamical processes represent true orbits?*, J. Complexity, 3 (1987), pp. 136–145.
[19] S. Hammel, J. A. Yorke, and C. Grebogi, *Numerical orbits of chaotic processes represent true orbits*, Bull. Amer. Math. Soc., 19 (1988), pp. 465–470.
[20] L. V. Kantorovich and G. P. Akilov, *Functional Analysis*, 2nd ed., Pergamon Press, Elmsford, NY, 1982.
[21] G. Meurant, *A review on the inverse of symmetric tridiagonal and block tridiagonal matrices*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 707–728.
[22] K. J. Palmer, *Shadowing and Silnikov chaos*, Nonlinear Anal., 27 (1996), pp. 1075–1093.
[23] S. Reich, *Backward error analysis for numerical integrators*, SIAM J. Numer. Anal., 36 (1998), pp. 1549–1570.
[24] T. Sauer and J. A. Yorke, *Rigorous verification of trajectories for computer simulations of dynamical systems*, Nonlinearity, 4 (1991), pp. 961–979.
[25] A. M. Stuart and A. R. Humphries, *Dynamical Systems and Numerical Analysis*, Cambridge University Press, London, 1996.
[26] E. S. Van Vleck, *Numerical shadowing near hyperbolic trajectories*, SIAM J. Sci. Comput., 16 (1995), pp. 1177–1189.

SIAM J. SCI. COMPUT.
Vol. 22, No. 3, pp. 802–821
© 2000 Society for Industrial and Applied Mathematics

AN EFFICIENT STOCHASTIC ALGORITHM FOR STUDYING COAGULATION DYNAMICS AND GELATION PHENOMENA*

ANDREAS EIBECK† AND WOLFGANG WAGNER†

Abstract. A new class of efficient stochastic algorithms for the numerical treatment of coagulation processes is proposed. The algorithms are based on the introduction of fictitious jumps combined with an acceptance-rejection technique for distributions depending on particle size. The increased efficiency is demonstrated by numerical experiments. In particular, gelation phenomena are studied.

Key words. coagulation equation, stochastic particle method, simulation algorithm, gelation phenomena

AMS subject classifications. 65C05, 82C22, 60K35

PII. S1064827599353488

1. Introduction.

The continuous coagulation equation

$$\frac{\partial}{\partial t}\,c(t,x) = \frac{1}{2}\int_0^x K(x-y,y)\;c(t,x-y)\;c(t,y)\;dy - \int_0^\infty K(x,y)\;c(t,x)\;c(t,y)\;dy \tag{1.1}$$

with initial condition

$$c(0,x) = c_0(x) \geq 0 \tag{1.2}$$

describes the time evolution of the average concentration of particles of size $x > 0$. Here the nonnegative and symmetric function $K(x,y)$ denotes the coagulation rate of clusters of size x and y. According to (1.1), the concentration $c(x,t)$ can increase by coagulation of clusters of size $y < x$ and $x-y$ (first term) and decrease by coagulation of an x-cluster with a cluster of any size y (second term).

If the clusters can take only sizes $k = 1, 2, \ldots$, then the coagulation dynamics is governed by a discrete version of equation (1.1), where all integrals are replaced by sums. The discrete coagulation equation

$$\frac{\partial}{\partial t}\,c(t,k) = \frac{1}{2}\sum_{j=1}^{k-1} K(k-j,j)\,c(t,k-j)\,c(t,j) - \sum_{j=1}^{\infty} K(k,j)\,c(t,k)\,c(t,j) \tag{1.3}$$

was first published in [23] and is referred to as Smoluchowski's coagulation equation. Both the continuous and the discrete equations have a wide range of applications, e.g., in astrophysics, biology, chemistry, and meteorology (see the survey papers [6] and [2]).

Stochastic particle systems related to the coagulation equation were introduced by several authors (cf. [18], [11], [17]). They are of the form

$$x_i(t)\,, \quad i = 1, \ldots, n(t)\,, \qquad t \geq 0\,, \tag{1.4}$$

*Received by the editors March 22, 1999; accepted for publication (in revised form) March 5, 2000; published electronically August 31, 2000. This work was supported by Deutsche Forschungsgemeinschaft (Schwerpunktprogramm "Interagierende stochastische Systeme von hoher Komplexität").
http://www.siam.org/journals/sisc/22-3/35348.html
†Weierstrass Institute for Applied Analysis and Stochastics, Mohrenstraße 39, D–10117 Berlin, Germany (eibeck@wias-berlin.de, wagner@wias-berlin.de).

where $x_i(t)$ represents the size of particle i at time t and $n(t)$ is the number of particles in the system. The initial system consisting of a large number $n(0)$ of particles is supposed to approximate the initial condition (1.2). A random dynamics is defined such that the system at later times approximates the solution to (1.1). The stochastic approach to coagulation was reviewed in [2]. Recently, rigorous results concerning the weak law of large numbers for the relevant stochastic particle systems with general kernels have been obtained (see [2, Problem 10(a)], [13], [14], [15], [19], [7]). Besides the derivation of the coagulation equation, stochastic particle systems play a significant role in numerical algorithms for that equation (see [12], [5]). We refer to [21] for a survey on Monte Carlo methods and to [20] concerning recent applications.

The purpose of this paper is to introduce a new class of efficient stochastic algorithms for the numerical treatment of the coagulation equation. The idea was mentioned in [7], where we studied the convergence problem. Here we give a detailed description of the numerical procedure and apply it to the investigation of some interesting phenomena.

The paper is organized as follows. In section 2 we describe the new class of algorithms. Markov processes with fictitious jumps are introduced, which are based on appropriate majorant kernels. An efficient acceptance-rejection procedure for the generation of the relevant probability distributions is proposed. In section 3 we apply the numerical technique to the study of coagulation dynamics. For several coagulation kernels we investigate approximation properties and gelation phenomena. In section 4 we discuss the results and mention some directions for further study.

2. Description of the algorithm.

2.1. Markov processes with fictitious jumps. We introduce a sequence of jump processes

$$U^N(t)\,, \qquad t \geq 0\,, \qquad N = 1, 2 \dots\,, \tag{2.1}$$

taking values in the space of discrete measures

$$\mathcal{S}^N = \left\{ p = \frac{1}{N} \sum_{i=1}^{n} \delta_{x_i}\,, \quad x_i > 0\,, \quad n = 1, 2, \dots \right\},$$

where δ denotes the Dirac measure. The process (2.1) is represented by a system of particles (1.4),

$$U^N(t, dx) = \frac{1}{N} \sum_{i=1}^{n(t)} \delta_{x_i(t)}(dx)\,. \tag{2.2}$$

For $\Phi \in C_b(\mathcal{S}^N)$ the infinitesimal generator of the process is defined as

$$\mathcal{K}^N \Phi(p) = \frac{1}{2N} \sum_{1 \leq i \neq j \leq n} \Big[\Phi(J(p, i, j)) - \Phi(p) \Big] K(x_i, x_j)\,, \qquad p \in \mathcal{S}^N\,, \tag{2.3}$$

where

$$J(p, i, j) = p + \frac{1}{N} (\delta_{x_i + x_j} - \delta_{x_i} - \delta_{x_j})\,. \tag{2.4}$$

The initial condition is assumed to be deterministic and such that (cf. (1.2))

$$\lim_{N\to\infty} \int_0^\infty \varphi(x)\, U^N(0, dx) = \int_0^\infty \varphi(x)\, c_0(x)\, dx \tag{2.5}$$

for a sufficiently wide class of test functions φ. In particular, for $\varphi = 1$, one obtains

$$n(0) \sim N \int_0^\infty c_0(x)\, dx\,. \tag{2.6}$$

Convergence of $U^N(t)$ (as $N \to \infty$) to the solution of a measure-valued version of (1.1) was studied in [19].

We will describe a class of simulation algorithms related to the stochastic systems determined by (2.3)–(2.5). For that purpose, the generator (2.3) is rewritten in the usual form of a jump process generator [8, Chapter 4, section 2]

$$\mathcal{K}^N \Phi(p) = \int_{\mathcal{S}^N} \Big[\Phi(q) - \Phi(p)\Big]\, \mathcal{Q}(p, dq)\,, \tag{2.7}$$

where

$$\begin{aligned} &\mathcal{Q}(p, dq) \\ &\quad = \frac{1}{2N} \sum_{1\le i\ne j\le n} \Big\{ K(x_i, x_j)\, \delta_{J(p,i,j)}(dq) + \Big[\hat{K}(x_i, x_j) - K(x_i, x_j)\Big]\, \delta_p(dq) \Big\} \end{aligned} \tag{2.8}$$

and $\hat{K}$ is an appropriate majorant kernel satisfying

$$K(x, y) \le \hat{K}(x, y) \qquad \forall\, x, y > 0\,. \tag{2.9}$$

The pathwise behavior of a jump process with the infinitesimal generator (2.7)–(2.8) is as follows. Coming to a state $p \in \mathcal{S}^N$, the process stays there for a random waiting time $\tau(p)$, which has an exponential distribution with the parameter

$$\varrho(p) = \mathcal{Q}(p, \mathcal{S}^N) = \frac{1}{2N} \sum_{1\le i\ne j\le n} \hat{K}(x_i, x_j)\,, \tag{2.10}$$

i.e.,

$$\mathrm{Prob}\,\{\tau(p) \ge s\} = \exp(-\varrho(p)\, s)\,, \qquad s \ge 0\,. \tag{2.11}$$

After the time $\tau(p)$, the process jumps into a state $q \in \mathcal{S}^N$, which is distributed according to the jump distribution

$$\begin{aligned} &\varrho(p)^{-1}\, \mathcal{Q}(p, dq) \\ &\quad = \sum_{1\le i\ne j\le n} \frac{\hat{K}(x_i, x_j)}{2N\, \varrho(p)} \left\{ \frac{K(x_i, x_j)}{\hat{K}(x_i, x_j)}\, \delta_{J(p,i,j)}(dq) + \left[1 - \frac{K(x_i, x_j)}{\hat{K}(x_i, x_j)}\right] \delta_p(dq) \right\}. \end{aligned} \tag{2.12}$$

Distribution (2.12) is represented as a superposition of simpler distributions.

This description leads to the following algorithm for generating trajectories of the process.

0. Construct the initial state $U^N(0) = p \in \mathcal{S}^N$ in accordance with (2.5).

1. Wait in state p during a random time step $\tau(p)$ distributed according to (2.11).
2. Choose a pair i, j according to the index distribution (cf. (2.12))

$$\frac{\hat{K}(x_i, x_j)}{2N\,\varrho(p)}\,, \qquad 1 \le i \ne j \le n\,. \tag{2.13}$$

3. With probability

$$1 - \frac{K(x_i, x_j)}{\widehat{K}(x_i, x_j)}\,, \tag{2.14}$$

perform a fictitious jump (cf. (2.12)), i.e., stay in state p, and go to step 1.
4. Replace p by the new state $J(p, i, j)$ (cf. (2.4)), i.e., remove the clusters x_i and x_j and add the cluster $x_i + x_j$, and go to step 1.

Note the mass conservation property

$$\int_0^\infty x\,U^N(t, dx) = \int_0^\infty x\,U^N(0, dx)\,, \qquad t \ge 0\,, \tag{2.15}$$

which is a consequence of (2.4).

The distribution of the Markov process and its convergence behavior do not depend on the choice of the majorant kernel $\hat{K}$ in (2.9), since those properties are completely determined by the generator (2.3) and the initial condition (2.5). However, the choice of this function is of importance for numerical purposes, since it provides different ways of generating trajectories of the process and thus influences the efficiency of the simulation procedure. We illustrate this by two examples.

Example 2.1. The choice

$$\hat{K}(x, y) = K(x, y)\,, \qquad x, y > 0\,, \tag{2.16}$$

corresponds to the direct "physical" simulation of the process (see [12], [5]). No fictitious jumps occur (cf. (2.14)). This is the smallest possible majorant kernel satisfying (2.9). Thus, the time steps, having the average value $\varrho(p)^{-1}$ (cf. (2.11)), are as large as possible. However, the calculation of (2.10) and the generation of the distribution (2.13) are very time consuming if n is large and K has a complicated structure.

Example 2.2. The choice

$$\hat{K}(x, y) = K_{max}(p) = \max_{i,j=1,\dots,n} K(x_i, x_j)\,, \qquad x, y > 0\,, \tag{2.17}$$

represents the other extreme case compared with Example 2.1, providing the largest (reasonable) majorant kernel satisfying (2.9) (see, e.g., [12], [10], [21]). In this case the calculation of (2.10) and the generation of the distribution (2.13) are extremely simple. Indeed, the waiting time parameter takes the form

$$\varrho(p) = K_{max}(p)\,\frac{n\,(n-1)}{2N}\,, \tag{2.18}$$

and the index distribution (2.13) is uniform. However, the time steps are much smaller compared with Example 2.1, and many fictitious jumps occur (cf. (2.14)). This leads to low efficiency of the algorithm in situations where K is unbounded and clusters of significantly different sizes are contained in the system.

2.2. Choice of the majorant kernel. First we specify the algorithm from the previous subsection for the majorant kernel (cf. (2.9))

$$\hat{K}(x,y) = C\, x^{\varepsilon_1}\, y^{\varepsilon_2}\,, \qquad C>0\,, \quad \varepsilon_1\,, \varepsilon_2 \geq 0\,. \tag{2.19}$$

The waiting time parameter (2.10) takes the form

$$\varrho(p) = \frac{C}{2N} \sum_{i=1}^{n} x_i^{\varepsilon_1} \sum_{j=1,\ldots,n,\, j\neq i} x_j^{\varepsilon_2} = \frac{C}{2N} \left[\sum_{i=1}^{n} x_i^{\varepsilon_1} \sum_{j=1}^{n} x_j^{\varepsilon_2} - \sum_{i=1}^{n} x_i^{\varepsilon_1+\varepsilon_2} \right]. \tag{2.20}$$

The index distribution (2.13) is generated using the acceptance-rejection technique (cf., e.g., [9, section 3.6]). To this end the events $i=j$ are added to the state space and the zero probability for $i=j$ in (2.13) is replaced by an appropriate term proportional to $x_i^{\varepsilon_1}\, x_j^{\varepsilon_2}$. Then the index distribution takes the form

$$\frac{x_i^{\varepsilon_1}}{\sum_{l=1}^{n} x_l^{\varepsilon_1}}\, \frac{x_j^{\varepsilon_2}}{\sum_{l=1}^{n} x_l^{\varepsilon_2}}\,, \qquad i,j=1,\ldots,n\,,$$

so that the indices i and j are independent. The simulation procedure is as follows.

1. Generate i according to

$$\frac{x_i^{\varepsilon_1}}{\sum_{l=1}^{n} x_l^{\varepsilon_1}}\,, \qquad i=1,\ldots,n\,. \tag{2.21}$$

2. Generate j according to

$$\frac{x_j^{\varepsilon_2}}{\sum_{l=1}^{n} x_l^{\varepsilon_2}}\,, \qquad j=1,\ldots,n\,. \tag{2.22}$$

3. If $i=j$, then go to step 1.

A method for generating random variables with distributions of the form (2.21), (2.22) will be introduced in the next subsection.

Next we consider the majorant kernel

$$\hat{K}(x,y) = \hat{K}_1(x,y) + \cdots + \hat{K}_L(x,y)\,, \qquad L \geq 2\,, \tag{2.23}$$

where the components $\hat{K}_k\,,\ k=1,\ldots,L\,,$ are of the form (2.19). The waiting time parameter (2.10) is

$$\varrho(p) = \varrho_1(p) + \cdots + \varrho_L(p)\,,$$

where

$$\varrho_k(p) = \frac{1}{2N} \sum_{1\leq i\neq j\leq n} \hat{K}_k(x_i,x_j)\,, \qquad k=1,\ldots,L\,.$$

The index distribution (2.13) takes the form

$$\sum_{k=1}^{L} \frac{\varrho_k(p)}{\varrho(p)}\, \frac{\hat{K}_k(x_i,x_j)}{2N\,\varrho_k(p)}\,, \qquad 1\leq i\neq j\leq n\,.$$

Thus, one first chooses a number k according to the probabilities

$$\frac{\varrho_k(p)}{\varrho(p)}\,, \qquad k=1,\dots,L\,, \tag{2.24}$$

and then generates i,j according to the distribution

$$\frac{\hat K_k(x_i,x_j)}{2N\,\varrho_k(p)}\,, \qquad 1\le i\ne j\le n\,,$$

which is of the form (2.19).

Example 2.3. Consider the case

$$\hat K(x,y)=C\,(x+y)\,, \qquad x,y>0\,. \tag{2.25}$$

This kernel has the form (2.23). One obtains the waiting time parameter

$$\varrho(p)=C\,m_1\,(n-1)\,, \tag{2.26}$$

where $m_1=\frac{1}{N}\sum_{i=1}^n x_i$. Note that

$$m_1=\int_0^\infty x\,U^N(0,dx) \tag{2.27}$$

according to the mass conservation property (2.15). The index i is generated according to the distribution

$$\frac{x_i}{N\,m_1}\,, \qquad i=1,\dots,n\,. \tag{2.28}$$

The index j is generated uniformly on the set $\{k=1,\dots,n\,,\quad k\ne i\}$ avoiding the acceptance-rejection step. The random choice according to (2.24) is omitted, since the result of the interaction (2.4) does not depend on the order of the indices.

Example 2.4. Consider the case

$$\hat K(x,y)=C\,x\,y\,, \qquad x,y>0\,. \tag{2.29}$$

This kernel has the form (2.19) with $\varepsilon_1=\varepsilon_2=1$. From (2.20) one obtains the waiting time parameter

$$\varrho(p)=\frac{C\,N\,m_1^2}{2}-\frac{C}{2\,N}\sum_{i=1}^n x_i^2\,,$$

where m_1 is defined in (2.27). Note that $n=1$ implies $x_1=N\,m_1$ and $\varrho(p)=0$. The indices i,j are generated independently, according to the distribution (2.28). They are rejected if $i=j$.

2.3. Generation of a size dependent distribution. It remains to describe a procedure for generating samples of a random variable with a distribution of the form (cf. (2.21), (2.22), (2.28))

$$\frac{x_j^\varepsilon}{\sum_{l=1}^n x_l^\varepsilon}\,, \qquad j=1,\dots,n\,, \qquad \varepsilon>0\,. \tag{2.30}$$

A direct application of the inverse transform method (cf., e.g., [9, section 3.2.4]) is rather time consuming when n is large. The same applies to the straightforward version of the acceptance-rejection technique, since the particle sizes $x_1, \ldots, x_n$ vary significantly.

In order to generate the distribution (2.30) efficiently, we introduce a group structure of the particle system. The particles are organized in γ groups, i.e., their sizes are denoted by

$$y_{j,k}\,, \qquad j = 1, \ldots, \gamma\,, \quad k = 1, \ldots, \alpha_j\,,$$

so that

$$b_{j-1} < y_{j,k} \le b_j \qquad \forall j = 1, \ldots, \gamma\,, \quad k = 1, \ldots, \alpha_j\,,$$

where

$$0 =: b_0 < b_1 < \cdots < b_\gamma \tag{2.31}$$

and (cf. (2.27))

$$N\, m_1 \le b_\gamma\,. \tag{2.32}$$

Note that $N\, m_1$ is the upper bound for the particle size.

The distribution (2.30) is rewritten as

$$\frac{y^\varepsilon_{j,k}}{c}\,, \qquad j = 1, \ldots, \gamma\,, \quad k = 1, \ldots, \alpha_j\,, \tag{2.33}$$

where the normalizing constant is

$$c = \sum_{l=1}^{n} x^\varepsilon_l = \sum_{j=1}^{\gamma} \sum_{k=1}^{\alpha_j} y^\varepsilon_{j,k}\,. \tag{2.34}$$

The representation (2.33), (2.34) suggests the following simulation procedure using the inverse transform method at step 1 and the acceptance-rejection technique at steps 2 and 3.

1. Choose the group index according to the probabilities

$$P_j = \frac{1}{c} \sum_{k=1}^{\alpha_j} y^\varepsilon_{j,k}\,, \qquad j = 1, \ldots, \gamma\,. \tag{2.35}$$

2. Choose the particle index $k = 1, \ldots, \alpha_j$ uniformly within the group j.
3. The particle index is accepted with probability

$$\frac{y^\varepsilon_{j,k}}{b^\varepsilon_j}\,. \tag{2.36}$$

Otherwise, go to step 2.

With

$$b_j = \beta^{j-1}\,, \qquad j = 1, 2, \ldots\,,$$

where $\beta > 1$, the acceptance probability (2.36) for groups $j \geq 2$ is bounded from below by

$$\frac{1}{\beta^{\varepsilon}}\,. \tag{2.37}$$

The necessary number of groups $\gamma = \gamma(N, \beta)$ is obtained from the estimate (cf. (2.32))

$$\beta^{\gamma-2} < N\,m_1 \leq \beta^{\gamma-1} \qquad \text{or} \qquad \gamma - 2 < \frac{\log(N\,m_1)}{\log\beta} \leq \gamma - 1\,. \tag{2.38}$$

Remark 2.5. For $\beta \searrow 1$, the number of rejections decreases according to (2.37), leading to higher efficiency. On the other hand, the number of groups γ increases according to (2.38), and generating the distribution (2.35) becomes more and more time consuming.

3. Applications to coagulation dynamics. According to (2.2), functionals of the solution of (1.1) are approximated as

$$\int_0^\infty \varphi(x)\,c(t,x)\,dx \sim \int_0^\infty \varphi(x)\,U^N(t,dx) = \frac{1}{N}\sum_{i=1}^{n(t)} \varphi(x_i(t))\,, \qquad t \geq 0, \tag{3.1}$$

where φ is some test function. Many functionals of interest are expressed via moments of the solution

$$m_\delta(t) = \int_0^\infty x^\delta\,c(t,x)\,dx\,, \qquad \delta \geq 0\,. \tag{3.2}$$

In particular, $m_0(t)$ is the total concentration of particles, $m_1(t)$ is the total mass of the system, and

$$S(t) = \frac{m_1(t)}{m_0(t)} = \frac{\int_0^\infty x\,c(t,x)\,dx}{\int_0^\infty c(t,x)\,dx}$$

is the average particle size. Moments (3.2) are approximated according to (3.1) as

$$m_\delta(t) \sim \frac{1}{N}\sum_{i=1}^{n(t)} x_i(t)^\delta\,. \tag{3.3}$$

In the following we restrict our considerations to the discrete equation (1.3) with monodispersal initial condition

$$c(0,1) = 1\,, \qquad c(0,k) = 0\,, \quad k = 2,3,\dots\,. \tag{3.4}$$

In this case some explicit solutions are available (cf., e.g., [2, section 2.2]). This is convenient for validating the algorithm. The solution represents the concentration of particles of given size and is approximated as

$$c(t,k) \sim \frac{1}{N}\,\#\{i\,:\,x_i(t) = k\}\,, \qquad k = 1,2,\dots\,, \quad t \geq 0\,. \tag{3.5}$$

According to (3.4) we start with the particle system (cf. (2.5), (2.6))

$$x_i(0) = 1\,, \qquad i = 1,\dots,N\,. \tag{3.6}$$

Confidence bands with confidence level 99.9% are constructed using R independent runs, where $R\,N = 10^7$.

3.1. Explicit solutions. The special case of constant coagulation rate and initial condition (3.4) was solved in the original paper [23]. For

$$K(i,j) = 1$$

one obtains

$$c(t,k) = \frac{4}{(2+t)^2}\left(\frac{t}{2+t}\right)^{k-1}, \qquad t \geq 0\,, \quad k = 1,2,\dots\,,$$

and

$$m_0(t) = \frac{2}{2+t}\,, \qquad m_1(t) = 1\,, \qquad m_2(t) = 1+t\,, \qquad t \geq 0\,.$$

In the special case

$$K(i,j) = i + j \tag{3.7}$$

one obtains

$$c(t,k) = e^{-t}\,\frac{k^{k-1}}{k!}\,(1-e^{-t})^{k-1}\,e^{-k(1-e^{-t})}\,, \qquad t \geq 0\,, \quad k = 1,2,\dots\,, \tag{3.8}$$

and

$$m_0(t) = e^{-t}\,, \qquad m_1(t) = 1\,, \qquad m_2(t) = e^{2t}\,, \qquad t \geq 0\,.$$

In the special case

$$K(i,j) = i\,j \tag{3.9}$$

the global solution of (1.3) is given by (cf. [16, Theorem 2.2])

$$c(t,k) = \begin{cases} \frac{k^{k-2}}{k!}\,t^{k-1}\,\exp(-kt) & \text{if } 0 \leq t \leq 1\,, \\ \\ \frac{k^{k-2}}{k!}\,\exp(-k)\,t^{-1} & \text{if } 1 < t\,. \end{cases} \tag{3.10}$$

For the total concentration one obtains

$$m_0(t) = \begin{cases} 1 - \frac{t}{2} & \text{if } 0 \leq t \leq 1\,, \\ \frac{1}{2t} & \text{if } 1 < t\,. \end{cases}$$

The total mass behaves like

$$m_1(t) = \begin{cases} 1 & \text{if } 0 \leq t \leq 1\,, \\ \frac{1}{t} & \text{if } 1 < t\,, \end{cases} \tag{3.11}$$

and the second moment takes the form

$$m_2(t) = \frac{1}{1-t}\,, \qquad 0 \leq t < 1\,.$$

Remark 3.1. Note that the function

$$\tilde{c}(t,x) = \mu\,c(\mu\,\sigma\,t,x)\,, \qquad t \geq 0\,, \quad x > 0\,, \quad \mu,\sigma \geq 0\,,$$

solves (1.1) with the kernel $\tilde{K} = \sigma\,K$ instead of K and initial condition $\tilde{c}_0 = \mu\,c_0$ instead of c_0.

3.2. Examples of coagulation kernels. The kernel K in (1.3) represents physical properties of the medium governing the coagulation process. For example, when coagulation of spherical particles is controlled by Brownian diffusion, Smoluchowski [23] derived the kernel

$$K(i,j) = (i^{\frac{1}{3}} + j^{\frac{1}{3}})\,(i^{-\frac{1}{3}} + j^{-\frac{1}{3}})\,.$$

For spherical particles in a laminar shear field the coagulation kernel takes the form

$$K(i,j) = \kappa\,(i^{\frac{1}{3}} + j^{\frac{1}{3}})^3\,.$$

This kernel has also been used in connection with coagulation processes in turbulent flows (cf. [20]). In this case the constant κ depends on physical quantities like the mean rate of dissipation of kinetic energy and kinematic viscosity. More examples of practically relevant coagulation kernels can be found in [6, section 4.3].

In our numerical investigations we use three coagulation kernels. One arises in applications; the other two are model kernels, for which some analytical results are available. This allows us to validate the numerical method and to study some special phenomena.

The first kernel is

$$K_1(i,j) = \frac{1}{4}\,(i^{\frac{1}{3}} + j^{\frac{1}{3}})^3\,. \tag{3.12}$$

Note that

$$K_1(u\,i, u\,j) = u\,K_1(i,j) \qquad \forall u > 0\,. \tag{3.13}$$

According to

$$\frac{1}{4}\,(i+j) \le K_1(i,j) \le i + j\,,$$

the majorant kernel (2.25) is used (cf. (2.9)).

The second kernel is

$$K_2^{(a)}(i,j) = \frac{2\,i^a\,j^a}{(i+j)^a - i^a - j^a}\,, \qquad a \in (1,2]\,. \tag{3.14}$$

For this kernel an explicit formula for some moments is known (see, e.g., [1]), namely

$$m_a(t) = \frac{1}{1-t}\,, \qquad t \in [0,1)\,. \tag{3.15}$$

Note that

$$K_2^{(a)}(u\,i, u\,j) = u^a\,K_2^{(a)}(i,j) \qquad \forall u > 0\,. \tag{3.16}$$

According to

$$K_2^{(a)}(i,j) \le \frac{2}{2^a - 2}\,i\,j\,, \tag{3.17}$$

the majorant kernel (2.29) is used. In case $a = 2$ there is equality in (3.17).

The third kernel is

$$K_3^{(a)}(i,j) = i^a\,j^a\,, \qquad a \in (0.5, 1]\,. \tag{3.18}$$

This kernel has the form (2.19) and the (trivial) majorant kernel (2.16) will be used. Note that

$$K_3^{(a)}(u\,i, u\,j) = u^{2a}\,K_3^{(a)}(i,j) \qquad \forall u > 0\,. \tag{3.19}$$

3.3. Approximation properties. To get a first impression of how the algorithm works, we consider the linear kernel (3.7). Here one takes $\hat{K} = K$ (cf. Example 2.3) and no fictitious jumps occur. Using a particle number $N = 10^6$ we calculate the solution $c(t,k)$ and some integrated functionals of the form

$$C(t,k) = \sum_{l \geq k} c(t,l) \tag{3.20}$$

for different values of size k. The results are displayed in Figure 1, where the analytic curves (cf. (3.8)) are represented by solid lines and the confidence bands by dotted lines. The components of the solution $c(t,k)$ are sufficiently well approximated up to $k = 200$. The time instants where these functions take their maximum can be clearly detected. The fluctuations grow with increasing size, since the values of the functionals become very small. Functionals of the form (3.20) are well approximated even up to $k = 2000$.

Next we compare the solution for the kernel (3.12) with the solution for the linear kernel (3.7). Since at time zero both kernels are identical in case of initial configuration (3.6), it is of interest to study the deviation of both solutions during the time evolution. The confidence bands for the solution corresponding to the kernel (3.12) are shown in Figure 2 and compared with the exact solutions for the kernel (3.7) (solid lines).

Finally we illustrate the convergence with respect to the initial particle number N for the kernel (3.14) with $a = 1.5$. We calculate the behavior of the moment $m_{1.5}(t)$ on the time interval $[0,1]$ via (3.3) and compare the results with the explicit expression (3.15). The results are displayed in Figure 3. For $N = 10^4$ the exact solution (solid line) is almost indistinguishable from the confidence band up to the time $t = 0.8$. For larger N the exact solution leaves the confidence band at later times.

3.4. Gelation phenomena. In the case of the multiplicative kernel (3.9) the total mass is not conserved (cf. (3.11)). This effect is interpreted as formation of infinite clusters and is called gelation (cf. [2], [15]). Our test kernels (3.12), (3.14), and (3.18) are homogeneous (cf. (3.13), (3.16), (3.19)) with exponents 1, a, and $2a$, respectively. Homogeneous kernels with exponents greater than 1 are expected to be gelling (see [2, section 2.3]).

In the finite particle system (1.4) the total mass is always conserved (cf. (2.15)). An appropriate indicator for gelation is the behavior of the largest component $M_1(t)$ in the system. For the multiplicative kernel (3.9), the particle with the largest size is of order N after the gelation time

$$t_g = \inf \{t > 0 \,:\, m_1(t) < m_1(0)\}$$

and of lower order before t_g (see [2, section 4.4]). The quantity $\frac{M_1(t)}{N}$ is convenient for studying general gelling kernels (see [2, section 5.2]) using the stochastic algorithm. Note that $\frac{M_1(t)}{N} \in [0,1]$ and $M_1(t) \leq M_1(t')\,,\ t \leq t'$.

As an example we consider the kernel (3.18) with $a = 0.7, 0.8, 0.9, 1.0$. The curves for the quantity $\frac{M_1(t)}{N}$ with $N = 10^4, 10^5, 10^6$ are shown by dotted, dashed, and solid lines, respectively, in Figure 4. The behavior of the second-largest component divided by N for $N = 10^4$ (dotted), $N = 10^5$ (dashed), and $N = 10^6$ (solid) is shown in Figure 5. For comparison the curves for both components and $N = 10^6$ are displayed together on the same scale in Figure 6, where, in addition, the dashed lines represent

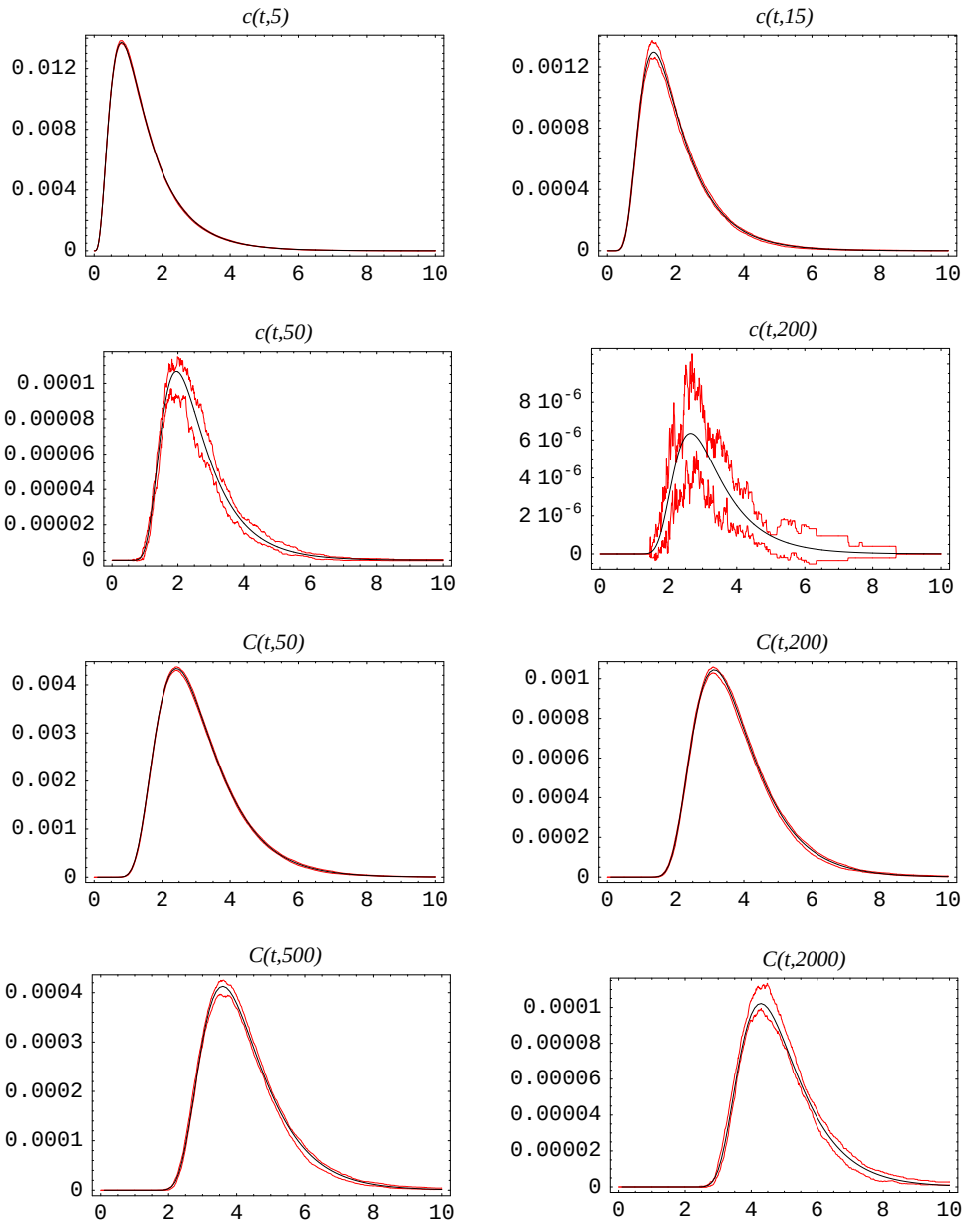


FIG. 1. *Concentration functionals* $c(t,k)$ *and* $C(t,k)$ *for the linear kernel* (3.7).

the total concentration $m_0(t)$. Analogous results for the kernel (3.14) with $a = 1.5$ are shown in Figures 7 and 8.

Next we consider the multiplicative kernel (3.9) and the equation (cf. [22])

$$\frac{\partial}{\partial t}\,\hat{c}(t,k) = \frac{1}{2}\sum_{j=1}^{k-1} j\,(k-j)\,\hat{c}(t,j)\,\hat{c}(t,k-j) \;-\; k\,\hat{c}(t,k)\,. \tag{3.21}$$

According to (3.11), equation (1.3) coincides with (3.21) up to the gelation point

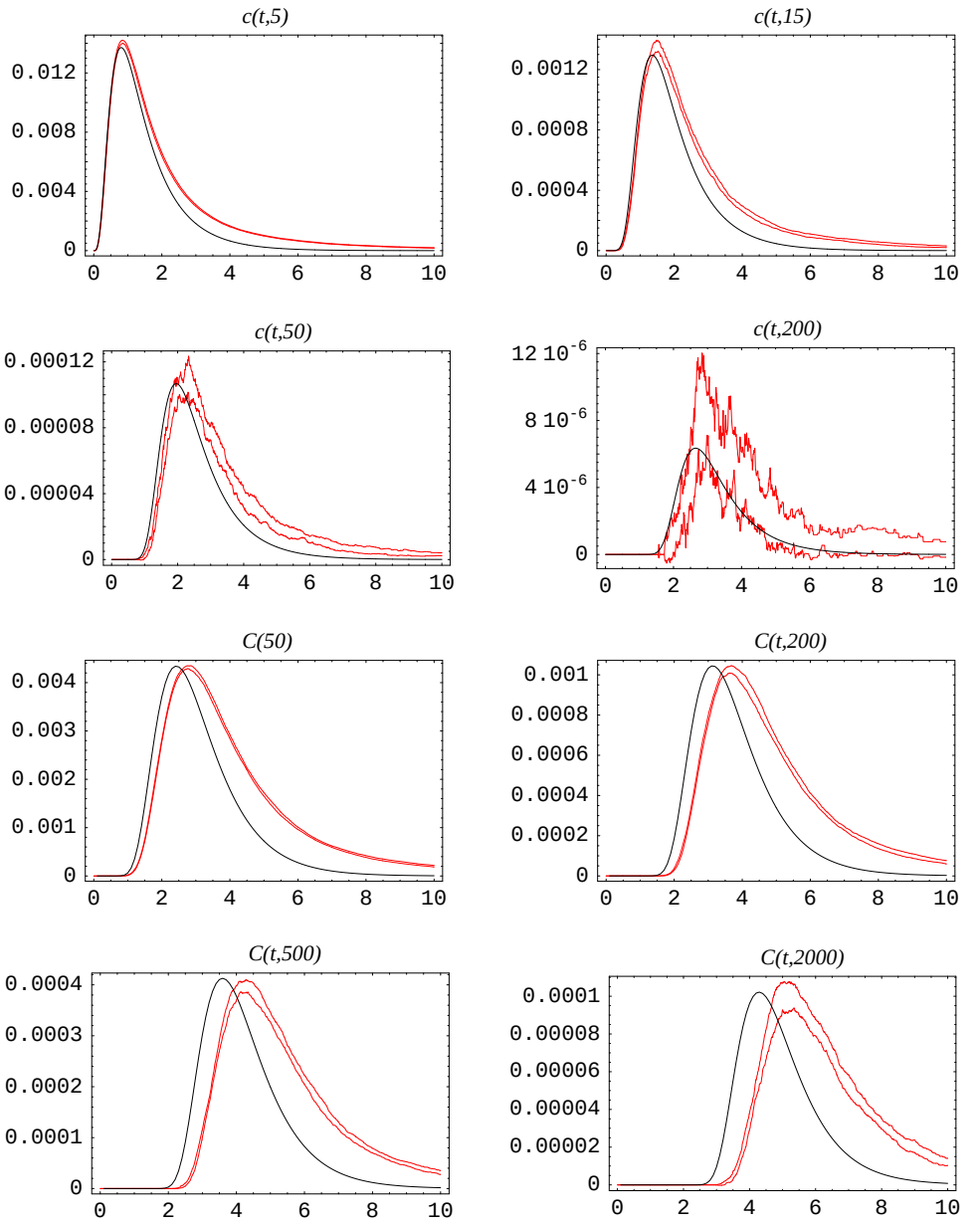


FIG. 2. *Concentration functionals* $c(t,k)$ *and* $C(t,k)$ *for the kernel* (3.12).

$t_g = 1$. The unique solution of (3.21) is (compare with (3.10))

$$\hat{c}(t,k) = \frac{k^{k-2}}{k!}\, t^{k-1}\, \exp(-kt)\,, \qquad t \geq 0\,. \tag{3.22}$$

The total mass has the form (compare with (3.11))

$$\widehat{m}_1(t) = \begin{cases} 1 & \text{if } 0 \leq t \leq 1\,, \\ \frac{t^*}{t} & \text{if } 1 < t\,, \end{cases} \tag{3.23}$$

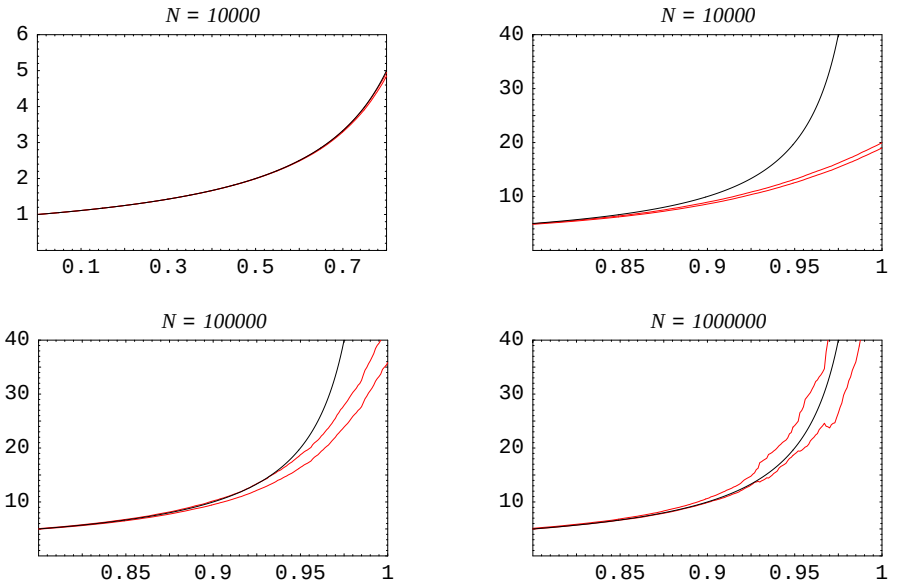


FIG. 3. *Moment* (3.15) *for the kernel* (3.14) *with* $a = 1.5$.

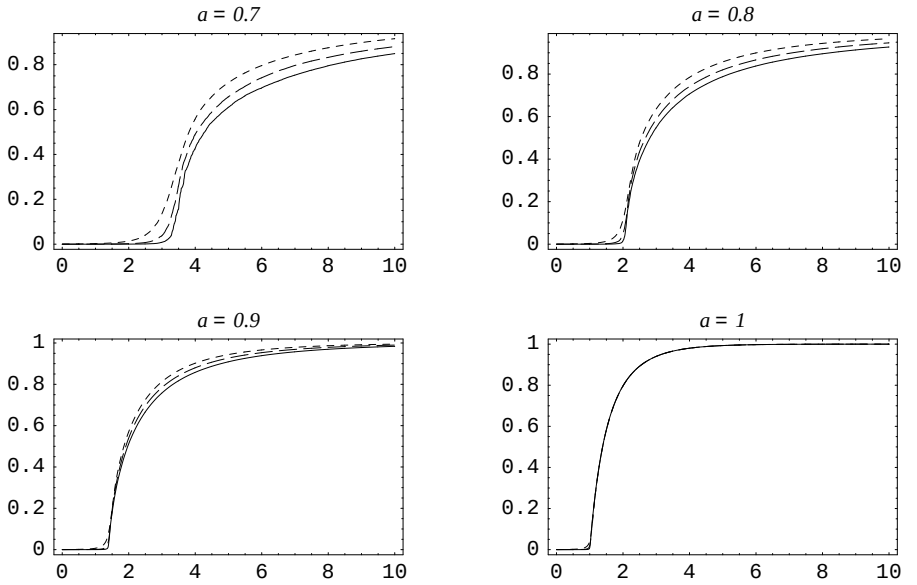


FIG. 4. *Largest component* $\frac{M_1(t)}{N}$ *for the kernel* (3.18) *and different* N.

where $t^* = t^*(t)$ is determined by the equation

$$t^* \exp(-t^*) = t \exp(-t)\,, \qquad t^* \in (0,1)\,, \quad t > 1\,.$$

The solution (3.22) describes the limiting behavior (cf. [4, Corollary 1]) of the stochastic particle system beyond the gelation point. The results of the stochastic algorithm for different concentration functionals are shown in Figure 9. In these

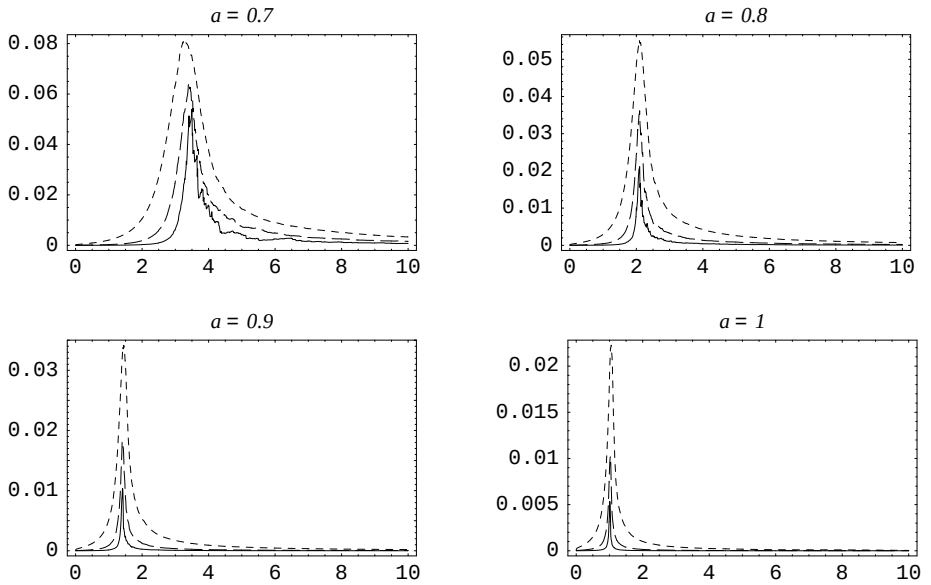


FIG. 5. *Second largest component* $\frac{M_2(t)}{N}$ *for the kernel* (3.18) *and different* N.

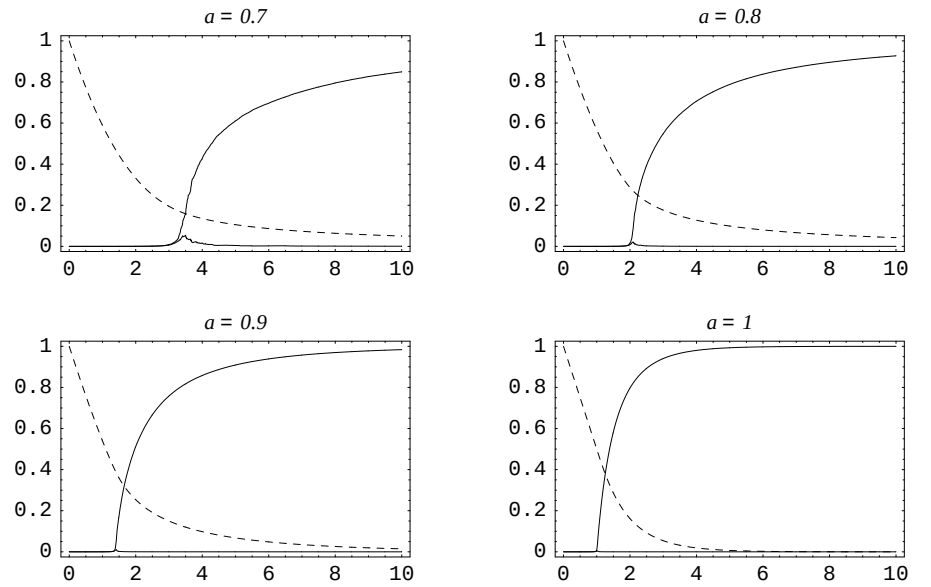


FIG. 6. *Two largest components and total concentration* $m_0(t)$ *for the kernel* (3.18).

calculations the initial number of particles is $N = 10^6$. The exact curves for (3.21) are shown by solid lines, the confidence bands by dotted lines, and the curves for the Smoluchowski equation (1.3) by dashed lines. The loss of total mass (cf. (3.23)) is approximated by the largest component of the particle system (see Figure 10)

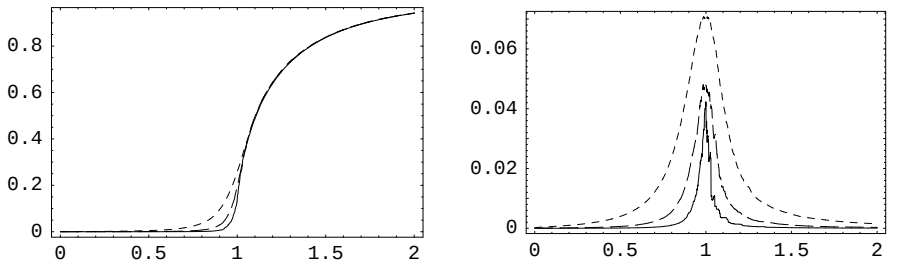


FIG. 7. *Largest and second largest components for the kernel* (3.14) *with* $a = 1.5$.

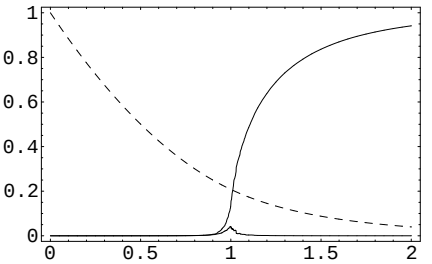


FIG. 8. *Two largest components and* $m_0(t)$ *for the kernel* (3.14) *with* $a = 1.5$.

illustrating the property

$$1 - \widehat{m}_1(t) = \lim_{N\to\infty} \frac{M_1(t)}{N}\,. \tag{3.24}$$

The confidence band is almost indistinguishable from the solid line.

3.5. Efficiency. The decisive criterion for efficiency is the necessary computation time for reaching a sufficiently low error. Two basic indicators influencing the computational effort are the number of fictitious jumps and the number of rejections. The number of fictitious jumps is determined by the majorant kernel (cf. (2.14)) indicating its "quality." The number of rejections is determined by the choice of the group bounds (2.31) (cf. (2.36)). We consider the kernel (3.12) and compare the efficiency of the algorithms with two different majorant kernels. The initial number of particles is $N = 10^6$. The dashed lines correspond to the linear majorant kernel (2.25) and the solid lines to the maximum majorant kernel (2.17).

The absolute numbers of jump attempts and the relative numbers of fictitious jumps are shown in Figure 11. The number of jump attempts equals the number of time steps (cf. (2.18), (2.26)). The relative number of fictitious jumps is about 13% in one case and 97% in the other case at time $t = 4$. Note that the number of real jumps does not depend on the choice of the majorant kernel and can be determined from the total concentration via $N\,(1 - m_0(t))$.

In the algorithm with the maximum majorant kernel no rejections occur. The relative number of rejections for the linear majorant kernel are shown in Figure 12. The proportion of rejections for this algorithm is about 35% (the basis $\beta = 2$ was used). The upper bound (2.37) for the relative number of the rejections takes the form $1 - \frac{1}{\beta}$ (cf. (2.28), (2.30)).

A comparison of the CPU time (in seconds) for both algorithms is given in Fig-

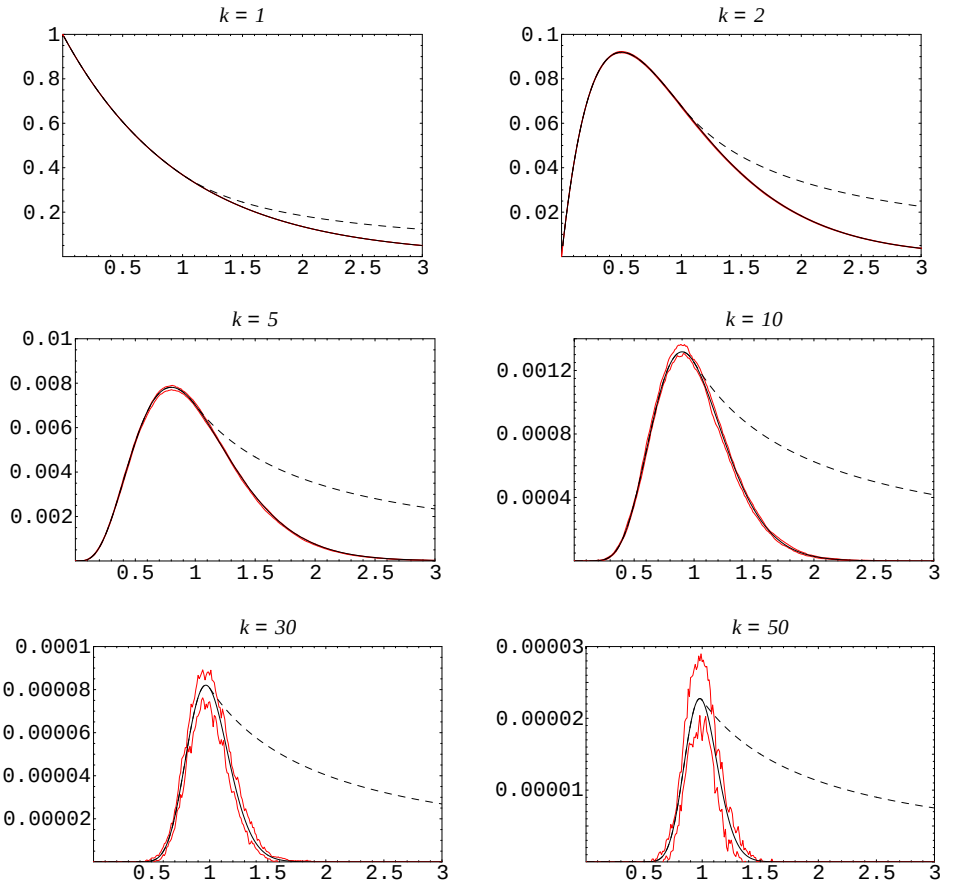


FIG. 9. *Solutions* (3.10) *(dashed) and* (3.22) *(solid) for the multiplicative kernel* (3.9).

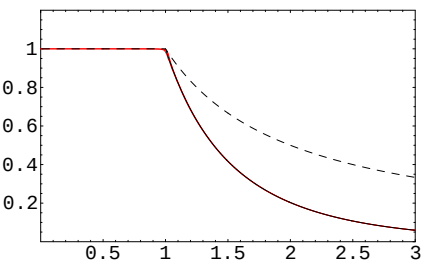


FIG. 10. *Functionals* (3.11) *(dashed) and* (3.23) *(solid) for the multiplicative kernel* (3.9).

ure 13. There is a significant gain factor depending on the length of the time interval. Note that for gelling kernels calculations with the maximum majorant kernel (2.17) become very time consuming when approaching the gelation point.

4. Concluding remarks. A class of stochastic algorithms for the numerical treatment of coagulation processes was introduced. By an appropriate choice of the majorant kernel, a remarkable gain in efficiency has been achieved. This effect is based mainly on a significant reduction of the number of fictitious jumps for systems

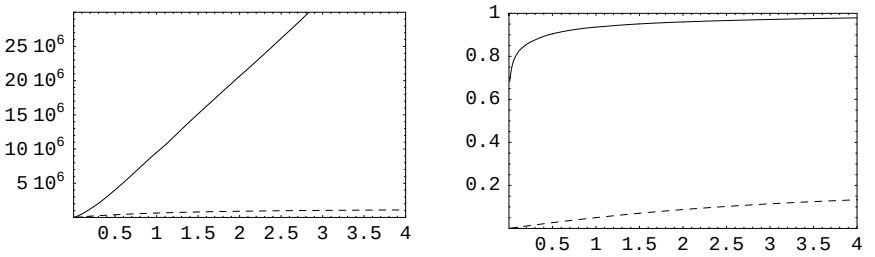


FIG. 11. *Number of time steps (left) and relative number of fictitious jumps (right).*

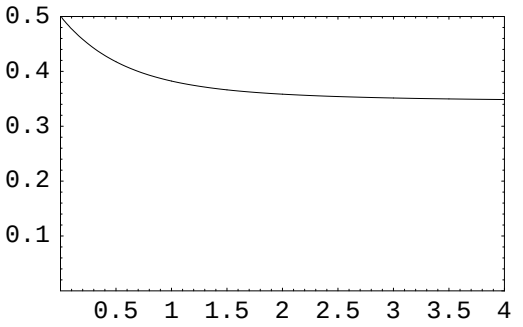


FIG. 12. *Rejections.*

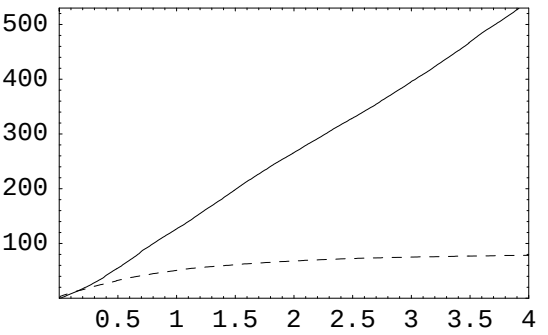


FIG. 13. *CPU time.*

with unbounded kernels and strongly varying cluster sizes.

The algorithm works both in the discrete and the continuous case. Generalizations to equations with fragmentation [7] and source terms are straightforward. Further improvements are possible. In particular, the rejections in the case $i = j$ (cf. (2.21), (2.22)) can be avoided, which is of some relevance in situations where large particles occur. In the discrete case it is more efficient to use the particle number representation (cf. (3.5)) instead of the particle representation (1.4) for particles with lower sizes.

The algorithm in its present form is not suitable for calculations on very long time intervals, since the number of simulation particles is strictly decreasing. There are some interesting ideas in the literature on how to handle this problem. The number

of particles in the system may be doubled, when necessary, by dividing each particle into two equal parts, or, more generally, particles with variable weights may be used (see [21]). An algorithm based on the mass flow instead of the particle number flow was proposed in [3]. Here the number of simulation particles is preserved. These ideas can be implemented in the framework of the new class of algorithms and will be studied in the near future.

The main practical effect of the improved efficiency of our algorithm is that it can be used as a tool for calculations up to and even beyond the gelation point. This seems to be very useful for getting some heuristical insight into the gelation phenomenon for general gelling kernels.

Rigorous convergence results for general coagulation kernels are known up to the gelation point [19]. Convergence to the solution of the Smoluchowski equation after the gelation point is expected [15] for kernels $K(i,j) = o(i)\,o(j)$. However, the case of multiplicative kernel (3.9) shows that the limiting behavior of the stochastic particle system after the gelation point is described by an equation (cf. (3.21)) different from the Smoluchowski equation (1.3). Numerical observations illustrate the known results and confirm the hypothesis of convergence after the gelation point to some deterministic limit.

The numerical studies of the behavior of the largest component in the particle system suggest that several properties, known for the multiplicative kernel (3.9), might be true for larger classes of gelling kernels. There are numerical indications that the quantity $\frac{M_1(t)}{N}$ converges to a deterministic limit. This limit can be used to determine the time of emergence of a cluster of order N as

$$t_g^s = \inf\left\{ t > 0 \,:\, \lim_{N\to\infty} \frac{M_1(t)}{N} > 0 \right\}.$$

It is of interest to find out for which kernels $t_g^s = t_g$ holds. For the multiplicative kernel, this property is illustrated in Figure 4 ($a = 1$). For the kernel (3.14), Figure 7 provides numerical evidence for $t_g = t_g^s = 1$ (cf. [1]). The question for which kernels the limit of the normalized largest component determines (as in (3.24)) the loss of total mass of the corresponding solution is a related open problem.

Acknowledgments. The authors appreciate useful discussions with H. Babovsky, F. Guiaş, and K. Sabelfeld on numerical issues related to the coagulation equation.

REFERENCES

[1] D. Aldous, *Emergence of the giant component in special Marcus-Lushnikov processes*, Random Structures Algorithms, 12 (1998), pp. 179–196.

[2] D. J. Aldous, *Deterministic and stochastic models for coalescence (aggregation and coagulation): A review of the mean-field theory for probabilists*, Bernoulli, 5 (1999), pp. 3–48.

[3] H. Babovsky, *On a Monte Carlo scheme for Smoluchowski's coagulation equation*, Monte Carlo Methods Appl., 5 (1999), pp. 1–18.

[4] E. Buffet and J. V. Pulé, *Polymers and random graphs*, J. Statist. Phys., 64 (1991), pp. 87–110.

[5] Y. R. Domilovskiy, A. A. Lushnikov, and V. N. Piskunov, *Monte Carlo simulation of coagulation processes*, Izv. Acad. Sci. USSR Atmospher. Ocean. Phys., 15 (1979), pp. 129–134.

[6] R. L. Drake, *A general mathematical survey of the coagulation equation*, in Topics in Current Aerosol Research (Part 2), G. Hidy and J. Brock, eds., Pergamon Press, Oxford, 1972, pp. 201–376.

[7] A. Eibeck and W. Wagner, *Approximative Solution of the Coagulation-Fragmentation Equation by Stochastic Particle Systems*, Tech. Rep. 433, Weierstraß–Institut für Angewandte Analysis und Stochastik, Berlin, 1998. Stochastic Anal. Appl., to appear. Also available at http://www.wias-berlin.de/publications/preprints/433.

[8] S. N. Ethier and T. G. Kurtz, *Markov Processes, Characterization and Convergence*, Wiley, New York, 1986.

[9] G. S. Fishman, *Monte Carlo: Concepts, Algorithms, and Applications*, Springer Ser. Oper. Res., Springer-Verlag, New York, 1996.

[10] A. J. Garcia, C. Van den Broeck, M. Aertsens, and R. Serneels, *A Monte Carlo simulation of coagulation*, Phys. A, 143 (1987), pp. 535–546.

[11] D. N. Gillespie, *The stochastic coalescence model for cloud droplet growth*, J. Atmospheric Sci., 29 (1972), pp. 1496–1510.

[12] D. N. Gillespie, *An exact method for numerically simulating the stochastic coalescence process in a cloud*, J. Atmospheric Sci., 32 (1975), pp. 1977–1989.

[13] F. Guiaş, *A Monte Carlo approach to the Smoluchowski equations*, Monte Carlo Methods Appl., 3 (1997), pp. 313–326.

[14] F. Guiaş, *Coagulation-fragmentation processes: Relations between finite particle models and differential equations*, Tech. Rep. 41, Interdisziplinäres Zentrum für wissenschaftliches Rechnen der Universität Heidelberg (SFB 359), Heidelberg, 1998.

[15] I. Jeon, *Existence of gelling solutions for coagulation-fragmentation equations*, Comm. Math. Phys., 194 (1998), pp. 541–567.

[16] N. J. Kokholm, *On Smoluchowski's coagulation equation*, J. Phys. A, 21 (1988), pp. 839–842.

[17] A. A. Lushnikov, *Some new aspects of coagulation theory*, Izv. Akad. Nauk SSSR Ser. Fiz. Atmosfer. i Okeana, 14 (1978), pp. 738–743.

[18] A. H. Marcus, *Stochastic coalescence*, Technometrics, 10 (1968), pp. 133–148.

[19] J. R. Norris, *Smoluchowski's coagulation equation: Uniqueness, nonuniqueness and a hydrodynamic limit for the stochastic coalescent*, Ann. Appl. Probab., 9 (1999), pp. 78–109.

[20] K. K. Sabelfeld, *Stochastic models for coagulation of aerosol particles in intermittent turbulent flows*, Math. Comput. Simulation, 47 (1998), pp. 85–101.

[21] K. K. Sabelfeld, S. V. Rogazinskii, A. A. Kolodko, and A. I. Levykin, *Stochastic algorithms for solving Smoluchowski coagulation equation and applications to aerosol growth simulation*, Monte Carlo Methods Appl., 2 (1996), pp. 41–87.

[22] P. G. J. van Dongen and M. H. Ernst, *Fluctuations in coagulating systems*, J. Statist. Phys., 49 (1987), pp. 879–926.

[23] M. von Smoluchowski, *Drei Vorträge über Diffusion, Brownsche Molekularbewegung und Koagulation von Kolloidteilchen*, Phys. Z., 17 (1916), pp. 557–571, 585–599.

SIAM J. SCI. COMPUT.
Vol. 22, No. 3, pp. 822–834
© 2000 Society for Industrial and Applied Mathematics

AN IMPROVED LAGUERRE EIGENSOLVER FOR UNSYMMETRIC MATRICES*

JORGEN L. NIKOLAJSEN†

Abstract. A Laguerre iteration procedure is described for finding the eigenvalues of unsymmetric matrices with improved efficiency. Compared to the QR method, the processing time for dense matrices is reduced by roughly a factor of 1.6 and for sparse matrices by a factor of up to 2.8 without sacrificing accuracy. This is achieved primarily by means of a new procedure for reducing the original matrix to sparse Hessenberg form. Alternatively, the Laguerre procedure will typically provide one additional significant digit, compared to the QR method, when allowed to run for as long as the QR method.

Key words. unsymmetric matrix, eigenvalues, Laguerre's method, matrix reduction

AMS subject classification. 15A18

PII. S106482759834963X

1. Introduction.

Many science and engineering problems lead to the standard eigenvalue problem, [A] $\{x\} = z\{x\}$, where [A] is an unsymmetric matrix. The eigenvalues z are typically calculated by means of the QR method of Francis [1]. This remains one of the most efficient and robust methods available despite efforts by many researchers to find more efficient alternatives.

The current effort to improve the efficiency of eigencalculations focuses mainly on reducing the original dense, unsymmetric matrix [A] beyond Hessenberg form to a sparser form, which permits more efficient eigenvalue extraction without loss of accuracy. Dax and Kaniel [2] largely initiated this direction of work with an improvement in the stability of the reduction to tridiagonal form, which is amenable to very rapid eigencalculation, for example, by LR iteration. This work was resumed by Geist [3], who further improved the stability of tridiagonalization and was able to calculate the eigenvalues by LR iteration more that three times faster than with the QR method. However, Geist observed an unacceptable deterioration in accuracy with matrix size. In a subsequent paper, Dongarra, Geist, and Romine [4] used this procedure to obtain approximate eigenpairs which were subsequently refined iteratively to surpass the QR method in accuracy. However, only 20% of the eigenpairs could be calculated during the time it took for the QR method to calculate them all. Howell [5] pointed out that the stability of Geist's tridiagonalization procedure could be further improved by omitting the least stable similarity transformations. This led to a band matrix with increasing bandwidth whose eigenvalues could still be calculated by LR iteration but with reduced efficiency. Further testing appears to be needed to determine the efficiency of this approach.

This paper seeks to alleviate the instability problems associated with full matrix tridiagonalization by terminating the elimination process when no further stable similarity transformations can be found. This can be arranged to produce a sparse Hessenberg matrix whose eigenvalues can be calculated by Laguerre iteration. The procedure was found to be typically 1.6 times faster than the QR method without

*Received by the editors December 22, 1998; accepted for publication (in revised form) February 11, 2000; published electronically August 31, 2000.
http://www.siam.org/journals/sisc/22-3/34963.html
†School of Engineering and Advanced Technology, Staffordshire University, Stafford ST18 0DF, UK (j.l.nikolajsen@staffs.ac.uk).

$$\begin{bmatrix} a_{11} & a_{12} & 0 & 0 & 0 & 0 \\ a_{21} & a_{22} & a_{23} & a_{24} & a_{25} & a_{26} \\ 0 & a_{32} & a_{33} & a_{34} & a_{35} & a_{36} \\ 0 & 0 & a_{43} & a_{44} & a_{45} & a_{46} \\ 0 & 0 & a_{53} & a_{54} & a_{55} & a_{56} \\ 0 & 0 & a_{63} & a_{64} & a_{65} & a_{66} \end{bmatrix} \qquad \begin{bmatrix} h_{11} & h_{12} & 0 & 0 & 0 & 0 \\ h_{21} & h_{22} & h_{23} & h_{24} & h_{25} & h_{26} \\ 0 & h_{32} & h_{33} & h_{34} & h_{35} & h_{36} \\ 0 & 0 & h_{43} & h_{44} & h_{45} & h_{46} \\ 0 & 0 & 0 & h_{54} & h_{55} & h_{56} \\ 0 & 0 & 0 & 0 & h_{65} & h_{66} \end{bmatrix}$$

(a) Stage 1 reduction (b) Stage 2 reduction

FIG. 1. *Matrix reduction to sparse Hessenberg form.*

sacrificing accuracy. Alternatively, one additional significant digit can typically be achieved, compared to the QR method, within the QR processing time. The procedure is outlined below.

2. Matrix reduction to sparse Hessenberg form. The eigensolver described here uses the familiar steps of matrix balancing, matrix reduction, and eigeniteration. The standard balancing procedure of Parlett and Reinsch [6] is used. The subsequent matrix reduction procedure transforms a dense unsymmetric matrix to upper Hessenberg form with a sparse upper triangle. The efficiency of the subsequent Laguerre iteration increases with the sparseness of the upper triangle but is unaffected by the location of the nonzero elements. The matrix reduction procedure is therefore designed to minimize the number of nonzero elements without any concern for their location. The reduction is performed in two stages, both by elementary similarity transformations.

2.1. Stage 1 reduction. The standard procedure for reduction to Hessenberg form (see Martin and Wilkinson [7]) is extended to include alternate row eliminations to the right of the superdiagonal, as first suggested by Geist [3]. The row eliminations continue for as long as sufficient stability can be ensured, as defined by the parameter c introduced below. Figure 1(a) shows a matrix with column 1, row 1, and column 2 reduced in that order. The next step, reduction of row 2, will be described in detail.

All the *column* reductions (toward upper Hessenberg form) use standard partial pivoting with all multipliers less than unity to ensure stability. This is also the case for the row reductions except that the superdiagonal element (a_{23} in row 2) cannot participate unless it is already the pivot (largest) element. This is because the associated row and column interchanges would destroy zeros in the lower triangle. However, standard partial pivoting can be used to eliminate all the row elements to the right of a_{24}, after which a choice can be made as to whether it is acceptable to eliminate a_{24} with a_{23} as a pivot.

Thus, the reduction of row 2 in Figure 1(a) proceeds as follows:

1. Identify the largest (pivot) element a_p from among $|a_{23}|$, $|a_{24}|$, $|a_{25}|$, and $|a_{26}|$.
2. If $a_p = |a_{23}|$, then a_{24}, a_{25}, and a_{26} are eliminated by stable similarity transformations and the process moves on to column 3, then to row 3, etc.
3. If $a_p \neq |a_{23}|$, then a_p is moved to position a_{24} by appropriate row and column interchanges and a_{25} and a_{26} are eliminated with a_p as pivot. These eliminations are stable and do not destroy previous zeros.

4. If $a_p \leq |a_{23}| \cdot 2^c$ (where c is a user supplied parameter), then $a_p = (\text{new } a_{24})$ is eliminated using a_{23} as the pivot and the process moves on to column 3. Thus, the user decides, through the choice of c, whether $|a_{23}|$ is large enough to eliminate a_p with the required accuracy. $c = 0$ provides the stability of standard partial pivoting but $c \geq 1$ usually provides satisfactory accuracy for all eigenvalues within the current overall strategy, as demonstrated in section 6.
5. If $a_p > |a_{23}| \cdot 2^c$, then the elimination of $a_p = (\text{new } a_{24})$ is deemed too unstable and is omitted. This single failure prevents any further zeroing of elements in the upper triangle during Stage 1 reduction, since previous zeros would be destroyed. In this case, standard Hessenberg reduction takes over until the Hessenberg matrix of Figure 1(b) has emerged.

2.2. Stage 2 reduction. In Figure 1(b), Stage 1 reduction, including alternate row reductions, has been completed with one successful row elimination. (As seen later, a much larger number of rows can normally be eliminated during Stage 1.)

The row eliminations are now resumed, without pivoting which would destroy the zero lower triangle as follows:

1. Find the largest (pivot) element p_1 from among $|h_{23}|$, $|h_{24}|$, $|h_{25}|$, and $|h_{26}|$ and use it to eliminate all the elements to the right of p_1. These eliminations are stable.
2. Find the largest element p_2 to the left of p_1 (and to the right of h_{22}) and use it to eliminate all the elements between p_1 and p_2. These eliminations are stable.
3. If $p_1 \leq p_2 \cdot 2^c$ (where c is a user-supplied parameter), then p_1 is eliminated with p_2 as the pivot. As in Stage 1 reduction, $c = 0$ provides the stability of standard partial pivoting but satisfactory accuracy of all eigenvalues is usually obtained in practice with $c \geq 1$.
4. If $p_1 > p_2 \cdot 2^c$, then the elimination of p_1 is deemed too unstable and is omitted. When this happens, no element from p_1's column can ever again be used as a pivot since this would destroy previous zeros.
5. Go back to step 2 until elimination of all the elements to the right of h_{23} has been attempted.
6. Proceed with the following row until elimination of all elements above the superdiagonal has been attempted.

3. Laguerre iteration. Laguerre's method is well known as one of the most efficient procedures for finding the roots of polynomials. Convergence is virtually global and the rate of convergence is cubic in the vicinity of simple roots. Parlett [8] appears to have been the first to apply Laguerre iteration to unsymmetric eigenproblems. Parlett provides a comprehensive discussion of Laguerre iteration applied to eigensolution with dense Hessenberg matrices. Thus, only a summary will be given here. Parlett's procedure was applied by Ward [9] who found it comparable to the QR method in efficiency on parallel computers. However, tests by Foster [10] indicated that Laguerre's method was only half as efficient as the QR method.

Finding the eigenvalues z of a Hessenberg matrix [H] is equivalent to finding the roots of its characteristic polynomial $p(z) = \det([\mathrm{H}] - z[\mathrm{I}])$. Laguerre iteration toward a root z proceeds as follows:

$$z_{i+1} = z_i - \Delta z_i, \tag{3.1}$$

where

$$\Delta z_i = n \Big/ \left[f_i \pm \sqrt{(n-1)(nh_i - f_i^2)} \right], f_i = p_i'/p_i, \text{ and } h_i = f_i^2 - p_i''/p_i. \tag{3.2}$$

p_i' and p_i'' are the polynomial derivatives and n is the order of [H].

Roots z_j, already found, are suppressed by setting

$$p = p \Big/ \sum_j (z - z_j), \tag{3.3}$$

which turns out to be equivalent to setting

$$f = f - \sum_j 1/(z - z_j) \text{ and } h = h - \sum_j 1/(z - z_j)^2. \tag{3.4}$$

Iteration toward complex eigenvalues requires slight modifications (see Parlett [8]), but (3.1)–(3.4) are sufficient for later illustration of the implementation.

4. Hyman's method. The determinant value p and its derivatives p' and p'' in (3.2) are calculated by Hyman's method; see Wilkinson [11]. For maximum efficiency, an additional similarity transformation is performed on the Hessenberg matrix to obtain unit subdiagonal elements as shown in (4.1); see [11]. As outlined earlier, the upper triangle of [H] will usually be sparse.

$$[\mathrm{H}] = \begin{bmatrix} h_{11} & h_{12} & h_{13} & h_{14} & h_{15} & h_{16} \\ 1 & h_{22} & h_{23} & h_{24} & h_{25} & h_{26} \\ 0 & 1 & h_{33} & h_{34} & h_{35} & h_{36} \\ 0 & 0 & 1 & h_{44} & h_{45} & h_{46} \\ 0 & 0 & 0 & 1 & h_{55} & h_{56} \\ 0 & 0 & 0 & 0 & 1 & h_{66} \end{bmatrix}. \tag{4.1}$$

Hyman's method leads to the following recursive formulas for p, p', and p'' for use in (3.2):

$$\begin{cases} p_5 = (z - h_{66}), \\ p_4 = (z - h_{55})p_5 - h_{56}, \\ p_3 = (z - h_{44})p_4 - h_{45}p_5 - h_{46}, \\ p_2 = (z - h_{33})p_3 - h_{34}p_4 - h_{35}p_5 - h_{36}, \\ p_1 = (z - h_{22})p_2 - h_{23}p_3 - h_{24}p_4 - h_{25}p_5 - h_{26}, \\ -p = (z - h_{11})p_1 - h_{12}p_2 - h_{13}p_3 - h_{14}p_4 - h_{15}p_5 - h_{16}, \end{cases} \tag{4.2}$$

$$\begin{cases} p_4' = (z - h_{55}) + p_5, \\ p_3' = (z - h_{44})p_4' - h_{45} + p_4, \\ p_2' = (z - h_{33})p_3' - h_{34}p_4' - h_{35} + p_3, \\ p_1' = (z - h_{22})p_2' - h_{23}p_3' - h_{24}p_4' - h_{25} + p_2, \\ -p' = (z - h_{11})p_1' - h_{12}p_2' - h_{13}p_3' - h_{14}p_4' - h_{15} + p_1, \end{cases} \tag{4.3}$$

$$
(4.4)\qquad \begin{cases} p_3'' = 2(z - h_{44}) + 2p_4', \\ p_2'' = (z - h_{33})p_3'' - 2h_{34} + 2p_3', \\ p_1'' = (z - h_{22})p_2'' - h_{23}p_3'' - 2h_{24} + 2p_2', \\ -p'' = (z - h_{11})p_1'' - h_{12}p_2'' - h_{13}p_3'' - 2h_{14} + 2p_1'. \end{cases}
$$

Equations (4.2) through (4.4) are easily generalized. p, p', and p'' are usually complex. They must be calculated at each Laguerre iteration step for each eigenvalue. Apart from the matrix reduction, this is the most time-consuming part of the eigencalculations. Zeroing of a large number of elements h in the upper triangle of [H], as discussed earlier, becomes essential to reduce the work load below that of the QR method. The numerical implementation must check for and skip all multiplications and additions involving zero elements.

5. Numerical implementation. An excellent outline of the numerical implementation of Laguerre iteration for nonsymmetric eigenproblems can be found in Parlett [8]. The current implementation is based on Parlett's work and focuses on improvements that maximize efficiency. The main improvement is the matrix reduction to sparse Hessenberg form, as outlined earlier. This section highlights additional improvements, mainly in the choice of start values and stop criteria, to minimize the number of iterations.

The current implementation rejects eigenproblems which are so ill-conditioned that one or more eigenvalues can only be determined to one significant digit. This simplifies the procedure considerably and is justified by arguing that eigenvalues with one significant digit are probably inadequate in most applications. The specific parameter values recommended in the following are for double-precision implementation (15–16 significant digits). These parameter values are based on extensive experimentation and test experience.

5.1. First start value. The choice of the start value z_0 for the first eigenvalue is problematic because of the assumed total lack of knowledge of the locations of all eigenvalues. Parlett [8] recommended using "the Laguerre iterate of infinity" which is larger than the modulus of the largest eigenvalue. This rarely fails to work. The drawback is that a relatively large number of iterations may be required to reach the nearest eigenvalue.

In the current implementation, a first start value z_0 near zero is used instead. This is, on average, much closer to an eigenvalue and therefore provides faster convergence. In addition, the search for an acceptable z_0 near zero can be arranged to automatically identify all roots near zero, which greatly simplifies the stop criteria outlined later. Thus z_0 is found as follows:

1. Set $z_0 = 0$.
2. Do one complex Laguerre iteration, in accordance with (3.1) and (3.2), and check on the size of $|\Delta z_i|$.
3. If $|\Delta z_i| > 8^*$matrix-norm, including processor-defined "infinity" and "NAN" (not a number), and p has underflow, then z_0 is so close to a root that the Laguerre iteration has broken down. Set the flag to indicate near-zero root, set $z_0 = z_0 + 2^{-7}$ to get away from the root, and go back to 2.
4. If $|\Delta z_i| > 8^*$matrix-norm, including "infinity" and "NAN," but p does not underflow, then z_0 is a local extremum point causing the iterate to shoot toward infinity. Remove the extremum temporarily by declaring a dummy

root at z_0 (to be removed when the next true root has been found). Set $z_0 = z_0 + 2^{-7}$ to get away from the dummy root, and go back to 2.

5. If $|\Delta z_i| < 2^{-7}$, then z_0 is probably close to a root. Set the flag to indicate near-zero root, set $z_0 = z_0 + 2^{-7}$ to get away from the root, and go back to 2.
6. If $2^{-7} < |\Delta z_i| < 8^*$matrix-norm and the flag indicates near-zero root (set during the previous iteration), then shift the eigenproblem by z_0 (i.e., subtract z_0 from all the diagonal elements), enter the mainstream iteration (as defined by (3.1)–(3.4)), and find the (shifted) root using $-z_0$ as the start value. Shift the eigenproblem (including the root) back again, record and suppress the root (see (3.4)), clear the flag for near-zero root, set $z_0 = z_0 + 2^{-7}$, and go back to 2 to look for additional near-zero roots. (Shifting the root away from zero allows it to be determined using the stop criteria outlined below.)
7. If $2^{-7} < |\Delta z_i| < 8^*$matrix-norm and near-zero root is not flagged, then no undiscovered near-zero roots exist. Proceed with mainstream iteration, as defined by (3.1)–(3.4).

5.2. Subsequent start values. Parlett [8] suggested that the penultimate Laguerre iterate z_{qi} toward the root z_q could be used to provide a "free" Newton iteration step toward the next root z_τ as follows:

$$z_{\tau 1} = z_q - 1/s, \tag{5.1}$$

where

$$s = \frac{p''(z_{qi})}{2p'(z_{qi})} - \sum_j \frac{1}{z_{qi} - z_j}, \tag{5.2}$$

where z_j are all the roots found previously, excluding z_q.

Parlett's start value $z_{\tau 1}$ is adopted here, except that, when z_q is complex, its complex conjugate, z_q^*, is included among the z_j's in (5.2). This removes the attraction of z_q^*, resulting in a better approximation of $z_{\tau 1}$ to the next root z_τ.

The use of $z_{\tau 1}$ speeds up convergence considerably, but it can fail because Newton iteration lacks global convergence and because there is no guaranteeing that it will bring $z_{\tau 1}$ outside the area of indeterminacy of z_q, in which case the iteration may converge toward z_q again. A safe alternative start value $z_{\tau 0}$ is therefore needed. It is chosen as the last iterate toward z_q which has less than seven significant binary digits (∼2 decimal digits). This is the smallest number of digits accepted for any root without aborting the eigencalculations. It thus insures that z_0 is outside the area of indeterminacy of all roots. A check is made to determine whether (1) $z_{\tau 1}$ is within the area of indeterminacy of z_q, (2) $|z_{\tau 1}| > 8 \cdot$matrix-norm, or (3) $|z_{\tau 1}| > 8 \cdot z_q$. In either case, $z_{\tau 1}$ is rejected as the start value and $z_{\tau 0}$ is used instead.

5.3. Stop criteria. The following stop criteria were devised to ensure that the iteration stops immediately when all available significant digits have been determined:

1. If $|z_i/\Delta z_i| > 2^{52}$ (see (3.1)), then z_{i+1} has at least 52 significant binary digits (∼15 decimal digits) and is declared a root. This stop criterion normally handles convergence after one or two iterations.
2. If $|z_{i+1}/\Delta z_{i+1}| \le |z_i/\Delta z_i|$ (i.e., number of significant digits of $z_{i+1} \le$ number of significant digits of z_i), then z_{i+1} has reached the area of indeterminacy, thus, no further significant digits can be extracted and a root is declared. This is because once convergence has started, the convergence rate does not slow down until the area of indeterminacy has been reached. This check primarily finds ill-conditioned roots that have relatively few significant digits.

3. If $(z_{i+1}/\Delta z_{i+1})^2/|z_i/\Delta z_i| > 2^{52}$, then z_{i+2} will be a root. Example: If $|z_i/\Delta z_i| \cong 2^7$ and $|z_{i+1}/\Delta z_{i+1}| \cong 2^{20}$, then the current convergence rate is roughly $20/7 \cong 2.9$ (i.e., almost cubic). The next iterate, z_{i+2}, will therefore either achieve at least $(20/7) \cdot 20 \cong 57$ significant binary digits (more than double-precision accuracy) or it will reach the area of indeterminacy. Thus z_{i+2} will be a root. This check primarily finds well-conditioned roots, which have many significant digits, at both fast and slow convergence rates.
4. Finally, if a root has not been found within 32 iterations, then the current iterate is accepted as a root and a warning code is set.

Convergence must have started before stop criterion step 2 can be applied safely. Convergence is deemed to have started if both of the following criteria are satisfied:

1. $z_i/\Delta z_i \geq 2^3$, $z_{i+1}/\Delta z_{i+1} \geq 2^4$, and $z_{i+2}/\Delta z_{i+2} \geq 2^5$. In other words, the number of significant binary digits of three consecutive iterates must be at least 3, 4, and 5.
2. $(z_{i+1}/\Delta z_{i+1})/(z_i/\Delta z_i) \geq 2^1$ and $(z_{i+2}/\Delta z_{i+2})/(z_{i+1}/\Delta z_{i+1}) \geq 2^1$. This is equivalent to saying that at least one additional significant binary digit must have been found in each of the two last consecutive iterations.

The stop criteria have been condensed into the current form by extensive experimentation and testing. Their simplicity and robustness have been achieved partly by making them applicable to nonzero roots only. Near-zero roots are extracted and suppressed beforehand, as outlined earlier.

When a root $z = x + iy$ has been found, its complex conjugate z^* is declared and suppressed simultaneously provided z is definitely complex. Complexity of z is considered confirmed if $|y/x| > 2^{-7}$ in which case y is outside the area of indeterminacy of x for the most ill-conditioned root accepted without aborting the eigencalculations. If $|y/x| < 2^{-7}$, then only z is declared a root. z^* will then emerge separately if z is truly complex.

Double or multiple roots slow the convergence rate from cubic to near linear. When a near-linear convergence rate persists, the iteration increment Δz_i in (3.2) is replaced by

$$\Delta z_i = n \Big/ \Big[f_i \pm \sqrt{(n-k)/k \cdot (nh_i - f_i^2)} \Big], \tag{5.3}$$

which provides cubic convergence for a k-fold root. The first such iteration is "free" if p, p', and p'' from the previous iteration are used. The overhead involved in switching to (5.3) was found on balance to be justified only for $k = 2$ (i.e., double-roots). If $k = 2$ does not improve convergence, then $k = 1$ is reinstated for the rest of the iteration toward the current root.

Both a real and a complex iteration path were implemented to prevent the inefficient use of complex arithmetic for finding real roots. Switching between the two paths is automatic and biased toward a switch from complex to real arithmetic whenever warranted.

5.4. Miscellaneous improvements. The following additional measures were taken to further improve the efficiency and accuracy of the eigencalculations:

1. The matrix was scaled to prevent overflow and underflow. The similarity transformations leading to unit subdiagonal elements (see section 4) are particularly prone to overflow.
2. Matrix splitting is employed when negligible sub- or superdiagonal elements are encountered. This leads to two or more smaller eigenproblems, which are solved separately.

6. Test results. The Laguerre procedure was tested extensively for speed, accuracy, and robustness on a wide variety of matrices. The results were compared with those from a commercially available QR procedure based on EISPACK. The tests were carried out on a slow (33MHz) 80486 based PC to provide good resolution in the processing time measurements. Both the Laguerre and the QR procedure were implemented entirely in assembly language and in double precision. Both employ the familiar sequence of matrix balancing, matrix reduction, and eigenvalue iteration.

Equation (6.1) shows one of the matrix types used for testing:

$$[\mathrm{A}] = \left[\begin{array}{cc:cc:cc:c} 1 & -1 & 0 & 0 & 0 & 0 & . \\ 1 & 1 & 0 & 0 & 0 & 0 & . \\ \hdashline 2 & 2 & 2 & -2 & 0 & 0 & . \\ 2 & 2 & 2 & 2 & 0 & 0 & . \\ \hdashline 3 & 3 & 3 & 3 & 3 & -3 & . \\ 3 & 3 & 3 & 3 & 3 & 3 & . \\ \hdashline . & . & . & . & . & . & . \end{array}\right]. \tag{6.1}$$

These matrices have exact eigenvalues of $\lambda_e = 1 \pm i,\ 2 \pm 2i,\ 3 \pm 3i$, etc., allowing exact error calculations. Neither the Laguerre nor the QR method recognized the lower Hessenberg form of [A]. Both proceeded with balancing and reduction to upper Hessenberg form followed by eigeniteration.

Figure 2 shows a 30×30 matrix, of the type indicated by (6.1), after reduction to sparse Hessenberg form for various values of parameter c. The 1's in Figure 2 represent all nonzero elements. A negative c invokes standard Hessenberg reduction with row elimination disabled. A nonnegative c results in multipliers less than 2^c being accepted during elimination of row-pivots; see section 2.2. Thus, with $c = 0$, standard partial pivoting is applied to all row reductions, i.e., all multipliers are less than unity. With $c = 1$, the procedure accepts multiplier values of up to 2 for row-pivot elimination, etc. c does not affect the column eliminations which are always carried out with multipliers less than unity. Notice that a significant number of upper triangle elements can be zeroed with standard partial pivoting ($c = 0$). For random matrices with elements between -1 and 1, $c = 0$ results in zeroing of approximately half the upper triangle. Surprisingly, the nonzero upper triangle elements tend to congregate near the diagonal rather than being randomly distributed across the triangle. The explanation appears to be that matrix balancing, and to some extent Hessenberg reduction, tend to produce larger near-diagonal elements for matrices of the type shown in (6.1). This promotes elimination of row elements far from the diagonal. Preliminary investigations have confirmed that rebalancing after Stage 1 reduction (see section 2.1) increases the size of the near-diagonal elements significantly, allowing a large number of additional upper triangle elements to be zeroed with standard partial pivoting ($c = 0$). The efficiency of rebalancing is currently being investigated. It has not yet been implemented.

Notice in Figure 2(d) that $c = 4$ leads to full tridiagonalization. Generally, the larger the matrix, the larger the c-value required for tridiagonalization.

All the matrix types tried, including, for example, random matrices and the test matrices from Gregory and Karney [12], show very similar levels of sparsity for given values of c. They also show the same trend of bunching of elements near the diagonal.

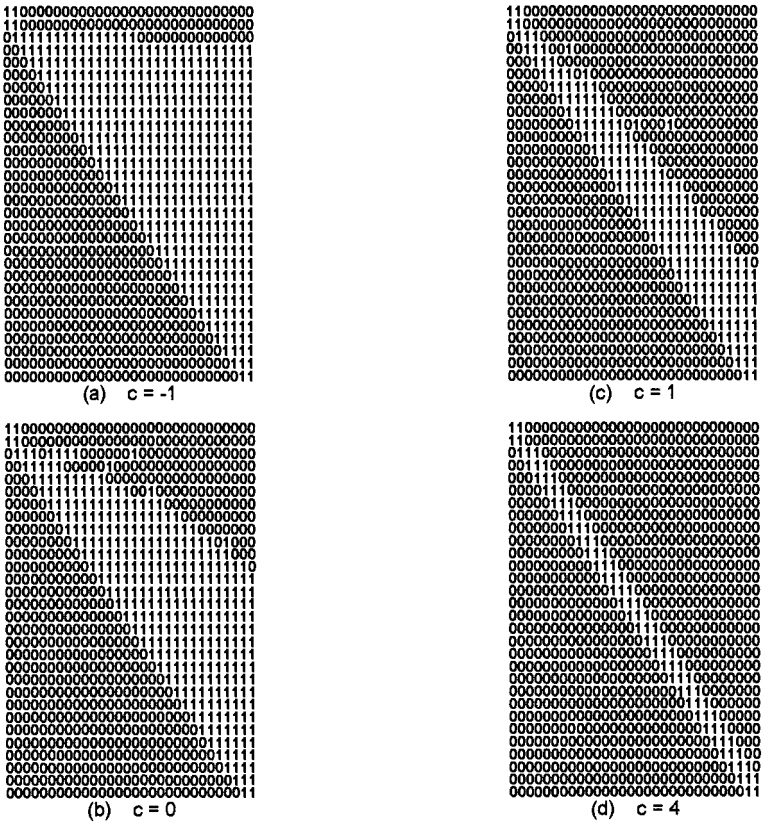


FIG. 2. *Effect of multiplier size on sparseness of Hessenberg form.*

Figure 3 shows the maximum relative error, $|(\lambda_e - \lambda_a)/\lambda_e|_{\max}$ of the calculated eigenvalues for matrices of the type shown in (6.1). λ_e and λ_a are the exact and the calculated eigenvalues, respectively. Comparison with the QR method shows that the Laguerre procedure usually provides approximately one additional significant digit when row elimination is disabled ($c = -1$). When the matrices are almost fully tridiagonalized ($c = 4$), the Laguerre method still matches the QR method in terms of accuracy in most cases. However, the error on the Laguerre procedure is becoming erratic and drops well below the QR accuracy for matrices of order 60 and 70. Care should be taken to ensure that the error is acceptable for the specific problem at hand. This can be done with a preliminary run with maximum accuracy ($c = -1$) before "production runs" with larger c's which reduce processing time as demonstrated later.

The overall reduction in accuracy with increasing c seems surprisingly small. Two explanations are offered.

(1) As outlined in section 2.2, irrespective of the value of c, standard partial pivoting is used to eliminate all but the relatively few row-pivots themselves, thus minimizing the number of unstable eliminations.

(2) The QR iteration procedure, by its nature, repeatedly modifies and stores the matrix elements in their double precision storage locations (15–16 significant digits). This is accompanied by a small but steady accumulation of round-off errors. The Laguerre iteration does not modify the elements of the reduced matrix and therefore

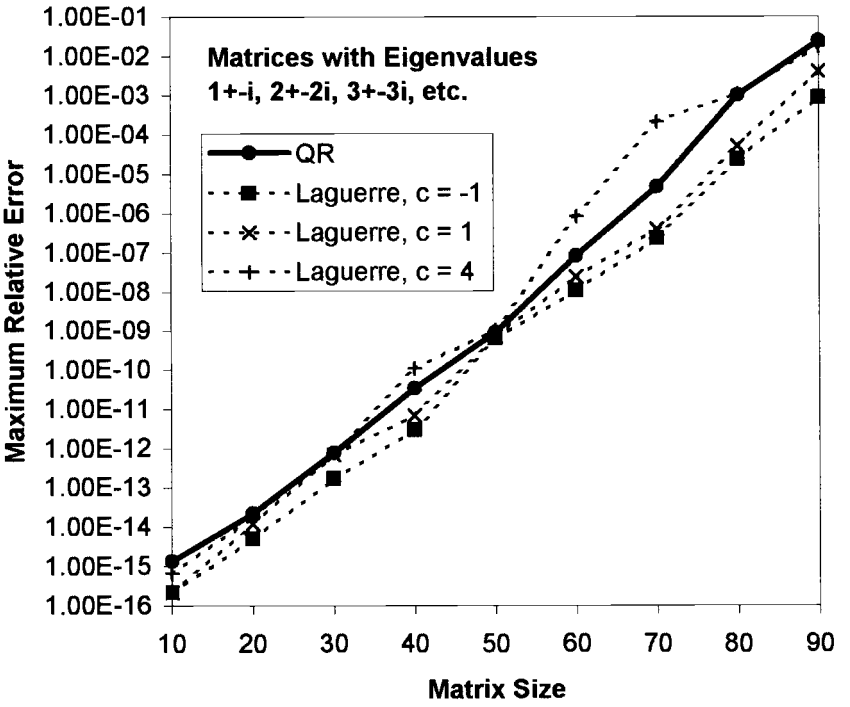


FIG. 3. *Comparison of maximum errors for dense matrices.*

does not incur these round-off errors. Instead, the Laguerre procedure is able to accumulate inner products throughout the iteration phase, in this case with the standard 19 significant digits of a PC processor.

Notice that the graphs in Figure 3 are not smooth. This is in line with expectations since the properties of matrices, of the type shown in (6.1), do not necessarily change smoothly with matrix size.

The corresponding speed comparison is shown in Figure 4. Roughly 1/3 of the upper triangle elements are zero when $c = 0$; see, for example, Figure 2. However, the Laguerre procedure does not become faster than the QR procedure until $c = 1$, when roughly 2/3 of the upper triangle is zero. This is most likely due to the additional time needed for the reduction to sparse Hessenberg form, before Laguerre iteration, compared to the shorter time needed for standard Hessenberg reduction, prior to QR iteration. The result is that a relatively large number of upper triangle zeros are required before the speed advantage of the Laguerre iteration can be realized. The maximum speed advantage is about a factor of 1.6 for $c = 4$, and this is relatively independent of matrix size ($c = 4$ tridiagonalizes the matrices up to order 30×30.)

The speed advantage for very sparse matrices is illustrated in Figure 5 for Fibonacci matrices of the type

$$A = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \tag{6.2}$$

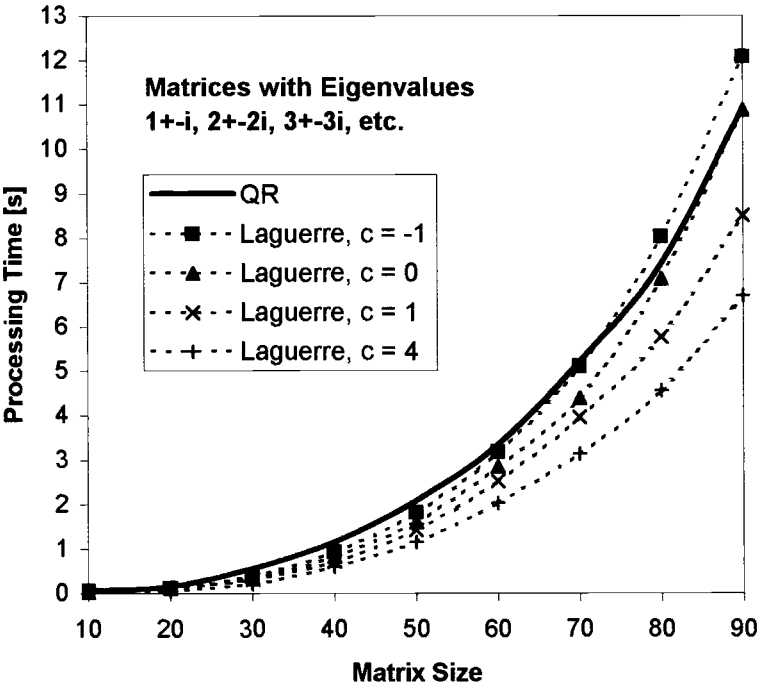


FIG. 4. *Comparison of processing time for dense matrices.*

Such matrices have all their eigenvalues located on the unit circle in the complex plane. Figure 5 shows a speed advantage of about 2.8 for the Laguerre procedure. The main reasons for the large speed advantage are probably the following:

1. Matrix reduction is unnecessary, leaving the QR method without the advantage of the quicker reduction to standard Hessenberg form.
2. The QR iterations destroy the zeros of the upper triangle, i.e., they do not benefit from the sparseness of [A].

All the eigenvalues from these test runs had full double precision accuracy (15–16 significant digits).

Some of the small ill-conditioned test matrices from Gregory and Karney [12] were also tried. These matrices were too small to provide a speed comparison. In terms of accuracy, no firm trend could be established. In some cases the QR method was slightly more accurate, and in other cases the Laguerre method with $c = -1$ was slightly more accurate. But on average, the QR method had a slight edge. One likely contributing factor is that the matrices are too small for the QR iteration to accumulate any appreciable round-off errors. The QR and Laguerre performances being so close, no attempt was made to improve the Laguerre performance for these small matrices.

7. Summary. In summary, the Laguerre procedure has been found to provide considerable versatility for large matrices, allowing the user to either increase the processing speed or improve the accuracy of the calculated eigenvalues, as compared to the QR method. This is useful in design work where repeated eigencalculations are often required to gauge the effect of small design changes or when Campbell diagrams

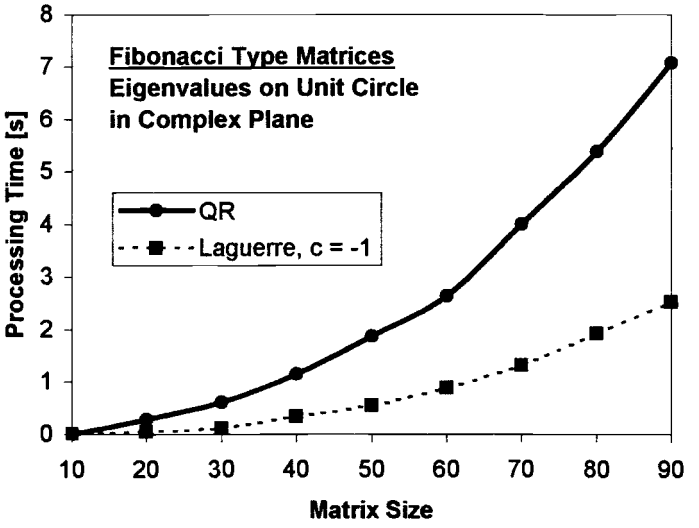


FIG. 5. *Comparison of processing time for sparse matrices.*

must be generated to trace the eigenvalues of matrices which are functions of a parameter. The Laguerre procedure can save time in such work by allowing the user to gain processing speed in exchange for significant digits, which may not be needed. Alternatively, when all possible digits must be extracted from an ill-conditioned matrix, the Laguerre procedure will often provide one additional significant digit compared to the QR method. A large number of test runs with a wide variety of matrices have confirmed these general conclusions.

There is scope for further improvement of the efficiency of the Laguerre eigensolver as follows:

- Replace the Newton start step with a Laguerre start step as suggested by Foster [10] to further reduce the number of iterations.
- Implement an option for user-supplied start values to reduce the number of iterations in design calculations and generation of Campbell diagrams for matrices, which are functions of a parameter.
- Calculate only the required eigenvalues, which often are those with a modulus below a user-supplied value. Such a procedure is currently being implemented based on the work of Delves and Lyness [13].
- Rebalance the matrix after Stage 1 Hessenberg reduction. This is currently being investigated as a means of increasing the size of the near-diagonal elements. Preliminary results show that this leads to stable elimination of many additional elements in the upper triangle.
- A reviewer of this paper has kindly suggested that advantage be taken of the envelope structure of the matrices; see Figure 2. The address of the rightmost nonzero element of each row could be saved and used to truncate the Hyman equations, (4.2)–(4.4).

8. Conclusion. A Laguerre iteration procedure has been implemented for eigensolution with unsymmetric matrices. Extensive testing indicates that the procedure

is superior to the QR method for large matrices, providing a speed advantage of between 1.6 and 2.8, often without sacrificing accuracy. When accuracy is paramount, a parameter can be set to extract typically one additional significant digit, compared to the QR method, at roughly the same processing speed as the QR method.

REFERENCES

[1] J.G. Francis, *The QR transformation—Part* 2, Comput. J., 4 (1961), pp. 332–345.

[2] A. Dax and S. Kaniel, *The ELR method for computing the eigenvalues of a general matrix*, SIAM J. Numer. Anal., 18 (1981), pp. 597–605.

[3] G.A. Geist, *Reduction of a general matrix to tridiagonal form*, SIAM J. Matrix Anal. Appl., 12 (1991), pp. 362–373.

[4] J.J. Dongarra, G.A. Geist, and C.H. Romine, *Algorithm* 710. *Fortran subroutines for computing the eigenvalues and eigenvectors of a general matrix by reduction to general tridiagonal form*, ACM Trans. Math. Software, 18 (1992), pp. 392–400.

[5] G.W. Howell, *Efficient computation of eigenvalues of randomly generated matrices*, Appl. Math. Comput., 66 (1994), pp. 9–24.

[6] B.N. Parlett and C. Reinsch, *Balancing a matrix for calculation of eigenvalues and eigenvectors*, Numer. Math., 13 (1969), pp. 293–304.

[7] R.S. Martin and J.S. Wilkinson, *Similarity reduction of a general matrix to Hessenberg form*, Numer. Math., 12 (1968), pp. 349–368.

[8] B. Parlett, *Laguerre's method applied to the matrix eigenvalue problem*, Math. Comp., 18 (1964), pp. 464–485.

[9] R.C. Ward, *The QR algorithm and Hyman's method on vector computers*, Math. Comp., 30 (1976), pp. 132–142.

[10] L.V. Foster, *Generalizations of Laguerre's method: Higher order methods*, SIAM J. Numer. Anal., 18 (1981), pp. 1004–1018.

[11] J.H. Wilkinson, *The Algebraic Eigenvalue Problem*, Oxford University Press, Oxford, UK, 1965.

[12] R.T. Gregory and D.L. Karney, *A collection of Matrices for Testing Computational Algorithms*, John Wiley, New York, 1969.

[13] L.M. Delves and J.N. Lyness, *A numerical method for locating the zeros of an analytic function*, Math. Comp., 21 (1967), pp. 543–560.

SIAM J. SCI. COMPUT.
Vol. 22, No. 3, pp. 835–852
© 2000 Society for Industrial and Applied Mathematics

RESIDUAL REPLACEMENT STRATEGIES FOR KRYLOV SUBSPACE ITERATIVE METHODS FOR THE CONVERGENCE OF TRUE RESIDUALS*

HENK A. VAN DER VORST† AND QIANG YE‡

Abstract. In this paper, a strategy is proposed for alternative computations of the residual vectors in Krylov subspace methods, which improves the agreement of the computed residuals and the true residuals to the level of $O(\mathbf{u})\|A\|\|x\|$. Building on earlier ideas on residual replacement and on insights in the finite precision behavior of the Krylov subspace methods, computable error bounds are derived for iterations that involve occasionally replacing the computed residuals by the true residuals, and they are used to monitor the deviation of the two residuals and hence to select residual replacement steps, so that the recurrence relations for the computed residuals, which control the convergence of the method, are perturbed within safe bounds. Numerical examples are presented to demonstrate the effectiveness of this new residual replacement scheme.

Key words. Krylov subspace methods, finite precision, residuals, residual replacement

AMS subject classifications. 65F10, 65G50

PII. S1064827599353865

1. Introduction.

Krylov subspace iterative methods for solving a large linear system $Ax = b$ typically consist of iterations that recursively update approximate solutions x_n and the corresponding residual vectors r_n $(= b - Ax_n)$. They can be written in a general form as follows.

ALGORITHM 1. Template for Krylov subspace methods.
Input: an initial approximation x_0; $r_0 = b - Ax_0$;
For $n = 1, 2, \ldots$ until convergence
    generate a correction vector q_n by the method;
    $x_n = x_{n-1} + q_n$,
    (the vector x_n does not occur in other statements),
    $r_n = r_{n-1} - Aq_n$
End for

Most Krylov subspace iterative methods, including the conjugate gradient method (CG) [13], the biconjugate gradient method (BiCG) [4, 14], CGS [21], and BiCGSTAB [24], fit in this framework (see [2, 12, 17] for other methods).

In exact arithmetic, the recursively defined r_n in Algorithm 1 is exactly the residual for the approximate solution x_n, because $b - Ax_n - r_n = b - Ax_{n-1} - r_{n-1} = b - Ax_0 - r_0 = 0$. In a floating point arithmetic, however, the round-off patterns for x_n and r_n will be different. It is important to note that any error made in the computation of x_n is not reflected by a corresponding error in r_n, or in other words, computational errors to x_n do not force the method to correct, since x_n has no influ-

*Received by the editors April 9, 1999; accepted for publication March 12, 2000; published electronically August 31, 2000.
http://www.siam.org/journals/sisc/22-3/35386.html
†Department of Mathematics, Utrecht University, P.O. Box 80010, NL-3508 TA Utrecht, The Netherlands (vorst@math.uu.nl).
‡Department of Mathematics, University of Manitoba, Winnipeg, Manitoba, Canada R3T 2N2 (ye@cc.umanitoba.ca). Current address: Department of Mathematics, University of Kentucky, Lexington, Kentucky 40506-0027 (qye@ms.uky.edu). This research was supported by grants from the University of Manitoba Research Development Fund and the Natural Sciences and Engineering Research Council of Canada.

ence on the iteration process. This leads to the well known situation that $b-Ax_n$ and r_n may differ significantly. This phenomenon has been extensively discussed in the literature; see [20, 11, 12, 19] and the references cited therein. Indeed, if we denote the computed results of x_n, r_n by $\hat{x}_n, \hat{r}_n$, respectively (but we still use q_n to denote the computed update vector of the algorithm), then we have

(1) $\hat{x}_n = fl(\hat{x}_{n-1} + q_n) = \hat{x}_{n-1} + q_n + \psi_n, \qquad |\psi_n| \le \mathbf{u}|\hat{x}_n| + O(\mathbf{u}^2),$
(2) $\hat{r}_n = fl(\hat{r}_{n-1} - Aq_n) = \hat{r}_{n-1} - Aq_n + \eta_n, \quad |\eta_n| \le \mathbf{u}(|\hat{r}_n| + N|A||q_n|) + O(\mathbf{u}^2),$

where $fl(z)$ denotes the computed result of z in finite arithmetic, the absolute value and inequalities on vectors are componentwise, $\mathbf{u}$ is the machine roundoff unit, and N denotes the maximum number of nonzero entries per row of A. The vectors ψ_n and η_n represent rounding error terms, and they can be bounded by a straightforward error analysis (see section 3 for details). In particular, the relations (1) and (2) show that ψ_n and η_n depend only on the iteration vectors $\hat{x}_n$, $\hat{r}_n$, and q_n.

We will call $b - A\hat{x}_n$ the *true residual* for the approximation $\hat{x}_n$ and call $\hat{r}_n$, as obtained by recurrence formula (2), the *computed residual* (or the *updated residual*). Then the difference between the two satisfies (using the finite precision recurrences (1) and (2))

$$\begin{aligned} b - A\hat{x}_n - \hat{r}_n &= b - A\hat{x}_{n-1} - \hat{r}_{n-1} - A\psi_n - \eta_n, \\ &= -\Sigma_{i=1}^{n}(A\psi_i + \eta_i), \end{aligned}$$

where we assume for now that $b-Ax_0-r_0 = 0$. Hence, the difference between the true and the updated residuals is a result of accumulated rounding errors. In particular, a significant deviation of $b - A\hat{x}_n$ from $\hat{r}_n$ may be expected, if there is a $\hat{x}_i$ or $\hat{r}_i$ with large norm during the iteration (a not uncommon situation for BiCG and CGS). On the other hand, even when all ψ_i and η_i are small (as is common for CG), but if it takes a relatively large number of iterations for convergence, the sheer accumulation of ψ_i and η_i could also lead to a nontrivial deviation.

What makes all this so important is that, in a finite precision implementation, the sequence $\hat{r}_n$ satisfies, almost to machine precision $\mathbf{u}$, its defining recurrence relation, and as was observed for many Krylov subspace methods, this is the driving force behind convergence of $\hat{r}_n$ [10, 11, 15, 16, 19, 22]. Indeed, residual bounds have been obtained in [22] for CG and BiCG, which show that even a significantly perturbed recurrence relation (with perturbations much larger than the machine precision) usually still leads to eventual convergence of the computed residuals. This theoretical insight has been our motivation and justification for the residual replacement scheme to be presented in section 2.1. On the other hand, the true residual $b - A\hat{x}_n$ itself has no self-correcting mechanism for convergence, mainly because any perturbation made to $\hat{x}_n$ does not have an effect on the iteration parameters, whereas errors in $\hat{r}_n$ immediately lead to other iteration parameters.

Thus, in a typical convergent iteration process, $\hat{r}_n$ converges to a level much smaller than $\mathbf{u}$ eventually, but the true residual $b-A\hat{x}_n$ can only converge to the level dictated by $\Sigma_{i=1}^{n}(A\psi_i + \eta_i)$, since

$$b - A\hat{x}_n = \hat{r}_n - \Sigma_{i=1}^{n}(A\psi_i + \eta_i).$$

Namely, when $\hat{r}_n$ is still bigger than the accumulated error $\Sigma_{i=1}^{n}(A\psi_i + \eta_i)$, $b - A\hat{x}_n$ agrees well with $\hat{r}_n$ in magnitude, but when $\hat{r}_n$ has converged to a level that is smaller than the accumulated error, then $b - A\hat{x}_n \sim \Sigma_{i=1}^{n}(A\psi_i + \eta_i)$ is just the accumulated

error and has no agreement at all with $\hat{r}_n$. In summary, a straightforward implementation would reduce the true residuals at best to $\Sigma_{i=1}^{n}(A\psi_i + \eta_i)$. A bound for this has been obtained in [11] and it is called the *attainable accuracy*. We note that this term could be significant even if only one of ψ_i or η_i is large, or if n is large.

The above problems become most serious in methods such as CGS and BiCG where intermediate $\hat{x}_n$ and $\hat{r}_n$ can have very large norm, and this may result in a large ψ_n or η_n. Several popular methods, such as BiCGSTAB [24], BiCGSTAB(ℓ) [18], QMR [7], TFQMR [5], and composite step BiCG [1], have been developed to reduce the norm of $\hat{r}_n$ (see [6] for details). We note that controlling the size of $\|\hat{r}_n\|$ alone does not solve the deviation problem in all situations, as intermediate vectors in these methods may still have a large norm which, while invisible in $\|\hat{r}_n\|$, could still have significant effects to the difference between the *true* and *computed* residuals. In any case, the accumulation of tiny errors over a long iteration may also result in a nontrivial deviation.

A simple approach for solving the deviation problem is to replace the computed residuals by the true residuals at some iteration step to restore the agreement. Then the deviation at subsequent steps will be the error accumulation after that iteration only. This includes a complete replacement strategy that simply computes r_n by $b - Ax_n$ at every iteration, and a periodic replacement strategy that updates r_n by $b - Ax_n$ only at intervals of the iteration count. While such a strategy maintains agreement of the two kinds of residuals, it turns out that the convergence of the r_n may deteriorate (as we will see, it may result in unacceptably large perturbations to the Lanczos recurrence relation for the residual vectors that steers the convergence; see section 2.3). Recently, Sleijpen and van der Vorst [19], motivated by suggestions made by Neumaier (see [12, 19]), introduced an efficient replacement scheme that includes a so-called *flying-restart* procedure. It was demonstrated that this new residual replacement strategy can be very effective in the sense that it can improve the convergence of the true residuals by several orders of magnitude. For practical implementations, such a strategy is very useful because it leads to meaningful residuals and this is important for stopping the iteration process at the right point. Of course, one could, after termination of the classical process, simply test the true residual, but the risk is that the true residual stagnated already long before termination, so that much work has been done in vain.

The present paper will follow the very same idea of replacing the computed residual by the true residual at selected steps, in order to maintain close agreement between the two residuals, but we propose a simpler strategy so that the replacement is done only when it is necessary and at phases in the iteration where it is harmless, that is, when convergence mechanism for $\hat{r}_n$ is not destroyed. Specifically, we shall present a rigorous error analysis for iterations with residual replacement and we will propose computable bounds for the deviation between the computed and true residuals. This will be used to select the replacement phases in the iteration in such a way that the Lanczos three term recurrence among $\hat{r}_n$ is sufficiently well maintained. For the resulting strategy, it will be shown that, provided that the computed residuals converge, the true residual will converge to the level $O(\mathbf{u})\|A\|\|x\|$, the smallest level that one can expect for an approximation.

We note that, in many practical applications, one is only interested in a modest reduction of the residual norm, rather than to the level of $O(\mathbf{u})\|A\|\|x\|$ considered here, but even in such cases, the true residual could potentially stagnate at a level above the desirable threshold if there is sufficiently large growth of intermediate vec-

tors. This is not necessarily a result of ill-conditioning of A but could simply arise when a Ritz value in the underlying Lanczos process for BiCG, as determined by the initial approximation, is very close to 0. Although such a situation must be rare, it poses a risk to direct implementations. Therefore, it would still be beneficial to use the residual replacement scheme as a guard in such applications, given that it invokes very little extra cost.

The paper has been organized as follows. In section 2, we develop a refined residual replacement strategy and we discuss some strategies that have been reported by others. We give an error analysis in section 3, and we derive some bounds for the deviation to be used in the replacement condition. We present a complete implementation in section 4. It turns out that the residual replacement strategy can easily be incorporated in existing codes. Some numerical examples are reported in section 5, and we finish with remarks in section 6.

The vector norm used in this paper is one of the 1, 2, or ∞-norm.

2. Residual replacement strategy. In this section, we develop a replacement strategy that maintains the convergence of the true residuals. A formal analysis is postponed to the next section. The specific iterative method can be any of those that fit in the general framework of Algorithm 1. Throughout this paper, we shall consider only iteration processes for which the computed residual $\hat{r}_n$ converges to a sufficiently small level.

As mentioned in section 1, we follow the basic idea to replace the computed residual $\hat{r}_m$ by the true residual $fl(b - A\hat{x}_m)$ at some selected steps $m = m_1, m_2, \ldots, m_k$. We will refer to such an iteration step as one where *residual replacement* occurs. Hence, the residual generated at an arbitrary step n could be either the usual updated residual $\hat{r}_n = fl(\hat{r}_{n-1} - Aq_{n-1})$ or the true residual $fl(b - A\hat{x}_n)$, depending on whether replacement has taken place or not at step n. In order to distinguish the two possible formulations, we denote by r_n the residual obtained at step n of the process with the replacement strategy, that is,

$$r_n = \begin{cases} fl(b - A\hat{x}_n), & \text{if } n = m_1, m_2, \ldots, m_k, \\ \hat{r}_n = fl(r_{n-1} - Aq_{n-1}), & \text{otherwise.} \end{cases}$$

With the residual replacement at step m ($m = m_1, m_2, \ldots, m_k$), the residual deviation is immediately reduced to

$$\delta_m \equiv b - A\hat{x}_m - r_m = b - A\hat{x}_m - fl(b - A\hat{x}_m) = -\xi_m,$$

and it can be shown (see Lemma 1 of section 2.2) that $|\xi_m| \leq \mathbf{u}(|r_m| + N|A||\hat{x}_m|) + O(\mathbf{u}^2)$. For the subsequent iterations $n > m$, but before the next replacement step, we clearly have that

$$\begin{aligned} \delta_n &= b - A\hat{x}_n - r_n = b - A\hat{x}_m - r_m - \Sigma_{i=m+1}^{n}(A\psi_i + \eta_i) \\ &= -\xi_m - \Sigma_{i=m+1}^{n}(A\psi_i + \eta_i). \end{aligned} \tag{3}$$

Therefore, the accumulated deviation before step m has no effect to the deviation after updating ($n > m$). However, in order for such a strategy to succeed, two conditions must be met, namely,

- the computed residual r_n should preserve the convergence mechanism of the original process that has been steered by the $\hat{r}_n$ vectors;
- from the last updating step m to the termination step K, the accumulated error $\Sigma_{i=m+1}^{K}(A\psi_i + \eta_i)$ should be small relative to $\mathbf{u}(|r_m| + N|A||\hat{x}_m|)$, which is the upperbound for $|\xi_m|$.

We discuss in the next two subsections how to satisfy these two objectives.

2.1. Maintaining convergence of computed residuals. In order that r_n maintains the convergence mechanism of the original updated residuals, it should preserve the property that gives rise to the convergence of the original $\hat{r}_n$. We therefore need to identify the properties that lead to convergence of the iterative method *in finite precision arithmetic*. While this may be different for each individual method, it has been observed for several Krylov subspace methods (including CG [11, 22], BiCG [22], CGS, BiCGSTAB, and BiCGSTAB(ℓ) [19]) that the recurrence $r_n = r_{n-1} - Aq_n$ and a similar one for q_n is satisfied almost to machine precision and this small local error is one of the properties behind the convergence of the computed residuals. Furthermore, the analysis of [22] suggests that convergence is well maintained even when the recurrence equations are perturbed with perturbations that are significantly greater than the machine precision. This latter property is the basis for our residual replacement strategy. Therefore, we briefly discuss this perturbation phenomenon for BiCG (or CG), as presented in [22].

Consider the BiCG iteration which contains $r_n = r_{n-1} - \alpha_{n-1}Ap_{n-1}$ and $p_n = r_n + \beta_n p_{n-1}$. In finite arithmetic, $\hat{r}_n$ and $\hat{p}_n$, which denote the computed results of r_n and p_n, respectively, satisfy the perturbed recurrence

$$\hat{r}_n = \hat{r}_{n-1} - \alpha_{n-1}A\hat{p}_{n-1} + \eta_n \quad \text{and} \quad \hat{p}_n = \hat{r}_n + \beta_n\hat{p}_{n-1} + \tau_n,$$

where η_n and τ_n are rounding error terms that can be bounded in terms of $\mathbf{u}$. Combining these two equations, we obtain the following perturbed matrix equation in a normalized form

$$AZ_n = Z_nT_n - \frac{1}{\alpha'_n}\frac{\hat{r}_{n+1}}{\|\hat{r}_1\|}e_n^T + F_n \quad \text{with} \quad Z_n = \left[\frac{\hat{r}_1}{\|\hat{r}_1\|}, \ldots, \frac{\hat{r}_n}{\|\hat{r}_n\|}\right], \tag{4}$$

where T_n is an invertible tridiagonal matrix,[1] $\alpha'_n = \|\hat{r}_n\|\alpha_n/\|r_1\| = e_n^T T_n^{-1} e_1$ and $F_n = [f_1, \ldots, f_n]$ with

$$f_n = \frac{A\tau_n}{\|\hat{r}_n\|} + \frac{1}{\alpha_n}\frac{\eta_{n+1}}{\|\hat{r}_n\|} - \frac{\beta_n}{\alpha_{n-1}}\frac{\eta_n}{\|\hat{r}_n\|}. \tag{5}$$

We note that (4) is just an equation satisfied by an exact BiCG iteration under a perturbation F_n. In particular, detailed bounds on τ_n and η_n will, under some mild assumptions, lead to $F_n \sim O(\mathbf{u})$.

The main result of [22] states that if a sequence $\hat{r}_n$ satisfies (4) and Z_{n+1} has full rank, then we have

$$\|\hat{r}_{n+1}\| \le (1 + K_n) \min_{p\in\mathcal{P}_n, p(0)=1} \|p(A + \Delta A_n)\hat{r}_1\|, \tag{6}$$

where $K_n = \|(AZ_n - F_n)T_n^{-1}\|\,\|Z_{n+1}^+\|$ and $\Delta A_n = -F_n Z_n^+$. The case $F_n = 0$ reduces to the known theoretical bound for the exact BiCG residuals [1]. Therefore, even when $\hat{r}_n$ and its exact counterpart are completely different, their norms are bounded by similar quantities and are usually comparable. Of course, in both cases, the bounds depend on the quality of the constructed basis. More importantly, a closer examination of the bound reveals that even if the perturbation F_n is in magnitude much larger than $\mathbf{u}$, the quantities in the bound, and thus $\|\hat{r}_{n+1}\|$, may not be significantly affected. Indeed, in [22] numerical experiments were presented, where relatively large

[1] We assume that no breakdowns of the iteration process have occurred.

artificial random perturbations had been injected to the recurrence for r_n; yet it did not significantly affect the convergence mechanism.

An implication of this analysis is that, regardless of how $\hat{r}_n$ is generated but as long as it satisfies (4), its norm can be bounded by (6). Hence, we can replace $\hat{r}_n$ by $r_n = fl(b - Ax_n)$ when $\eta_n = r_n - (r_{n-1} - Aq_n)$ are not too large relative to $\|r_n\|$ and $\|r_{n-1}\|$ (see (5)), and we may still expect it to converge in a similar fashion. Indeed, this criterion explains why the residual replacement strategies like $r_n = fl(b - Ax_n)$ work sometimes but do not work always (see section 2.3). Here, it will be used to determine when it is safe to replace $\hat{r}_n$ by $r_n = fl(b - Ax_n)$. We note that the above discussion is for BiCG, but the phenomenon it reveals seems to be valid for many other methods, especially for those methods that are based on BiCG (CGS, BiCGSTAB, and others).

Now we consider the case that residual replacement is carried out at step m, that is, $r_m = fl(b - A\hat{x}_m) = b - A\hat{x}_m + \xi_m$. It follows from the definition of δ_m and $\hat{r}_m$ that $b - A\hat{x}_m = \hat{r}_m + \delta_m = r_{m-1} - Aq_{m-1} + \eta_m + \delta_m$. So, the updated residual r_m satisfies

$$r_m = r_{m-1} - Aq_{m-1} + \eta'_m \quad \text{with} \quad \eta'_m = \eta_m + \delta_m + \xi_m. \tag{7}$$

Thus depending on the magnitude of $\|\eta'_m\|$ relative to $\|r_m\|$ and $\|r_{m-1}\|$, the use of $r_m = fl(b - A\hat{x}_m)$ may result in large perturbations to the recurrence relation. Therefore, a residual replacement strategy should ensure that the replacement is only done when $\|\eta'_m\|/\min\{\|r_m\|, \|r_{m-1}\|\}$ is not too large.

In a typical iteration, as the iteration proceeds, $\|\delta_n\|$, and hence $\|\eta'_n\|$, increases while $\|\hat{r}_n\|$ decreases. Replacement will reduce δ_n but, in order to maintain the recurrence relation, it should be carried out before $\|\eta'_n\|$ becomes too large relative to $\|\hat{r}_n\|$. For this reason, we propose to set a threshold ϵ and carry out a replacement when $\|\eta'_n\|/\|\hat{r}_n\|$ reaches the threshold. To be precise, we replace the residual at step n by $r_n = b - Ax_n$, if

$$\|\eta'_{n-1}\| \le \epsilon\|\hat{r}_{n-1}\| \quad \text{and} \quad \|\eta'_n\| > \epsilon\|\hat{r}_n\|. \tag{8}$$

We note that, in principle, residual replacement can be carried out for all steps up to where $\|\eta'_n\|$ reaches a certain point. However, from the stability point of view, it is preferred to generate the residual by the recurrence as much as possible, since $\|\eta'_n\|$ is generally bigger than the recurrence rounding error $\|\eta_n\|$ (of order $\mathbf{u}$).

2.2. Groupwise solution updating to reduce error accumulations. From the discussions of section 2.1, we learn that residual replacement should only be carried out up to a certain point. In this subsection, we will discuss how to maintain, after the last replacement, the deviation at the order of $\mathbf{u}|A||x_n|$, in which case x_n is a backward stable solution. Note that, for any x_n, $\mathbf{u}\|A\|\|x_n\|$ is the lowest value one can expect for its residual. This is simply because even with the exact solution x, both $b - A(fl(x))$ and $fl(b - Ax) \sim \mathbf{u}|A||x|$.

If $m = m_k$ is the last updating step, which means that we are in the final phase of the iteration process, then, because of (3), the deviation at step $n > m$ is

$$\delta_n = -\xi_m - \Sigma_{i=m+1}^{n}\eta_i - \Sigma_{i=m+1}^{n}A\psi_i.$$

From our updating condition, we have that $\|r_n\| \le \|\eta'_n\|/\epsilon \sim O(\frac{\mathbf{u}}{\epsilon})$. So, if ϵ is chosen not too close to $\mathbf{u}$, $\|r_n\|$ is small and $\hat{x}_n \sim x$ for $n \ge m$. We now discuss the three different parts of δ_n. The discussion here is only to motivate the groupwise updating strategy; a more rigorous analysis will be given in the next section.

- $|\xi_m| \le \mathbf{u}(|r_m| + N|A||\hat{x}_m|) \sim O(\mathbf{u})|A||x|$.
- Since $|\hat{r}_i| \ll |b| \le |A||x|$ and $|\eta_i| \sim \mathbf{u}|\hat{r}_i|$, we have that $\Sigma_{i=m+1}^{n}|\eta_i| \ll O(\mathbf{u})|A||x|$.
- For the ψ_i part, $|\psi_i| \sim \mathbf{u}|\hat{x}_i| \sim \mathbf{u}|x|$. Hence, $\Sigma_{i=m+1}^{n}|A\psi_i| \sim \Sigma_{i=m+1}^{n}\mathbf{u}\|A\|\|x\| = (n-m)\mathbf{u}\|A\|\|x\|$. If $n-m$ is large, the accumulation of errors over $n-m$ steps can be significant. We note that this is the same type of error accumulation in evaluating a sum $S = \Sigma_{i=1}^{\infty}c_i$ of small numbers by direct recursive additions, which can fortunately be corrected through appropriately grouping the arithmetic operations as $S_1+S_2+\cdots = (c_1+\cdots+c_{m_1})+(c_{m_1+1}+\cdots+c_{m_2})+\cdots$ with terms of similar order of magnitude in the same group S_i and $S_1 \gg S_2 \gg \cdots$. In this way, the rounding errors associated with a large number of additions inside a group S_i is of the magnitude of $\mathbf{u}S_i$, which can be much smaller than $\mathbf{u}S$. The same technique can be adopted for computing x_n as

$$x_n = x_0 + \Sigma_{i=1}^{n}q_i = x_0 + (q_1 + \cdots + q_{m_1}) + (q_{m_1+1} + \cdots + q_{m_2}) + \cdots .$$

Specifically, the recurrence for x_n can be carried out in the following equivalent form:

Groupwise solution update: $z = x_0$; $\hat{x}_0 = 0$;
For $n = 1, 2, \ldots$ until convergence
  $\hat{x}_n = fl(\hat{x}_{n-1} + q_n) = \hat{x}_{n-1} + q_n + \psi_n$
  if $n = m_i$ (i.e., group update)
    $z = fl(z + \hat{x}_n) = z + \hat{x}_n + \zeta_n$,
    $\hat{x}_n = 0$
  end if
End for

Such a groupwise update scheme has been suggested by Neumaier, and it has been worked out by Sleijpen and van der Vorst (see [19] for both references). By doing so, the error in the local recurrence is reduced. Indeed, for $i \ge m$, $|\hat{x}_i| = |z+\hat{x}_i-z| \sim |x-z| \ll |x|$. Then $|\psi_i| \sim \mathbf{u}|\hat{x}_i|$ (instead of $\mathbf{u}|x_i|$). Hence, $\Sigma_{i=m+1}^{n}|A\psi_i| \ll (n-m)\mathbf{u}|A||x|$.

In summary, with groupwise updating of the approximated solution, all three parts of δ_n can be maintained at the level of $\mathbf{u}|A||x|$. We mention that groupwise updating can also be used to obtain better performance of a code for modern architectures, because it allows for level-3 BLAS operations. This has been suggested in [23, p. 52, note 5].

2.3. Some other residual replacement strategies. We briefly comment on some other residual replacement strategies.

For the naive strategy of "replacing always" (the residuals are computed always as $b - Ax_n$) or for "periodic replacement" (update periodically at every ℓ steps), replacement is carried out throughout the iteration, even when $\|r_n\|$ is very small. This, as we know, may result in large perturbations to the recurrence equations relative to $\|r_n\|$, since $|\eta'_n|$ is at least $|\xi_n| \sim \mathbf{u}|A||x_n|$; see (7). In that case, as $\|r_n\|$ decreases, the recurrence relation may be perturbed too much and hence the convergence property deteriorates. This is the typical behavior observed in such implementations.

We note that if ξ_n can be made to decrease as $\|r_n\|$ does, then replacement can be carried out at later stages of the iterations. This leads to the strategy of "flying-restart" of Sleijpen and van der Vorst [19], which significantly reduces ξ_n, and hence

η'_n, at a replacement step. In the flying-restart strategy b is replaced by $fl(b - Ax_m)$ at some but not all of the residual replacement steps (say $m = n_1, n_2, \ldots, n_l$), in addition to the residual replacement $r_m = fl(b - Ax_m)$. The advantage of this is that, at the flying-restart step n_{i+1}, the residual is updated by $r_{n_{i+1}} = fl(r_{n_i} - A\hat{x}_{n_{i+1}}) = r_{n_i} - A\hat{x}_{n_{i+1}} + \xi_{n_{i+1}}$ (noting that $b \leftarrow r_{n_i}$) and $|\xi_{n_{i+1}}| \sim \mathbf{u}(|r_{n_i}| + |A||\hat{x}_{n_{i+1}}|)$. Then

$$|r_{n_i} - A\hat{x}_{n_{i+1}} - r_{n_{i+1}}| = |\zeta_{n_{i+1}}| \sim \mathbf{u}(|r_{n_i}| + |A||\hat{x}_{n_{i+1}}|),$$

which decreases as r_{n_i} and $\hat{x}_{n_{i+1}}$ decrease. This is the term that determines the perturbation to the recurrence and can be kept small relative to r_n. However, the deviation satisfies

$$b - Ax_{n_{i+1}} - r_{n_{i+1}} = b - Ax_{n_i} - r_{n_i} - \xi_{n_{i+1}}$$

(assuming $x_{n_{i+1}} = x_{n_i} + \hat{x}_{n_{i+1}}$). Namely, the deviation at each flying-restart step carries forward to the later steps. Therefore flying-restart should only be used at carefully selected steps where $\xi_{n_i} \sim \mathbf{u}(\|b\| + N\|A\|\|x\|)$. However, it is not easy to identify a condition to monitor this. It is also necessary to have two different conditions for the residual replacement and flying-restart. Fortunately, our discussion in the last two subsections shows that carrying out replacement carefully at some selected steps, in combination with groupwise update, is usually sufficient. We shall not pursue the flying-restart idea further in this paper.

3. Error analysis of the residual replacement scheme. In this section, we formally analyze the residual replacement strategy as developed in section 2.1 (and presented in Algorithm 2 below). In particular, we develop a computable bound for $\|\delta_n\|$ and $\|\eta'_n\|$, that can be used for the implementation of the residual replacement condition.

We first summarize residual replacement strategy in the following algorithm, written in a form that identifies relevant rounding errors for later theoretical analysis.

ALGORITHM 2. Iterative method with residual replacement.
Given an initial approximation $z = x_0$ (a floating point vector);
set $\hat{x}_0 = 0$; $r_0 = fl(b - Ax_0) = b - Ax_0 + \xi_0$;
For $n = 1, 2, \ldots$ until convergence
    generate a correction vector q_n by the method;
    $\hat{x}_n = fl(\hat{x}_{n-1} + q_n) = \hat{x}_{n-1} + q_n + \psi_n$,
    $\hat{r}_n = fl(r_{n-1} - Aq_n) = r_{n-1} - Aq_n + \eta_n$,
    if residual replacement condition (8) holds
        $z = fl(z + \hat{x}_n) = z + \hat{x}_n + \zeta_n$,
        $\hat{x}_n = 0$,
        $r_n = fl(b - Az) = b - Az + \xi_n$
    else
        $r_n = \hat{r}_n$
    end if
    (denote but not compute $x_n = z + \hat{x}_n$ and $\delta_n = b - Ax_n - r_n$)
End for
$z = fl(z + \hat{x}_n) = z + \hat{x}_n + \zeta_n$

Note that x_n and δ_n are theoretical quantities as defined by the formulas and are not to be computed. The vectors $\psi_n, \eta_n, \zeta_n, \xi_n$ represent rounding error terms, due to finite precision arithmetic.

At step n of the iterative method, q_n is computed in finite precision arithmetic by the algorithm. However, the rounding errors involved in the computation of q_n are irrelevant for the deviation of the two residuals, which solely depends on the different treatment of q_n in the recurrences for r_n and x_n.

Throughout this paper, we assume that A is a floating point matrix. Our error analysis is based on the following standard model for roundoff errors in basic matrix computations [8, p. 66] (all inequalities are componentwise):

$$fl(x+y) = x + y + e \quad \text{with} \quad |e| \leq \mathbf{u}(|x+y|), \tag{9}$$

$$fl(Ax) = Ax + e \quad \text{with} \quad |e| \leq \mathbf{u}N|A||x| + O(\mathbf{u}^2), \tag{10}$$

where x, y are floating point vectors, N is a constant associated with the matrix-vector multiplication (for instance, the maximal number of nonzero entries per row of A).

It is easy to show that

$$fl(y + Ax) = y + Ax + e \quad \text{with} \quad |e| \leq \mathbf{u}(|y + Ax| + N|A||x|) + O(\mathbf{u}^2).$$

Using this, the following lemma, which includes (1) and (2), is obtained.

LEMMA 3.1. *The error terms in the computed recurrence of Algorithm* 2 *are bounded as follows:*

$$|\psi_n| \leq \mathbf{u}|\hat{x}_n| + O(\mathbf{u}^2), \tag{11}$$

$$|\eta_n| \leq \mathbf{u}(|\hat{r}_n| + N|A||q_n|) + O(\mathbf{u}^2). \tag{12}$$

For a step at which a residual replacement is carried out

$$|\zeta_n| \leq \mathbf{u}|x_n| + O(\mathbf{u}^2), \tag{13}$$

$$|\xi_n| \leq \mathbf{u}(|r_n| + N|A||x_n|) + O(\mathbf{u}^2). \tag{14}$$

Proof. From (9), we have that $|\psi_n| \leq \mathbf{u}|\hat{x}_{n-1} + q_n| \leq \mathbf{u}(|\hat{x}_n| + |\psi_n|)$. This leads to the bound for $|\psi_n|$. For a residual replacement step, the updated z is x_n by definition, that is, $x_n = z + \hat{x}_n + \zeta_n$. Therefore, $|\zeta_n| \leq \mathbf{u}|x_n| + O(\mathbf{u}^2)$. The bounds for η_n and ξ_n follow similarly. ∎

LEMMA 3.2. *Let m be the number of step at which a residual replacement is carried out and let $n > m$ denote a later step, but still before the next replacement step. Then, we have that*

$$\Sigma_{i=m+1}^{n}|\psi_i| \leq \mathbf{u}\Sigma_{i=m+1}^{n}|\hat{x}_i| + O(\mathbf{u}^2),$$

$$\Sigma_{i=m+1}^{n}|q_i| \leq (2+\mathbf{u})\Sigma_{i=m+1}^{n}|\hat{x}_i|,$$

and

$$\Sigma_{i=m+1}^{n}|\eta_i| \leq \mathbf{u}\Sigma_{i=m+1}^{n}|\hat{r}_i| + 2\mathbf{u}N|A|\Sigma_{i=m+1}^{n}|\hat{x}_i| + O(\mathbf{u}^2).$$

Proof. The first bound follows directly from Lemma 1. For $i \geq m+1$ we have that $q_i = \hat{x}_i - \hat{x}_{i-1} - \psi_i$. Noting that $\hat{x}_m = 0$, it follows that

$$\Sigma_{i=m+1}^{n}|q_i| \leq \Sigma_{i=m+1}^{n}(|\hat{x}_i| + |\hat{x}_{i-1}| + |\psi_i|) \leq (2+\mathbf{u})\Sigma_{i=m+1}^{n}|\hat{x}_i|.$$

Similarly,

$$\begin{aligned}\Sigma_{i=m+1}^{n}|\eta_i| &\le \mathbf{u}\Sigma_{i=m+1}^{n}(|\hat{r}_i| + N|A||q_i|) + O(\mathbf{u}^2)\\ &\le \mathbf{u}\Sigma_{i=m+1}^{n}|\hat{r}_i| + 2\mathbf{u}N|A|\Sigma_{i=m+1}^{n}|\hat{x}_i| + O(\mathbf{u}^2). \qquad \square\end{aligned}$$

We now consider the deviation of the two residuals.

LEMMA 3.3. *Let m be the number of an iteration step at which residual replacement is carried out and let $n > m$ denote a later iteration step, still before the next replacement step. Then, we have that $\delta_m = -\xi_m$ and*

$$\delta_n = \delta_{n-1} - (A\psi_n + \eta_n) = -\xi_m - \Sigma_{i=m+1}^{n}(A\psi_i + \eta_i).$$

Proof. At step m, by the definition of x_m in Algorithm 2, $x_m = z + \hat{x}_m = z$ with z being the updated z-vector and $\hat{x}_m = 0$. Furthermore, $r_m = fl(b - Az) = fl(b - Ax_m) = b - Ax_m + \xi_m$. Therefore $\delta_m = -\xi_m$. Hence, for the range of $n > m$, and before the next residual replacement step,

$$\begin{aligned}\delta_n &= b - Ax_n - r_n = b - A(z + \hat{x}_n) - \hat{r}_n\\ &= b - A(z + \hat{x}_{n-1} + q_n + \psi_n) - (\hat{r}_{n-1} - Aq_n + \eta_n)\\ &= \delta_{n-1} - A\psi_n - \eta_n\\ &= \delta_m - \Sigma_{i=m+1}^{n}(A\psi_i + \eta_i). \qquad \square\end{aligned}$$

With Lemma 2, we obtain the following computable bound on δ_n.

LEMMA 3.4. *Let m be the number of an iteration step at which residual replacement is carried out and let $n > m$ denote a later iteration step, still before the next replacement step. Then, we have $\|\delta_m\| \le \mathbf{u}(\|r_m\| + N\|A\|\|x_m\|) + O(\mathbf{u}^2)$ and*

$$\|\delta_n\| \le \mathbf{u}N\|A\|\|x_m\| + \mathbf{u}(1+2N)\|A\|\Sigma_{i=m+1}^{n}\|\hat{x}_i\| + \mathbf{u}\Sigma_{i=m}^{n}\|r_i\| + O(\mathbf{u}^2).$$

Proof. The bound for $\|\delta_m\|$ follows from that for ξ_m; see (14). From Lemma 2 and Lemma 3, it follows that

$$\begin{aligned}|\delta_n| &\le |\xi_m| + \Sigma_{i=m+1}^{n}(|A||\psi_i| + |\eta_i|)\\ &\le \mathbf{u}(|r_m| + N|A||x_m|) + |A|\mathbf{u}\Sigma_{i=m+1}^{n}|\hat{x}_i|\\ &\quad + \mathbf{u}\Sigma_{i=m+1}^{n}|\hat{r}_i| + 2\mathbf{u}N|A|\Sigma_{i=m+1}^{n}|\hat{x}_i| + O(\mathbf{u}^2)\\ &= \mathbf{u}N|A||x_m| + \mathbf{u}(1+2N)|A|\Sigma_{i=m+1}^{n}|\hat{x}_i| + \mathbf{u}\Sigma_{i=m}^{n}|r_i| + O(\mathbf{u}^2),\end{aligned}$$

which leads to the bound for δ_n in terms of norms. $\square$

We note that it is possible to obtain a sharper bound by accumulating the vectors in the bound for $|\delta_n|$. Our experiments do not show any significant advantage of such an approach. We next consider the perturbation to the recurrence.

THEOREM 3.5. *Consider step n of the iteration and let $m < n$ be the last step before n, at which a residual replacement is carried out. If replacement is also done at step n, then let $x'_n = fl(x_m + \hat{x}_n) = x_m + \hat{x}_n + \zeta_n$ be the computed approximate solution and $r'_n = fl(b - Ax'_n) = b - Ax'_n + \xi_n$ be the residual. Then the residual r'_n satisfies the following approximate recurrence:*

$$r'_n = r_{n-1} - Aq_n + \eta'_n, \tag{15}$$

where $\eta'_n = \eta_n + \delta_n - A\zeta_n + \xi_n$ and

$$\|\eta'_n\| \le \mathbf{u}\|A\|(1+2N)(\|x_m\| + \Sigma_{i=m+1}^{n}\|\hat{x}_i\|) + \mathbf{u}\Sigma_{i=m}^{n}\|r_i\| + O(\mathbf{u}^2). \tag{16}$$

Proof. First, in the notation of Algorithm 2, $x'_n = x_m + \hat{x}_n + \zeta_n = x_n + \zeta_n$. Then,

$$\begin{aligned} r'_n &= b - Ax_n - A\zeta_n + \xi_n \\ &= r_n + \delta_n - A\zeta_n + \xi_n \\ &= r_{n-1} - Aq_n + \eta_n + \delta_n - A\zeta_n + \xi_n \\ &= r_{n-1} - Aq_n + \eta'_n, \end{aligned}$$

where we have used that $b - Ax_n = r_n + \delta_n$ and $r_n = \hat{r}_n = r_{n-1} - Aq_n + \eta_n$. Furthermore, by Lemma 3,

$$\eta_n + \delta_n = \xi_m - \Sigma_{i=m+1}^{n} A\psi_i - \Sigma_{i=m+1}^{n-1}\eta_i.$$

Also, $\|A\zeta_n\| \leq \mathbf{u}\|A\|(\|x_m\| + \|\hat{x}_n\|)$ and $\|\xi_n\| \leq \mathbf{u}(\|r'_n\| + N\|A\|\|x'_n\|) + O(\mathbf{u}^2) \leq \mathbf{u}(\|r'_n\| + N\|A\|\|x_m\| + N\|A\|\|\hat{x}_n\|) + O(\mathbf{u}^2)$. Combining these three inequalities, and using that $r'_n = r_n + O(\mathbf{u})$, the bound on $\|\eta'_n\|$ is obtained as in Lemma 4. ∎

Note that bound (16) is computable at each iteration step. Therefore, we can implement the residual replacement criterion (8) with this bound instead of $\|\eta'_n\|$. We note that the factor 2 in the bound comes from the bound for q_i in Lemma 2, which is pessimistic since $q_i \sim \hat{x}_i$. Therefore, we can use the following d_n as an estimate for $\|\eta'_n\|$:

$$d_n \equiv \mathbf{u}N\|A\|(\|x_m\| + \Sigma_{i=m+1}^{n}\|\hat{x}_i\|) + \mathbf{u}\Sigma_{i=m}^{n}\|r_i\|. \tag{17}$$

Hence, we shall use the following residual replacement criterion, that is, residual replacement is done if

$$d_{n-1} \leq \epsilon\|\hat{r}_{n-1}\| \text{ and } \; d_n > \epsilon\|\hat{r}_n\|. \tag{18}$$

With this strategy, the replaced residual vector r_n satisfies the recurrence equation (15) with $\|\eta'_n\| \sim O(\epsilon)\|\hat{r}_n\|$. With this property, we consider situations where r_n converges. We now discuss convergence of the true residual.

THEOREM 3.6. *Consider Algorithm* 2 *with the residual replacement criterion* (18)*, and assume that the algorithm terminates at step* $n = K$ *with* $\|r_K\| < \mathbf{u}\|A\|\,\|x_K\|$*. Let* m *be the number of the last residual replacement iteration step before termination. If*

$$L = (K - m + 1)(1 + 2N)\|A\|\|A^{-1}\|(1 + 3/\epsilon) < 1/\mathbf{u}, \tag{19}$$

then

$$\begin{aligned} \|b - Ax_K\| &\leq \|r_K\| + \mathbf{u}N\|A\|\|x_K\|/(1 - \mathbf{u}L) + O(\mathbf{u}^2) \\ &\sim \mathbf{u}N\|A\|\|x_K\|. \end{aligned}$$

Proof. From (17), we have $d_K > \mathbf{u}N\|A\|(\|x_m\| + \|\hat{x}_K\|) \geq \mathbf{u}\|A\|\,\|x_K\|$. Furthermore, at the termination step, we have $\|r_K\| < \mathbf{u}\|A\|\,\|x_K\|$ and hence $d_K > \|r_K\| > \epsilon\|r_K\|$. Since m is the last updating step, we have for $n \geq m$, $d_n > \epsilon\|r_n\|$ as otherwise there would be another residual replacement after m. That implies $\|r_n\| < d_n/\epsilon \leq d_K/\epsilon$. Define

$$\tilde{d}_n \equiv \mathbf{u}N\|A\|\|x_m\| + \mathbf{u}\|A\|(1 + 2N)\Sigma_{i=m+1}^{n}\|\hat{x}_i\| + \mathbf{u}\Sigma_{i=m}^{n}\|\hat{r}_i\|,$$

which is an upper bound for $\|\delta_n\|$ (Lemma 4) and $\tilde{d}_n \geq d_n$. Then

$$\begin{aligned}\|\hat{x}_n\| &= \|x_n - x_m\| = \|A^{-1}((b - Ax_m) - (b - Ax_n))\| \\ &= \|A^{-1}(r_m + \delta_m - r_n - \delta_n)\| \\ &\leq \|A^{-1}\|(\|r_m\| + \|r_n\| + \|\delta_n - \delta_m\|) \\ &\leq \|A^{-1}\|(1 + 2/\epsilon)\tilde{d}_K + O(\mathbf{u}^2),\end{aligned}$$

where $\|\delta_n - \delta_m\| \leq \tilde{d}_n + O(\mathbf{u}^2) \leq \tilde{d}_K + O(\mathbf{u}^2)$. Thus,

$$\begin{aligned}\tilde{d}_K &= \mathbf{u}N\|A\|\|x_K - \hat{x}_K\| + \mathbf{u}(1 + 2N)\|A\|\Sigma_{i=m+1}^{K}\|\hat{x}_i\| + \mathbf{u}\Sigma_{i=m}^{K}\|\hat{r}_i\| \\ &\leq \mathbf{u}N\|A\|\|x_K\| + \mathbf{u}N\|A\|\|\hat{x}_K\| + \mathbf{u}(1 + 2N)\|A\|\Sigma_{i=m+1}^{K}\|\hat{x}_i\| + \mathbf{u}\Sigma_{i=m}^{K}\|\hat{r}_i\| \\ &\leq \mathbf{u}N\|A\|\|x_K\| + \mathbf{u}(K - m + 1)(1 + 2N)\|A\|\|A^{-1}\|(1 + 2/\epsilon)\tilde{d}_K \\ &\quad + \mathbf{u}(K - m + 1)\tilde{d}_K/\epsilon + O(\mathbf{u}^2) \\ &\leq \mathbf{u}N\|A\|\|x_K\| + \mathbf{u}(K - m + 1)(1 + 2N)\|A\|\|A^{-1}\|(1 + 3/\epsilon)\tilde{d}_K + O(\mathbf{u}^2) \\ &\leq \mathbf{u}N\|A\|\|x_K\| + \mathbf{u}L\tilde{d}_K + O(\mathbf{u}^2),\end{aligned}$$

which implies

$$\tilde{d}_K \leq \mathbf{u}N\|A\|\|x_K\|/(1 - \mathbf{u}L) + O(\mathbf{u}^2).$$

Thus the bound follows from

$$\|b - Ax_K\| \leq \|r_K\| + \|\delta_K\| \leq \|r_K\| + \tilde{d}_K + O(\mathbf{u}^2). \qquad \square$$

We add two remarks with respect to this theorem.

Remark 1. If the main condition (19) is satisfied, then the deviation, and hence the true residual, will remain at the level of $\mathbf{u}N\|A\|\|x_K\|$ at termination. Such an approximate solution is backward stable and it is the best one can expect. The condition suggests that ϵ should not be chosen too small. Otherwise, the replacement strategy will be terminated too early so that the accumulation after the last replacement might become significant. As can be expected, however, the theoretical condition is more restrictive than practically necessary and our numerical experience suggests that ϵ can be much smaller than what (19) dictates, without destroying the conclusion of the theorem.

Remark 2. On the other hand, in section 2.1 we have seen that ϵ controls perturbations to the recurrence of r_n, and for this reason it is desirable to choose it as small as possible. In our experience, there is a large range of ϵ around $\sqrt{\mathbf{u}}$ that balances the two needs.

4. Reliable implementation of iterative methods. In this section, we summarize the main results of the previous sections into a complete implementation. We also address some implementation issues.

It is easy to see from the definition of d_n (see (17)) that it increases except at the residual replacement steps when it is reset to $\mathbf{u}(N\|A\|\|x_m\| + \|r_m\|)$. Our residual replacement strategy is to reduce d_n whenever necessary (as determined by the replacement criterion) so as to keep it at the level of $\mathbf{u}N\|A\|\|x_K\|$ at termination. With the use of criterion (18), however, there are situations where the residual replacement is carried out in consecutive steps while d_n remains virtually unchanged, namely, when $\|r_n\|$ stays around $d_n/\epsilon \sim \mathbf{u}N\|A\|\|x_n\|/\epsilon$. From the stability point of view, it is preferred not to replace the residuals in such situations. To avoid unnecessary replacement in such cases, we impose as an additional condition that residual

replacement is carried out only when d_n has a nontrivial increase from the d_m of the previous replacement step m.

Therefore, we propose $d_n > 1.1d_m$ as a condition in addition to (18) for the residual replacement. The following scheme sketches a complete implementation.

ALGORITHM 3. Reliable implementation of Algorithm 1.

Input: an initial approximation $z = x_0$;
  a residual replacement threshold ϵ; an estimate of $N\|A\|$;
Set $r_0 = b - Ax_0$; $\hat{x}_0 = 0$; $d_{init} = d_0 = \mathbf{u}(\|r_0\| + N\|A\|\|x_0\|)$,
For $n = 1, 2, \ldots$ until convergence
  generate a correction vector q_n by the iterative method;
  $\hat{x}_n = \hat{x}_{n-1} + q_n$,
  $r_n = r_{n-1} - Aq_n$,
  $d_n = d_{n-1} + \mathbf{u}N\|A\|\,\|\hat{x}_n\| + \mathbf{u}\|r_n\|$,
  if $d_{n-1} \le \epsilon\|r_{n-1}\|$, $d_n > \epsilon\|r_n\|$ and $d_n > 1.1d_{init}$
    $z = z + \hat{x}_n$,
    $\hat{x}_n = 0$,
    $r_n = b - Az$,
    $d_{init} = d_n = \mathbf{u}(\|r_n\| + N\|A\|\|z\|)$
  end if
End for
$z = z + \hat{x}_n$

Remark. For this reliable implementation, we need to put a value for N (the maximal number of nonzero entries per row of A) and $\|A\|$. The number of nonzero entries may, in applications, vary from row to row, and selecting the maximum number may not be very realistic. In our experience with sparse matrices, the simple choice $N = 1$ still leads to a practical estimate d_n for $\|\delta_n\|$. In any case, we note that precise values are not essential, because the replacement threshold ϵ can be adjusted. We also need to choose this ϵ. Our extensive numerical testing (see section 5) suggests that $\epsilon \sim \sqrt{\mathbf{u}}$ is a practical criterion. However, there are examples where this choice leads to stagnating residuals at some unacceptable level. In such cases, choosing a smaller ϵ will regain the convergence to $O(\mathbf{u})$.

The presented implementation requires one extra matrix-vector multiplication when an replacement is carried out. Since only a few steps with replacement are required, this extra cost is marginal relative to the other costs. However, some savings can be made by selecting a slightly smaller ϵ and carrying out residual replacement at the step next to the one for which the residual replacement criterion is satisfied (cf. [19]). It also requires one extra vector storage for the groupwise solution update (for z) and computation of a vector norm $\|\hat{x}_n\|$ for the update of d_n ($\|r_n\|$ is usually computed in the algorithm for stopping criteria).

5. Numerical examples. In this section, we present some numerical examples to show how Algorithm 3 works and to demonstrate its effectiveness. We present our testing results for CG, BiCG, and CGS. All tests are carried out in MATLAB on a SUN Sparc-20 workstation, with $\mathbf{u} \approx 10^{-16}$.

In all examples, unless otherwise specified, the replacement threshold ϵ is chosen to be 10^{-8}. $\|A\|_\infty$ is explicitly computed and N is set to 1. In Examples 1 and 2, we also compare d_n and the deviation $\|\delta_n\|$.

Example 1. The matrix is a finite-difference discretization on a 64×64 grid for

$$-\nabla(a(x,y)\nabla u) = f(x,y) \text{ on } R = (0,1) \times (0,1),$$

with a homogeneous Dirichlet boundary condition. $a(x,y) = exp(y^2)$ and $f(x,y) = x^2y$. We apply CG and reliable CG (i.e., Algorithm 3) to solve this linear system and the convergence results are given in Figure 1.

In Figure 1 (and similarly in Figures 2 and 3 for the next example), we give in (a) the convergence history of the (normalized) computed residual for CG (solid line), the (normalized) true residuals for CG (dash line), and for reliable CG (dotted line). In (b), we also give the (normalized) deviations of the two residuals $\|\delta_n\| = \|b - Ax_n - r_n\|$ for CG (dash-dotted line) and for reliable CG (dotted line) and the bound d_n for reliable CG (in x-mark).

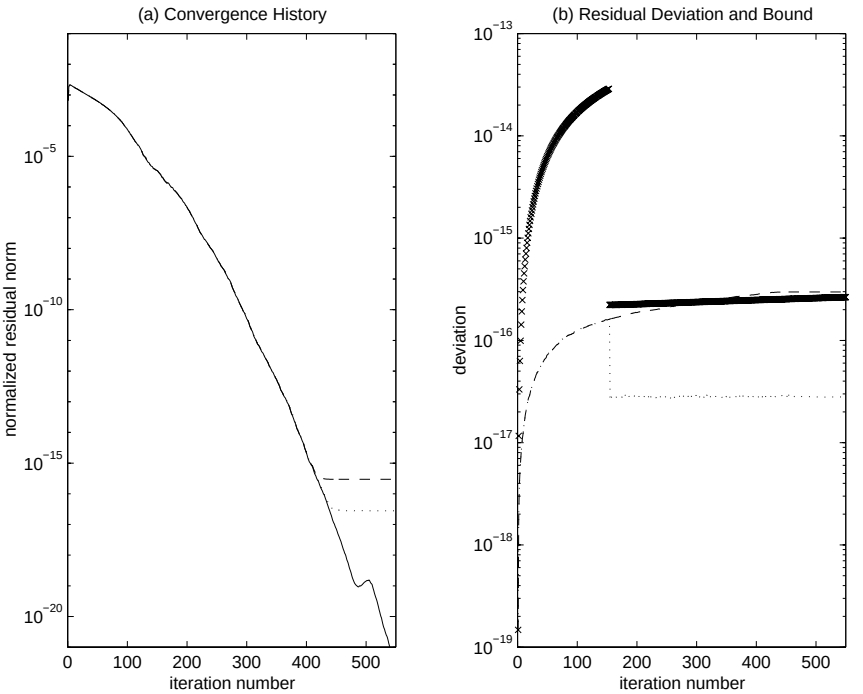


FIG. 1. *Example* 1. *CG:* (a) *solid—computed residual of CG; dash—true residual of CG; dotted—true residual of reliable CG;* (b) *dash-dotted—*$\|b - Ax_n - r_n\|$ *of CG; dotted—*$\|b - Ax_n - r_n\|$ *of reliable CG;* x*—bound* d_n *for reliable CG.*

This example is to illustrate that even for CG, our refined implementation can slightly improve the true residual. We note, however, that such an improvement is very minor and would only be useful when a solution of highest accuracy possible is wanted. Also note that in CG it is possible to estimate the A-norm of the true residual without explicitly computing it (see [9]).

We next consider applications to BiCG and CGS.

Example 2. The matrix is a finite-difference discretization on a 64×64 grid for the following convection diffusion equation:

$$-\triangle u + \gamma(xu_x + yu_y) + \beta u = f(x,y) \quad \text{on} \quad (0,1)^2,$$

with a homogeneous Dirichlet boundary condition. The function f is a constant. We consider BiCG and CGS for solving the linear systems with $\gamma = -250, \beta = 0$, and $\gamma = -10, \beta = 1$, respectively. The results are shown in Figure 2 for BiCG and in Figure 3 for CGS.

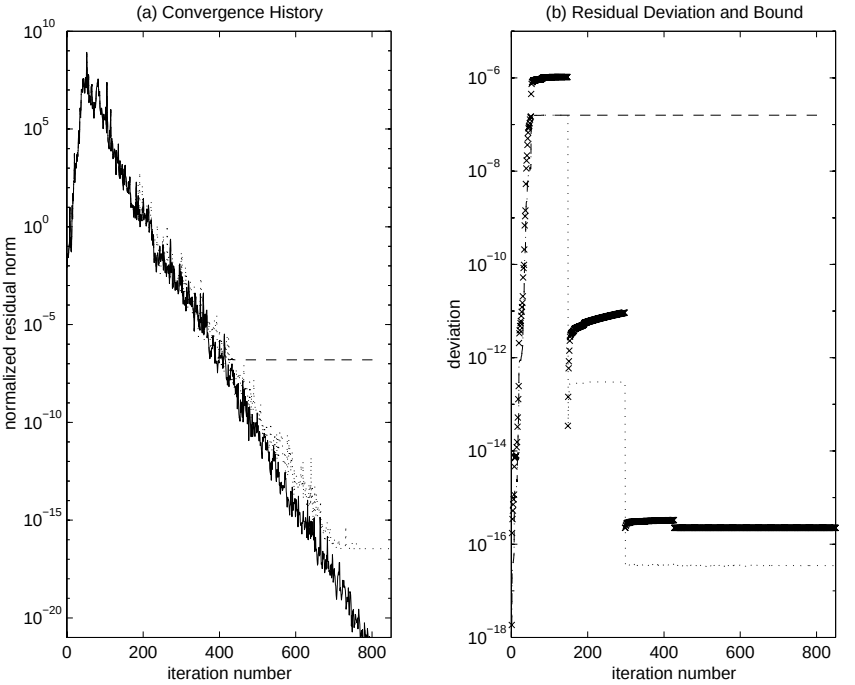


FIG. 2. *Example* 2. *BiCG:* (a) *solid—computed residual of BiCG; dash—true residual of BiCG; dotted—true residual of reliable BiCG;* (b) *dash—*$\|b - Ax_n - r_n\|$ *of BiCG; dotted—*$\|b - Ax_n - r_n\|$ *of reliable BiCG;* x*—bound* d_n *for reliable BiCG.*

In the above examples, we have observed the following typical convergence behavior. For the original implementations, the deviation increases and finally stagnates at some level, which is exactly where the true residual stagnates, while the computed residual continues to converge. With the reliable implementations, when the deviation increases to a certain level relative to r_n, a residual replacement is carried out and this reduces the error level. Eventually, the deviation and hence the true residual arrives at the level of $\mathbf{u}\|A\|\|x\|$. We also note that the bound d_n captures the behavior of $\|\delta_n\|$ very closely, although it may be an overestimate for δ_n by a few orders of magnitude. In all three cases, the final residual norms for the reliable implementation are smaller than the ones as obtained by the MATLAB function $A\backslash b$.

Example 3. In this case, we have tested the algorithm for BiCG (or CG if symmetric definite) and CGS on the Harwell–Boeing collection of sparse matrices [3]. We compare the original implementations, the reliable implementations, and the implementations of Sleijpen and van der Vorst [19] (based on their replacement criteria (16) and (18)). In Table 1, we give the results for those matrices for which the computed residuals converge to a level smaller than $\mathbf{u}\|A\|\|x\|$ so that there is a deviation of the two residuals. For those cases where b is not given, we choose it such that a given random vector is the solution. We note that for some matrices, it may take $10n$ iterations to achieve that, which is not practical. However, we have included these results in order to show that even with excessive numbers of iterations, we still arrive at small true residuals eventually. We list the normalized residuals $res = \|b - Ax_n\|/(\|A\|\|x_n\|)$ attained by the three implementations and by Gaussian elimination with partial pivoting (MATLAB $A\backslash b$). We also list the number of residual

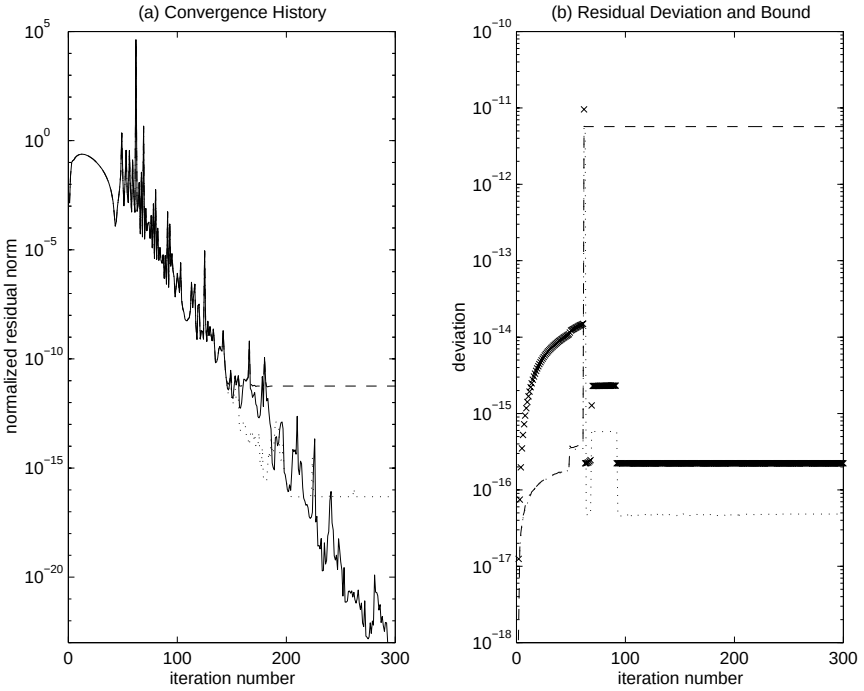


FIG. 3. *Example* 2. *CGS:* (a) *solid—computed residual of CGS; dash—true residual of CGS; dotted—true residual of reliable CGS;* (b) *dash—*$\|b - Ax_n - r_n\|$ *of CGS; dotted—*$\|b - Ax_n - r_n\|$ *of reliable CGS;* x*—bound* d_n *for reliable CGS.*

replacements (n_r) for our reliable implementations and the number of flying-restart (n_f) and the number of residual replacements (n_r) for the implementations of Sleijpen and van der Vorst (SvdV). There are two cases for which the computed residuals do not converge to $O(\mathbf{u})$ with the choice of $\epsilon = 1e-8$. For those cases, a slightly smaller ϵ will recover the stability and the results are listed in the last row of the table.

We see that in all cases, the reliable implementation reduces the normalized residual to $O(\mathbf{u})$ and $res2$ is the smallest among the three implementations, even smaller than MATLAB $A\backslash b$. The improvement on the true residual is more apparent in CGS than in BiCG (or CG). Except in a few cases, both the reliable implementation presented here and the implementation of Sleijpen and van der Vorst work well and are comparable. So the main advantage of the new approach is its simplicity and an occasional improvement in accuracy.

6. Concluding remarks. We have presented a new residual replacement scheme for improving the convergence of the true residuals in finite precision implementations of Krylov subspace iterative methods. By carefully monitoring the deviation of the computed residual and the true residual and incorporating the earlier ideas on residual replacement, we obtain a reliable implementation that preserves the convergence mechanism of the computed residuals, as well as sufficiently small deviations. An error analysis shows that this approach works under certain conditions, and numerical tests demonstrate its effectiveness. Comparison with an earlier approach shows that the new scheme is simpler and easier to implement as an add-on to existing implementations for iterative methods.

TABLE 1

Example 3. *Comparison of normalized residuals: res*0—$A\backslash b$; *res*1—*original implementation; res*2—*reliable implementation; res*3—*implementation of SvdV.*

	$A\backslash b$	BiCG (or CG)			CGS		
Matrix	*res*0	*res*1	*res*2, n_r	*res*3, n_f (n_r)	*res*1	*res*2, n_r	*res*3, n_f (n_r)
bcspwr06	NaN[1]	7e-15	1e-20, 19	1e-19, 5(9)	6e-13	2e-17, 14	3e-17, 32(48)
bcspwr07	NaN[1]	1e-15	2e-17, 46	9e-17, 2(6)	2e-12	2e-17, 20	1e-7, 220(404)
bcspwr08	NaN[1]	2e-15	3e-17, 9	1e-16, 5(7)	7e-14	2e-17, 14	4e-16, 79(103)
bcspwr09	NaN[1]	3e-15	2e-20, 42	6e-20, 5(6)	3e-13	2e-17, 13	4e-16, 40(70)
jpwh991	1e-16	9e-17	3e-17, 1	3e-17, 1(1)	7e-17	3e-17, 1	3e-17, 1(1)
fs6801	1e-17	7e-17	1e-17, 2	8e-18, 1(1)	2e-16	9e-18, 2	1e-17, 3(5)
fs6802	8e-18	1e-16	8e-18, 3	2e-17, 1(1)	4e-16	8e-18, 6	2e-17, 4(4)
fs6803	6e-18	3e-16	1e-13[2] 11	8e-16, 4(5)	4e-14	6e-17, 33	1e-17, 3(5)
fs7601	7e-18	7e-17	9e-18, 1	7e-18, 1(1)	5e-15	5e-18, 2	6e-18, 1(2)
jagmesh1	3e-16	4e-15	5e-17, 2	1e-17, 3(5)	1e-12	5e-17, 5	5e-15, 20(26)
nos3	1e-16	3e-16	6e-17, 2	7e-17, 1(1)	2e-16	6e-17, 2	7e-17, 1(1)
nos4	8e-17	2e-16	5e-17, 1	6e-17, 1(1)	2e-16	5e-17, 1	8e-17, 1(1)
nos5	1e-16	3e-16	5e-17, 2	6e-17, 1(1)	3e-16	6e-17, 2	7e-17, 1(1)
nos6	6e-17	4e-16	3e-17, 9	8e-17, 1(1)	4e-16	2e-17, 14	1e-16, 1(1)
1138bus	9e-18	2e-16	1e-17, 8	9e-17, 1(1)	7e-10	4e-12[3] 21	2e-10, 17(29)
orsirr1	4e-17	1e-15	1e-17, 2	2e-17, 5(9)	9e-14	1e-17, 6	2e-17, 11(18)
orsirr2	7e-17	2e-16	1e-17, 2	1e-17, 3(4)	5e-14	1e-17, 5	2e-16, 4(7)
orsreg1	2e-16	8e-16	7e-17, 1	4e-16, 1(1)	7e-15	8e-17, 2	6e-16, 3(5)
pores1	3e-17	2e-16	3e-17, 2	4e-17, 2(3)	5e-15	3e-17, 5	9e-17, 2(4)
pores3	3e-17	8e-16	2e-17, 3	3e-16, 4(5)	2e-12	2e-17, 11	5e-17, 16(28)
saylr3	NaN[1]	3e-16	3e-17, 2	6e-17, 1(1)	2e-16	3e-17, 2	7e-17, 1(1)
saylr4	3e-16	1e-15	8e-17, 4	5e-16, 1(1)	2e-15	4e-19, 27	7e-19, 8(15)
sherman1	5e-17	3e-16	3e-17, 2	5e-17, 1(2)	2e-10	3e-17, 3	4e-17, 4(41)
sherman3	6e-19	2e-17	4e-19, 9	1e-18, 23(114)	5e-10	6e-19, 30	1e-16, 62(407)
sherman4	6e-17	2e-16	3e-17, 1	3e-17, 1(4)	2e-12	3e-17, 2	3e-17, 1(11)
sherman5	7e-18	2e-14	3e-18, 2	3e-18, 2(52)	4e-8	3e-18, 17	7e-18, 31(215)
watt1	1e-22	2e-16	4e-23, 15	3e-22, 1(1)	5e-17	1e-22, 2	3e-22, 1(1)
watt2	5e-18	2e-16	4e-18, 26	5e-17, 2(2)	3e-15	3e-19, 125	5e-14, 83(125)

1: zero pivot encountered in $A\backslash b$
2: $res2 = 1e-17$, if $\epsilon = 1e-12$; 3: $res2 = 1e-17$, if $\epsilon = 1e-12$;

We point out that the basis for the present work is the understanding that the convergence behavior (of computed residuals) in finite precision arithmetic is preserved under small perturbations to the recurrence relations. Such a supporting analysis is available for BiCG (and CG) [22], but it is still an empirical observation for most other Krylov subspace methods. It would be interesting to derive a theoretical analysis confirming this phenomenon for those methods as well.

Acknowledgments. We would like to thank Ms. Lorrita McKnight for assistance in carrying out the tests on Harwell–Boeing matrices. We thank both referees for their helpful comments.

REFERENCES

[1] R. E. BANK AND T. F. CHAN, *An analysis of the composite step biconjugate gradient algorithm for solving nonsymmetric systems*, Numer. Math., 6 (1993), pp. 295–319.

[2] R. BARRETT, M. BERRY, T. CHAN, J. DEMMEL, J. DONATO, J. DONGARRA, V. EIJKHOUT, R. POZO, C. ROMINE, AND H. VAN DER VORST, *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*, SIAM, Philadelphia, PA, 1994.

[3] I. S. DUFF, R. G. GRIMES, AND J. G. LEWIS, *Sparse matrix test problems*, ACM Trans. Math. Software, 15 (1989), pp. 1–14.

[4] R. Fletcher, *Conjugate gradient methods for indefinite systems*, in Proceedings of the Dundee Conference on Numerical Analysis, 1975, G. A. Watson, ed., Lecture Notes in Math. 506, Springer-Verlag, Berlin, 1976, pp. 73–89.

[5] R. Freund, *A transpose-free quasi-minimal residual algorithm for non-Hermitian linear systems*, SIAM J. Sci. Comput., 14 (1993), pp. 470–482.

[6] R. Freund, G. Golub, and N. Nachtigal, *Iterative solutions of linear systems*, Acta Numer., 1 (1992), pp. 57–100.

[7] R. W. Freund and N. M. Nachtigal, *QMR: A quasi-minimal residual method for non-Hermitian linear systems*, Numer. Math., 60 (1991), pp. 315–339.

[8] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 3rd ed., The Johns Hopkins University Press, Baltimore, MD, 1996.

[9] G. H. Golub and G. Meurant, *Matrices, moments and quadrature* II; *How to compute the norm of the error in iterative methods,* BIT, 37 (1997), pp. 687–705.

[10] A. Greenbaum, *Behavior of slightly perturbed Lanczos and conjugate-gradient recurrences*, Linear Algebra Appl., 113 (1989), pp. 7–63.

[11] A. Greenbaum, *Estimating the attainable accuracy of recursively computed residual methods*, SIAM J. Matrix Anal. Appl., 18 (1997), pp. 535–551.

[12] M. Gutknecht, *Lanczos-type solvers for nonsymmetric linear systems of equations*, Acta Numer., 6 (1997), pp. 271–397.

[13] M. Hestenes and E. Stiefel, *Methods of conjugate gradients for solving linear systems*, J. Res. NBS, 49 (1952), pp. 409–436.

[14] C. Lanczos, *Solution of systems of linear equations by minimized iterations*, J. Res. Natl. Bur. Stand., 49 (1952), pp. 33–53.

[15] Y. Notay, *On the convergence rate of the conjugate gradients in presence of rounding errors*, Numer. Math., 65 (1993), pp. 301–317.

[16] C. Paige, *Accuracy and effectiveness of the Lanczos algorithm for the symmetric eigenproblem*, Linear Algebra Appl., 34 (1980), pp. 235–258.

[17] Y. Saad, *Iterative Methods for Sparse Linear Systems*, PWS Publishing, Boston, MA, 1996.

[18] G. Sleijpen and D. Fokkema, *BiCGstab(ℓ) for linear equations involving unsymmetric matrices with complex spectrum*, Electron. Trans. Numer. Anal., 1 (1993), pp. 11–32.

[19] G. Sleijpen and H. van der Vorst, *Reliable updated residuals in hybrid Bi-CG methods*, Computing, 56 (1996), pp. 144–163.

[20] G. L. G. Sleijpen, H. A. van der Vorst, and D. R. Fokkema, *BiCGstab(ℓ) and other hybrid Bi-CG methods*, Numer. Algorithms, 7 (1994), pp. 75–109.

[21] P. Sonneveld, *CGS, A fast Lanczos-type solver for nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 10 (1989), pp. 36–52.

[22] C. H. Tong and Q. Ye, *Analysis of the finite precision bi-conjugate gradient algorithm for nonsymmetric linear systems*, Math. Comp., to appear.

[23] H.A. van der Vorst, *The performance of FORTRAN implementations for preconditioned conjugate gradients on vector computers*, Parallel Comput., 3 (1986), pp. 49–58.

[24] H. A. van der Vorst, *Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 13 (1992), pp. 631–644.

SIAM J. SCI. COMPUT.
Vol. 22, No. 3, pp. 853–890
© 2000 Society for Industrial and Applied Mathematics

A PARTICLE-PARTITION OF UNITY METHOD FOR THE SOLUTION OF ELLIPTIC, PARABOLIC, AND HYPERBOLIC PDES*

MICHAEL GRIEBEL† AND MARC ALEXANDER SCHWEITZER†

Abstract. In this paper, we present a meshless discretization technique for instationary convection-diffusion problems. It is based on operator splitting, the method of characteristics, and a generalized partition of unity method. We focus on the discretization process and its quality. The method may be used as an h-version or a p-version. Even for general particle distributions, the convergence behavior of the different versions corresponds to that of the respective version of the finite element method on a uniform grid. We discuss the implementational aspects of the proposed method. Furthermore, we present the results of numerical examples, where we considered instationary convection-diffusion, instationary diffusion, linear advection, and elliptic problems.

Key words. meshless methods, gridless discretizations, particle methods, Galerkin methods, partition of unity methods, Lagrange multipliers, h-version, p-version, advection, convection-diffusion

AMS subject classifications. 65M60, 65M25, 65C05, 76N99, 65N30, 65N99

PII. S1064827599355840

1. Introduction.

The discretization of partial differential equations (PDEs) is usually obtained by covering the domain of interest by a suitable grid. Then, after time discretization, linearization, and space discretization (by finite elements, finite differences, or finite volumes), a linear system of discrete equations is set up and solved. Here, beside uniform structured grids which are most simple to handle, block structured grids and unstructured grids also are employed successfully. Furthermore, for dealing with nonsmooth data or solutions, adaptive refinement is a must for efficiency reasons; see [7, 8, 9, 10, 13, 22, 23, 39, 64].

However, all grid- or mesh-based methods are quite involved when it comes to time-dependent problems with complicated geometries, free moving boundaries, and interfaces. Then the geometry of the domain may change over time, the nonsmooth part of the data or the nonsmooth part of the solution changes with time, or the location of singularities of the solution may vary time dependently. Examples for such problems are crack propagation in elasticity theory [12], free surface flow or multiphase flow in fluid dynamics [65], or accretion disks, jets, and cloud simulations in magnetohydrodynamics [20].

In the Eulerian approach, an adaptive mesh technique (h-version, hp-version) must follow the time-dependent features of the data or solution by local refinement and coarsening of the mesh [7, 22, 39, 46, 64]. But time-dependent adaptive mesh refinement and coarsening is not simple, especially for three-dimensional (3D) problems. It is quite involved, programming is complicated, data structures are not easy to handle, and the storage overhead is significant. Besides, good local and global error estimators are necessary. Therefore, there exist only a few unstructured adaptive programs which are able to handle 3D application-oriented problems with time-dependent change of the geometry, the data, or the solution.

*Received by the editors May 12, 1999; accepted for publication (in revised form) March 31, 2000; published electronically August 31, 2000.
http://www.siam.org/journals/sisc/22-3/35584.html

†Sonderforschungsbereich 256 Nonlinear Partial Differential Equations, Project D Meshless numerical methods for the simulation of 3D flows with free boundaries, Institut für Angewandte Mathematik, Universität Bonn, Wegelerstr. 6, D-53115 Bonn, Germany (griebel@iam.uni-bonn.de, schweitz@iam.uni-bonn.de).

The Lagrangian viewpoint allows the mesh itself to be moved (r-method) [6, 53, 54]. But an implementation is still cumbersome, since the mesh may become tangled and twisted, elements may collapse, or angles of some elements may degenerate over time due to the movement of the nodes. The proper treatment of these problems is not an easy task, especially in 3D applications. At least for the two-dimensional (2D) case, the so-called monotone mesh method [65] can cope with these problems to some extent.

Besides, in real life engineering applications, a very time-consuming portion of the overall computation is the mesh generation from CAD input data. Typically, more than 70 percent of the overall computing time is spent by mesh generators.

Hence, especially within the engineering community, there is growing interest in other discretization methods which involve no mesh at all. These approaches are summarized under the term meshless or gridless methods. The main idea is to consider points only, i.e., we omit any fixed relation between the nodes such as element boundaries, and to move just these points in a time-dependent setting. Here, the location of the points and the distribution of the points account for the description of the changing geometry, the change in data, and the time-dependent changing variation of the solution or its gradient.

Generally speaking, there are two different types of meshless approaches—the classical particle methods [57, 58, 60, 61], and gridless discretizations based on data fitting techniques [11, 26]. Traditional particle methods stem from physics applications like Boltzmann equations [1, 33]. They are truly Lagrangian methods, i.e., they are based upon a time-dependent formulation or conservation law. In a particle method we use a discrete set of points to discretize the domain of interest and the solution at a certain time. The PDE is transformed into equations of motion for the discrete set of particles such that the particles can be moved via these equations. After time discretization of the equations of motion we obtain a certain particle distribution for every time step. Therefore, we get an approximate solution to the PDE via the definition of a density function for these particle distributions. These methods are easy to implement. However, they exhibit in general relatively poor convergence properties in weak norms.

The so-called gridless methods follow a different approach. Here, patches or volumina are attached to each point whose union forms an open covering of the domain. Then, local shape functions are constructed with the help of methods from data fitting. These shape functions are used in a Galerkin or collocation discretization process to set up a linear system of equations. Finally this system must be solved efficiently. In contrast to particle methods, such gridless discretizations may also be applied to stationary and elliptic problems. According to the data fitting method involved we can distinguish basically the following three approaches: Shepard's method [69], which has a consistency of first order only, the moving least squares method (MLSM) [44, 45], which generalizes Shepard's approach implicitly to the case of higher order shape functions, and the partition of unity p-version method, which generalizes Shepard's approach explicitly to higher consistency orders. Meanwhile, different realizations of these approaches exist. First, there is the smoothed particle hydrodynamics (SPH) technique of Lucy and Monaghan [31, 32, 51, 55, 56, 71], which resembles (up to area weighted scaling) Shepard's method. Then, Duarte and Oden [26, 27] used in their hp-cloud approach the moving least squares (MLS) idea. Belytschko and coworkers [11, 12] apply similar techniques based on the MLS approach to engineering problems. Furthermore, Dilts [24] used the MLS technique to extend the SPH method to the

so-called MLS particle hydrodynamics (MLSPH) method. Babuška and Melenk [3, 4] proposed the so-called partition of unity method (PUM), which mainly has been applied to uniform point distributions up to now. Liu, Jun, and Zhang [50] proposed variants of the SPH method based on the idea of reproducing kernels of higher order and wavelets. There also exist generalizations of the finite difference approach to the gridless setting [49]. Furthermore, Kansa [40, 41], Franke and Schaback [28, 29], and Wendland [73] used the radial basis approach from approximation theory to construct meshless methods for the discretization of PDEs. The mass-packet method of Yserantant [74, 75] is somewhat different from the classical particle methods. Here, the particles are not considered in the sense of statistical mechanics but they are understood as comparatively big mass-packets, and the conservation of mass is automatically guaranteed by this ansatz. For an overview on meshless methods see [76] and the references therein.

All these data fitting approaches do not depend (at least to a great extent) upon a mesh or any fixed relation between gridpoints (particles). However, the realization and implementation of such a method is not so simple in general: there are often problems with stability and consistency. Furthermore, in a Galerkin method, the discretization of the differential operator, i.e., the integration of the stiffness matrix entries, is in general quite involved in comparison with the conventional grid-based approach. Another challenging task is the discrete formulation of Dirichlet boundary conditions, since the constructed shape functions are in general noninterpolatory. Nevertheless, the different variants of gridless methods are interesting from both the practical and the theoretical point of view. These methods, which are up to now merely in an experimental premature state, possess some potential and might have an interesting future.

The meshless method we propose in this paper is based on the Lagrangian approach for time-dependent problems used within classical particle methods and the ideas of Babuška and Melenk [3, 4], but it allows for a random point distribution. We consider time-dependent convection-diffusion problems, which we decompose into a parabolic and a hyperbolic subproblem using a simple operator splitting. The hyperbolic problem is discretized via a Lagrange-type particle method, i.e., we define equations of motion for the particles corresponding to the problem and move the particles accordingly. We discretize the remaining parabolic subproblem with a two-step Particle–Galerkin method. In the first step we move the particles $\{x_i\}$ via a Monte Carlo step to adapt the particle distribution to the diffusion process. Then, an implicit Eulerian time discretization reduces the parabolic problem to an elliptic problem in every time step. For its discretization in space we employ the PUM approach to randomly distributed points $\{x_i\}$. Here, following ideas from scattered data approximation [21, 69], we set up a partition of unity $\{\varphi_i\}$ over the domain Ω using a localized version of Shepard's method. Since a partition of unity achieves first order consistency only, we need to improve on the approximation quality of the constructed space $V = \operatorname{span}(\{\varphi_i\})$ to allow its use in a collocation or Galerkin method. Thus, we expand the functions φ_i by multiplication with a local polynomial ψ^i of degree p_i, defined on $\operatorname{supp}(\varphi_i)$, and we define the space $V = \operatorname{span}(\{\varphi_i\psi^i\})$. We use these functions as trial and test functions in a Galerkin method. Altogether we obtain an analogue of the h-, p-, and hp-version of the finite element method for general particle distributions. Furthermore, the results of our numerical experiments show that the convergence properties of the proposed method are similar to those of the finite element method.

The remainder of this paper is organized as follows. In section 2, we develop a Lagrangian discretization of the instationary convection-diffusion problem using a simple operator splitting. The resulting hyperbolic subproblem is discretized via a particle method, which is presented in section 3. For the remaining parabolic problem we use a simple time stepping technique. It reduces the parabolic problem to a sequence of elliptic problems. In section 4, we discuss their discretization using a generalized Particle–Galerkin method based on the PUM. Then, in section 5, we present the results of our numerical experiments. Finally, we give some concluding remarks.

2. An operator splitting method for instationary convection-diffusion problems. A widely used approach for the discretization of instationary convection-diffusion problems

$$\begin{array}{rcll} \nabla A \nabla u + v \nabla u & = & u_t & \text{for all } (x,t) \in \Omega \subset \mathbb{R}^d \times (0,T), \\ u(x,0) & = & u_0(x) & \text{for all } x \in \Omega \subset \mathbb{R}^d, \\ Bu(x,t) & = & g(x,t) & \text{for all } (x,t) \in \partial\Omega \times (0,T) \end{array} \tag{2.1}$$

is the operator splitting method [70]. This method can be summarized as follows: Let $S(t)$ be the solution operator for the hyperbolic problem

$$v \nabla u^A = u_t^A, \quad u^A(x,0) = u_0^A(x) \tag{2.2}$$

such that $u^A(x,t) = S(t)u_0^A(x)$ is the entropy solution to (2.2), and let $H(t)$ be the solution operator for the parabolic problem

$$\nabla A \nabla u^D = u_t^D, \quad u^D(x,0) = u_0^D(x), \quad Bu^D(x,t) = g(x,t) \mid_{\partial\Omega} \tag{2.3}$$

such that $u^D(x,t) = H(t)u_0^D(x)$ is the solution to (2.3). A combination of these two continuous solution operators is used as an approximative solution operator for (2.1). This continuous solution operator is evaluated at certain times but is still kept continuous in space, i.e.,

$$u_n(x) = u(x, n\Delta t) \approx [H(\Delta t)S(\Delta t)]^n u_0(x) \tag{2.4}$$

is the semidiscrete approximative solution to (2.1) at time $n\Delta t$. Based on this principle, different variants of the method have been developed; see [37, 38, 48]. The splitting (2.4) inherits its approximation properties from $H(t)$ and $S(t)$, depending on the respective time discretization. In this paper we stick to the most straightforward approach (2.4) for reasons of simplicity.

Now the next step in the discretization process of (2.1) is the choice of approximations to the solution operators $S(t)$ and $H(t)$. The hyperbolic problem (2.2) can be solved analytically with the help of the method of characteristics. Therefore, a natural choice for the discretization of (2.2) is a method which can exploit this knowledge about a continuous solution to (2.2). Such a method would be a Lagrangian particle method, since here a characteristic through a given point x_i^A in space may be interpreted as the path the point will follow over time. This particle method works as follows: First, we sample the initial value u_0 in appropriate points $\{x_i^A\}$; this gives a set of pairs $\{(x_i^A, u_0^i)\}$. Then, we construct the characteristic through every particle x_i^A at time t and move each particle along its corresponding characteristic to its new position at time $t + \Delta t$. Finally, we define an approximate solution $\tilde{u}$ at time $t + \Delta t$

using shape functions that are constructed with respect to the particle distribution. This process is covered in more detail in section 3.

Now, the discretization of the parabolic problem (2.3) should be compatible to the discretization of (2.2). Since we use a Lagrangian particle method for (2.2), we need a discretization method for (2.3) which can handle (almost) random distributions of nodes (particles). Moreover, we want to treat (2.3) also from the Lagrangian point of view. In a pure particle method a parabolic problem like (2.3) is discretized using a Monte Carlo method. Here the shape functions themselves—not only the particles—are moved across the domain without being changed, which causes the poor accuracy of these methods. In our approach we also move the particles. The shape functions, however, are not only moved across the domain. We construct new shape functions in every time step based on the new particle distribution.

We move the particles across the computational domain to simulate the effect of the diffusive transport of (2.3) on the shape functions and approximation space which adapts the distribution of the spatial degrees of freedom only. An approximate solution of (2.3) is then computed via a variational approach with the newly generated approximation space. Therefore the proposed method inherits the advantages of particle methods in adapting the particle distribution via a Monte Carlo step but does not inherit the poor approximation quality of a Monte Carlo method.

Overall we propose the following three-step approximation of (2.3):

1. Time discretization of (2.3) and $u(x,t)$:

$$\begin{aligned} (0,T) &\rightarrow (t_0,\dots,t_n), \quad t_k = k\Delta t, \\ u(x,t) &\rightarrow u(x,t_k) = u^k(x), \\ \nabla A \nabla u^D = u_t^D &\rightarrow -\Delta t \nabla A \nabla u_{m+1}^D + u_{m+1}^D = u_m^D. \end{aligned}$$

2. Use a Monte Carlo step to translate the diffusion equation (2.3) into equations of motion for the particles $\{x_i^A\}$, i.e., use a random walk process

$$x_i^A \rightarrow x_i^A + \sqrt{2\Delta t} A^{\frac{1}{2}} \xi = x_i^D,$$

where ξ are independent random numbers with Gaussian distribution, vanishing mean, and variance one.

3. Now that the particle distribution $\{x_i^D\}$ is suitable for the approximation of the solution of (2.3) in the next time step, we approximate the remaining elliptic problem

$$-\Delta t \nabla A \nabla u_{m+1}^D + u_{m+1}^D = u_m^D \tag{2.5}$$

using a spatial discretization method which allows the use of a random node arrangement. Here, a gridless method seems to be the most natural choice.

In section 4 we present a generalized partition of unity method for elliptic problems, give a short discussion on the implementational difficulties—which are often neglected in other papers, and give some ideas on how to overcome these difficulties.

3. A Lagrangian discretization of hyperbolic problems using a particle method. The main problem of an Eulerian discretization is the following: The initial mesh may be adapted to the initial data but may not be appropriate for the resolution of the solution in future time steps. Therefore we have to use expensive coarsening and refinement strategies—or even complete remeshing—to adapt the mesh to the solution

over time. In contrast to that, a Lagrangian discretization uses the given PDE itself to define a transformation that maps the distribution of the spatial degrees of freedom to appropriate positions over time. Hence, the degrees of freedom follow the solution over time.

We consider the 2D transport problem

$$\frac{\partial u}{\partial t}(p,q,t) + a(p,q,t,u)\nabla_{(p,q)}u(p,q,t) = 0 \tag{3.1}$$

in $\Omega \times (0,T)$ with $\Omega \subset \mathbb{R}^2$ and the general transformation

$$T : (\xi,\eta,\tau) \mapsto (p,q,t) = (p(\xi,\eta,\tau), q(\xi,\eta,\tau), t(\xi,\eta,\tau)).$$

Now we suppose that the function $u(p,q,t)$ is carried into the function $\hat{u}(\xi,\eta,\tau)$ under the transformation T, i.e.,

$$u(p,q,t) = u \circ T(\xi,\eta,\tau) = \hat{u}(\xi,\eta,\tau). \tag{3.2}$$

Using the chain rule, we obtain the τ-derivative of $\hat{u}$

$$\begin{aligned}\frac{\partial \hat{u}(\xi,\eta,\tau)}{\partial \tau} = {} & \frac{\partial u(p,q,t)}{\partial p}\frac{\partial p(\xi,\eta,\tau)}{\partial \tau} + \frac{\partial u(p,q,t)}{\partial q}\frac{\partial q(\xi,\eta,\tau)}{\partial \tau} \\ & + \frac{\partial u(p,q,t)}{\partial t}\frac{\partial t(\xi,\eta,\tau)}{\partial \tau}\,.\end{aligned} \tag{3.3}$$

If we restrict ourselves to transformations T with $t(\xi,\eta,\tau) = \tau$, we have $\partial t/\partial \tau = 1$. Simple reordering of (3.3) gives

$$\frac{\partial u(p,q,t)}{\partial t} = \frac{\partial \hat{u}(\xi,\eta,\tau)}{\partial \tau} - \frac{\partial u(p,q,t)}{\partial p}\frac{\partial p(\xi,\eta,\tau)}{\partial \tau} - \frac{\partial u(p,q,t)}{\partial q}\frac{\partial q(\xi,\eta,\tau)}{\partial \tau}\,.$$

Now we plug this relation into (3.1) and obtain the so-called Lagrangian form [6] of our PDE (3.1)

$$\frac{\partial \hat{u}}{\partial \tau} + \frac{\partial u}{\partial p}\left(a_1 - \frac{\partial p}{\partial \tau}\right) + \frac{\partial u}{\partial q}\left(a_2 - \frac{\partial q}{\partial \tau}\right) = 0\,. \tag{3.4}$$

A solution of (3.4) independent of $\nabla_{(p,q)}u$ may now be obtained from the system of ODEs

$$\frac{\partial p(\xi,\eta,\tau)}{\partial \tau} = a_1(p,q,t,u)\,, \quad \frac{\partial q(\xi,\eta,\tau)}{\partial \tau} = a_2(p,q,t,u)\,, \quad \text{and} \quad \frac{\partial \hat{u}(\xi,\eta,\tau)}{\partial \tau} = 0\,.$$

Note that this is essentially the method of characteristics. The solution $(p,q,\hat{u})$ can now be approximated by a numerical ODE solver like an Euler or Runge–Kutta method, i.e., the transformation T of (3.2) is approximated by a transformation $\tilde{T}$ which is found by numerical integration. The solution to (3.1) can then be approximated by plugging this transformation $\tilde{T}$ into (3.2), which expresses the well-known fact that a solution of (3.1) is constant along characteristics, i.e., on the images of T, giving

$$\tilde{u}(p,q,t) = u \circ \tilde{T}(\xi,\eta,\tau) \simeq \hat{u}(\xi,\eta,\tau). \tag{3.5}$$

We can evaluate the solution at the current time by evaluating the solution at a former time at the corresponding position on the characteristic. Therefore, we use a backward

integration $I^B_{\Delta t}$ to compute $\tilde{T}$. But the particles $\{x_i^t\}$ are propagated forward in time along the approximated characteristic since they have to provide the spatial resolution of the current solution, i.e., a new particle distribution $\{x_i^{(t+\Delta t)}\}$ is defined in every time step. A spatial approximation $\tilde{u}_S(x, t+\Delta t) = \sum_i u_i^{(t+\Delta t)} \phi_i^{(t+\Delta t)}(x)$ to (3.5) is set up with the use of some shape functions $\{\phi_i^{(t+\Delta t)}\}$ which are centered in these new particle positions $\{x_i^{(t+\Delta t)}\}$. Here, the specific choice of the shape functions $\{\phi_i^{(t+\Delta t)}\}$ determines the respective particle method. Since we want to combine the particle method for the hyperbolic subproblem (2.2) with our gridless discretization method for the elliptic subproblem (2.5), we use the shape functions employed there; see section 4. Again, in a pure particle method the shape functions are fixed over time, i.e., they are attached to a certain particle and never changed during the computation. Hence, the current particle position and overall particle distribution are not taken into account. This is one of the reasons for the poor approximation quality of these methods. In our approach we move the particles and construct new shape functions based on the newly generated particle distribution. Therefore, the shape functions are not fixed over time but rather always adapted to the current particle distribution.

We obtain a discrete two-step particle scheme:

1. Time discretization:
   - Treat nonlinearities of a, choose ODE solver(s) $I^{F/B}_{\Delta t}$ and time intervals
   $$(0, T) \to (t_0, \dots, t_n), \quad t_k = k\Delta t.$$
2. Spatial discretization:
   - Propagate particles $\{x_i^{k-1}\}$ forward along their characteristic
   $$x_i^k = I^F_{\Delta t}(a, x_i^{k-1}).$$
   - Project the approximate solution $\tilde{u}_S^{k-1} \circ I^B_{\Delta t}(a, \cdot)$ onto new approximation space
   $$\tilde{u}_S^k(x) = \sum u_i^k \phi_i^k(x) = \tilde{u}_S^{k-1} \circ I^B_{\Delta t}(a, x) = \sum u_i^{k-1} \phi_i^{k-1} \circ I^B_{\Delta t}(a, x).$$

The shape functions $\{\phi_i^k\}$ are constructed in every time step k according to the new particle positions $\{x_i^k\}$ (see section 4), hence the shape functions themselves are propagated along the characteristic. We compute an approximate solution $\tilde{u}_S^k$ by testing a transformed of the solution $\tilde{u}_S^{k-1}$ from the former time step with the new basis functions $\{\phi_i^k\}$, i.e., we have to solve the mass matrix problem for the new shape functions $\{\phi_i^k\}$.

4. Gridless discretization of elliptic problems. In the following, we describe our approach for a gridless discretization of an elliptic problem. The approach is roughly as follows: The discretization is stated in terms of the points only. To obtain test and trial spaces, patches or volumina $\omega_i \subset \mathbb{R}^d$ are attached to each point x_i whose union forms an open cover $\{\omega_i\}$ of the domain Ω, i.e., $\Omega \subset \bigcup \omega_i$. Now, from given weight functions W_i, local shape functions are constructed by Shepard's method. They form a partition of unity $\{\varphi_i\}$. Then, each shape function φ_i is multiplied with a sequence of local polynomials $\{\psi_i^k\}$ to gain higher degree shape functions. These are finally plugged into the weak form to set up a linear system of equations.

4.1. Construction of trial and test space using PUM. Necessary conditions for a trial and test space to perform well in a Galerkin method are local approximability and interelement continuity. Here, local approximability means that

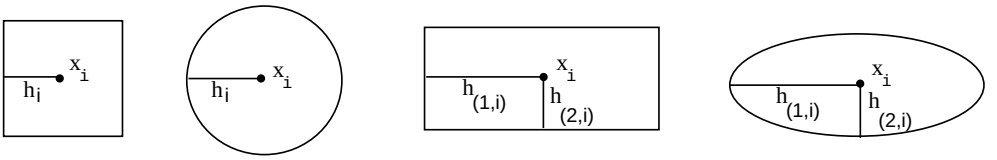


FIGURE 4.1. *Typical examples of the shapes of a patch* ω_i.

the shape functions can approximate the exact solution well locally, and interelement continuity means that any linear combination of shape functions satisfies some global continuity condition. In the finite element method (FEM) we achieve the local approximability via the choice of local polynomials and fulfill the condition of interelement continuity by imposing restrictions onto these polynomials on the element edges. The PUM approach [3, 4] instead focuses on the fulfillment of the condition of interelement continuity via the choice of an appropriate partition of unity (PU) $\{\varphi_i\}$ subordinate to a cover $\{\omega_i\}$. Local expansion of the functions φ_i by the multiplication with local approximation spaces $V_i(\omega_i)$ causes the generated space

$$V := \left\{ \sum_{i=1}^{N} \varphi_i v_i \,|\, v_i \in V_i \right\}$$

to fulfill the condition of local approximability. Theorem 4.3 (see below) states that the global approximation space V inherits the approximation quality of the local spaces V_i. Furthermore, the space V inherits the smoothness of the PU (and the local spaces V_i). Here, the approximation property of the space V may be achieved either by the smallness of the patches (h-version) or by the approximation quality of V_i (p-version).

4.1.1. Open covering of the domain. The starting point for any gridless discretization approach is a collection of N points

$$\{x_i \in \mathbb{R}^d : x_i \in \overline{\Omega}, i = 1, \ldots, N\}. \tag{4.1a}$$

Then, to each point x_i, a patch

$$\omega_i = \{x \in \mathbb{R}^d : ||x_i - x|| \leq h_i\} \subset \mathbb{R}^d \tag{4.1b}$$

is attached. Here, $h_i \in \mathbb{R}$ is the half of the diameter of the patch ω_i. The norm $||\cdot||$ is the Euclidean distance for circles or balls and it is the $||\cdot||_\infty$-norm for quadratic or cube type patches. It is easy to see that this concept can be generalized to patches of more general form: If we allow individual nonuniform sizes $h_i = (h_i^{(1)}, h_i^{(2)}, \ldots, h_i^{(d)})$ in the different coordinate directions and accordingly generalized additive norms $||\cdot||$, we obtain also patches with ellipsoid shape and rectangular or brick type objects.[1] Figure 4.1 shows typical examples of the shapes of a patch ω_i.

The construction of the patches $\{\omega_i\}$ from a given set of grid points $\{x_i\}$ is a first crucial step in the discretization process. Keeping in mind that these patches

[1]We performed experiments using different norms and thus different shapes of the local patches ω_i. It turned out that circles or balls are computationally much more difficult to handle than supports of quadratic or rectangular shape without giving substantial advantage. Therefore, we decided to stick to rectangular shapes in our gridless method.

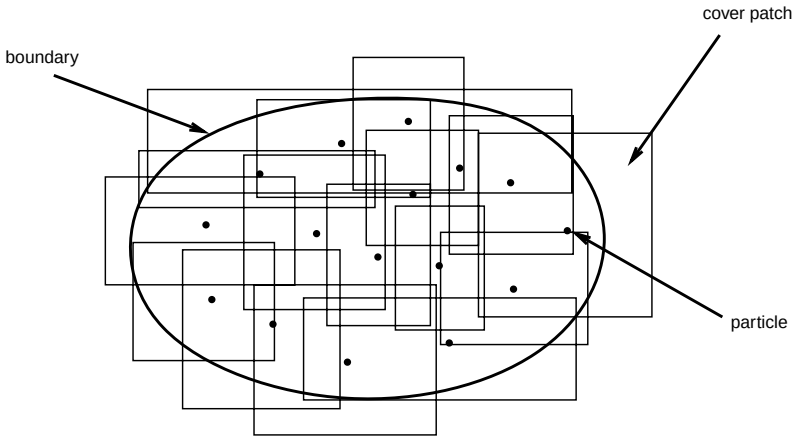


FIGURE 4.2. *Example of an open covering of* Ω.

will be the supports of the trial and test functions in a Galerkin method, the most basic property these patches have to fulfill is that they cover[2] the complete domain $\bigcup_{i=1}^{N} \omega_i \supset \overline{\Omega}$. In other words, for each point $x \in \overline{\Omega}$ there exists at least one patch ω_i which contains x. Figure 4.2 gives an example of an open covering of the domain Ω. A naive approach to the construction of such a cover $\{\omega_i\}$ would be the design of patches $\tilde{\omega_i}$ in such a way that every particle $x_i \in \tilde{\omega_j}$ for some $j \neq i$. But this procedure (in general) does not lead to a cover $\{\tilde{\omega_j}\}$ of the complete domain Ω, i.e., $\bigcup_j \tilde{\omega_j} \not\supset \overline{\Omega}$, since the particles $\{x_i\}$ may not be uniformly distributed in the domain Ω. Therefore, we have to use a more sophisticated algorithm. We use the following variant of an algorithm developed in [27] which resolves the problem mentioned above by using a set P of particles x_i and pseudoparticles ξ_k which we choose to be the nodes of a coarse mesh. The pseudoparticles ξ_k are introduced to guarantee that the patches ω_i completely cover the entire domain Ω.

1. For all $i = 1, \ldots, N$: Set ω_i such that $\operatorname{diam}(\omega_i) = 0$.
2. For all $y \in P$:
   - Evaluate the set $S_{y,R}$ of all particles x_i that fall within a searching square B_R which is centered in y and whose side length is equal to $2R$. If $S_{y,R} = \emptyset$ or $S_{y,R} = y$ in the case of $y = x_i$, increase the size of the searching square, i.e., R, and try again.
   - Compute the distances $h_i^y = \|y - x_i\|$ for all $x_i \in S_{y,R}$ with $x_i \neq y$.
   - Determine the particle $x_j \neq y$ with $h_j^y = \min_i h_i^y$.
   - If $y \notin \omega_j$ increase ω_j appropriately such that $y \in \omega_j$ holds.
3. For all $i = 1, \ldots, N$: Set $\operatorname{diam}(\omega_i) = \alpha \operatorname{diam}(\omega_i)$.

Crucial for the efficient implementation of the above algorithm is the evaluation of the sets $S_{y,R}$. Note that a similar search problem appears in the SPH. There an efficient algorithm for such problems was developed in [71]. Other algorithms for such a cover construction can be derived from tree-algorithms, such as quadtrees or AVL-trees [42]. The overall complexity of the above algorithm is $O(\operatorname{card}(P) \log(N))$.

Since the entries $a_{ij} = a(v_j, v_i)$ of the stiffness matrix A will be nonzero only for functions v_i, v_j with $\operatorname{supp}(v_i) \cap \operatorname{supp}(v_j) \neq \emptyset$, we want to control the number of

[2]Other gridless methods like SPH allow for holes in the domain Ω. Methods based on the MLSM [24, 27] have to impose more severe geometric conditions onto the cover $\{\omega_i\}$.

patches overlapping the same point $x \in \Omega$, i.e., the number of overlapping patches $\{\omega_i\}$ should be minimal. The amount of overlap of the patches $\{\omega_i\}$, however, has a significant effect on the smoothness of our shape functions. Therefore, the overlap $\omega_i \cap \omega_j$ of two patches ω_i and ω_j should be sufficiently large (if it is not empty). This motivates the stretch parameter α in step 3 of the algorithm above which is used to control the interplay between the smoothness of the shape functions and the density of the stiffness matrix.

4.1.2. Weight functions and partition of unity. Now we associate a certain weight function $W_i(x)$ to each subdomain ω_i. Since we decided to use rectangular patches only, i.e., $\omega_i = \bigotimes_{k=1}^{d} \omega_i^{(k)}, \omega_i^{(k)} = \{x^{(k)} \in \mathbb{R}, |x_i^{(k)} - x^{(k)}| \leq h_i^{(k)}\}$, the most natural choice is to use a product approach of one-dimensional local functions, i.e., $W_i(x) = \bigotimes_{k=1}^{d} W_i^{(k)}(x^{(k)})$, where $\operatorname{supp}(W_i) = \overline{\omega_i}$. If we use $[-1,1]$ as reference interval and define the affine linear mapping $T_i^{(k)} : [-1,1] \to \omega_i^{(k)}$, $x^{(k)} \to T_i^{(k)}(x^{(k)})$, we can define the respective weight functions as $W_i^{(k)}(x^{(k)}) = W(T_i^{(k)}(x^{(k)}))$, where the one-dimensional weight function W can be any nonnegative function. Furthermore, we obtain the mapping T_i from the reference element $[-1,1]^d$ to ω_i as the product $T_i : [-1,1]^d \to \omega_i, x \to T_i(x) = \bigotimes_{k=1}^{d} T_i^{(k)}(x^{(k)})$. The weight functions[3] W we use in this paper are the well-known linear, quadratic, and cubic B-splines.

Now, in the next step, we construct shape functions φ_i from these given weight functions W_i with the help of data fitting techniques. In general, a data fitting method [36] produces an approximation $\tilde{u}$ of a function u by

$$\tilde{u}(x) = \sum_{i=1}^{N} u_i \cdot \varphi_i(x),$$

where u_i are given data or are derived from that. Shepard's method uses the idea of inverse distance weighting, which leads to shape functions

$$\varphi_i(x) = \frac{W_i(x)}{\sum_{j=1}^{N} W_j(x)}, \tag{4.2}$$

where $W_i(x) = \|x - x_i\|^{-\beta}$. But since such shape functions have global support, they would lead to a dense stiffness matrix and a quadratic complexity of the method. We therefore use a localized version of Shepard's approach. There are basically two variants. In [45] a locally supported singular weight function such as

$$W_i(x) = L_i(x)\|x - x_i\|^{-\beta}, \quad \text{where } L_i \in \mathcal{C}^\infty \quad \text{and } \operatorname{supp}(L_i) = \omega_i,$$

is used. This approach generates an interpolatory partition of unity, i.e., $\varphi_i(x_j) = \delta_{ij}$. Another approach is to employ a locally supported smooth weight function, e.g., W_i is chosen to be a B-spline[4] [36]. We use the latter approach to circumvent the evaluation of a quotient of singular functions close to their singularity during the numerical integration of the stiffness matrix entries.

[3]Besides these weight functions, the thin-plate splines, Gaussians, and especially the so-called SPH-spline [11, 26, 31, 51] are used in other meshless methods. Note that circular patch shapes would allow for radial functions as weight functions W_i [28, 29, 40, 41, 73].

[4]Note that, together with collocation and area weighting of the resulting shape functions, this gives basically the smoothed particle hydrodynamics method SPH, which was first proposed in [51] and further elaborated in [32, 55, 56, 71].

Since we postulate that the union of the ω_i covers the domain Ω, by (4.2), we are at least able to reproduce constant functions exactly, i.e., the functions (4.2) form a *partition of unity*. Interestingly, this is valid for any choice of weight function, particle distribution, and topology of the cover. Thus, we obtain a consistency order[5] of one in the L^2-norm. The particle distribution $\{x_i\}$ and the cover $\{\omega_i\}$ affect only the computational effort necessary to evaluate the functions φ_i, since the sum in (4.2) has to be taken over all patches ω_j with $\omega_i \cap \omega_j \neq \emptyset$. Furthermore, the amount of overlap of the cover patches and the choice of weight functions W_i affect the smoothness of the Shepard functions φ_i. On the one hand, the partition of unity inherits the smoothness of the weight functions W_i since $\big(W_i \in \mathcal{C}^k(\mathbb{R}^d) \wedge \text{ for all } x \ \sum_i W_i(x) \neq 0\big) \Rightarrow \varphi_i \in \mathcal{C}^k(\mathbb{R}^d)$ holds. On the other hand, the amount of overlap affects the smoothness of φ_i. If the cover is minimal, i.e., there is exactly one patch ω_j for every $x \in \overline{\Omega}$ with $x \in \overline{\omega_j}$, the partition of unity degenerates to the characteristic functions $\varphi_i = \chi_{\omega_i}$ independent of the chosen weight functions W_i. Thus we see that small overlaps will cause very large gradients of φ_i close to the boundary of the respective support ω_i.

First order consistency is in general not sufficient. Hence, some effort is necessary to improve the consistency order. Here, the MLSM [11, 24, 26, 27] allows the construction of shape functions with higher reproduction and consistency order but increases the computational effort dramatically. Furthermore, one has to impose severe geometric restrictions onto the cover [27] to make the method work at all. Therefore we use a different approach. We use the partition of unity to collect local approximation spaces V_i defined on the cover patches ω_i, thus generating a global approximation space V on Ω (see the following section). This space may also reproduce the constant only, but it was shown in [3, 4] that the consistency order of the global space V is nevertheless the same as the consistency order of the local spaces V_i (see section 4.1.4).

4.1.3. Local polynomial expansion. The partition of unity functions φ_i are able to reproduce the constant function. However, for the discretization of a PDE by a Galerkin method, they are not yet sufficient. First, depending on the computed cover $\{\omega_i\}$ and the choice of weight functions W_i, their derivatives can be unbounded and the entries of the stiffness matrix cannot be evaluated in a stable way. We can circumvent these difficulties by the construction of cover patches ω_i with sufficient overlap, but still the consistency error of the discretization would be only of first order in the L^2-norm.

Therefore, we additionally use a hierarchy of local polynomial basis functions of higher degree. To this end, we multiply the partition of unity functions φ_i locally with polynomials. Since we use rectangular patches ω_i only, a local tensor-product space is the most natural choice.

Consider on the reference interval $[-1, 1]$ the Legendre polynomials

$$L_p(x) = \frac{2p+1}{p} x L_{p-1}(x) - \frac{p-1}{p} L_{p-2}(x), \qquad p = 2, 3, \dots, \tag{4.3}$$

with $L_0(x) = 1$ and $L_1(x) = x$. Now, we can define a multidimensional hierarchical

[5]Consistency orders are usually given as exponents of h, i.e., $\|u - \tilde{u}\| = O(h^\alpha)$ with α being the consistency order. However, this formulation is applicable to uniform node arrangements only. Hence, we have to use a different notation in a meshless method. Taking into account that we have $\|x_i - x_j\| = O(N^{-1/d})$ for equidistributed particles $\{x_i\}$ and that $h = N^{-1/d}$ for uniform node arrangements, it seems natural to define the consistency order α of a meshless method as $\alpha := (d \log(\|u - \tilde{u}\|))/\log(N)$.

basis $\{\mathcal{L}_k\}$ with $\mathcal{L}_k(x) = \bigotimes_{j=1}^d L_{k_j}(x^{(j)})$, where $k = (k_1, \ldots, k_d) \in \mathbb{N}^d$ is a multi-index. Then via the affine map T_i from $\omega_i = \bigotimes_{j=1}^d [x_i^{(j)} - h_i^{(j)}, x_i^{(j)} + h_i^{(j)}]$ to $[-1,1]^d$, with $T_i = \bigotimes_{j=1}^d T_i^{(j)}$, where $T_i^{(j)} : [x_i^{(j)} - h_i^{(j)}, x_i^{(j)} + h_i^{(j)}] \to [-1,1]$, we can define a local approximation space V_i on ω_i $V_i = \text{span}\ \{\mathcal{L}_k \circ T_i, \quad k \in I_i\}$. Here $I_i \subset \mathbb{N}_0^d$ is some set of multi-indices $k = (k_1, \ldots, k_d)$, e.g., the full tensor product set $I_i^T = \{k \in \mathbb{N}_0^d : |k|_\infty \le p_i\}$ or the set $I_i^C = \{k \in \mathbb{N}_0^d : |k|_1 \le p_i\}$. The use of the set I_i^C leads to the space V_i^C of complete polynomials of degree p_i, which was also used for the p-version of the finite element method [5].

The global approximation space V is then defined as

$$V = \sum_i \varphi_i V_i \,. \tag{4.4}$$

Note that the local spaces V_i are independent, i.e., there are no compatibility requirements among them. Therefore, not only can the degrees p_i for the sets I_i vary locally, but even different basis functions, e.g., monomials, Taylor polynomials, or harmonic functions [3, 4], might be used on some patches ω_i. In the following, we stick to the Legendre polynomials (4.3). Note that in a general situation with varying sets I_i and bases $\{\psi_k^i\}$, there are some further conditions on the amount of overlap of the cover $\{\omega_i\}$ to ensure that the products $\{\varphi_i \psi_k^i\}$ of the partition of unity functions $\{\varphi_i\}$ and the local basis functions $\{\psi_k^i\}$ are a basis of the global space (4.4); see [4, 68].

4.1.4. Basic convergence theory for PUM. In this section we state the notation and basic theory of the PUM as developed by Babuška and Melenk in [3, 4]. Crucial to the theory of the partition of unity method is the notion of an (M, C_∞, C_∇) partition of unity. The conditions formulated in the following definitions allow us to show the basic approximation properties of the PUM space V of (4.4) as given in Theorem 4.3.

DEFINITION 4.1 (partition of unity). *Let $\Omega \subset \mathbb{R}^d$ be an open set, and let $\{\omega_i\}$ be an open cover of Ω satisfying a point-wise overlap condition*

$$\exists M \in \mathbb{N} \quad \text{for all } x \in \Omega \quad \text{card}\,\{i \,|\, x \in \omega_i\} \le M \,.$$

Let $\{\varphi_i\}$ be a Lipschitz partition of unity subordinate to the cover $\{\omega_i\}$ satisfying

$$\begin{array}{rcl@{\qquad}rcl} \text{supp}(\varphi_i) & \subset & \overline{\omega_i} \quad \text{for all } i\,, & \sum_i \varphi_i & \equiv & 1 \text{ on } \Omega\,, \\ \|\varphi_i\|_{L^\infty(\mathbb{R}^d)} & \le & C_\infty\,, & \|\nabla \varphi_i\|_{L^\infty(\mathbb{R}^d)} & \le & \frac{C_\nabla}{\text{diam}\,(\omega_i)}\,, \end{array}$$

where C_∞ and C_∇ are two constants. Then $\{\varphi_i\}$ is called an (M, C_∞, C_∇) partition of unity subordinate to the cover $\{\omega_i\}$. The partition of unity is said to be of degree $k \in \mathbb{N}_0$ if $\varphi_i \in \mathcal{C}^k(\mathbb{R}^d)$ for all i. The covering sets ω_i are called patches.

DEFINITION 4.2 (PUM space). *Let $\{\omega_i\}$ be an open cover of $\Omega \subset \mathbb{R}^d$ and let $\{\varphi_i\}$ be a (M, C_∞, C_∇) partition of unity subordinate to $\{\omega_i\}$. Let $V_i \subset H^1(\Omega \cap \omega_i)$ be given. Then the space*

$$V := \left\{ \sum_{i=1}^n \varphi_i v_i \,|\, v_i \in V_i \right\}$$

is called a PUM space. The PUM space V is said to be of degree $k \in \mathbb{N}$ if $V \subset \mathcal{C}^k(\Omega)$. The spaces V_i are referred to as the local approximation spaces.

THEOREM 4.3 (approximation property). *Let $\Omega \subset \mathbb{R}^d$ be given. Let $\{\omega_i\}$, $\{\varphi_i\}$, and V_i be as in Definitions* 4.1 *and* 4.2. *Let $u \in H^1(\Omega)$ be the function to be approximated. Assume that the local approximation spaces V_i have the following approximation properties: On each patch $\Omega \cap \omega_i$, the function u can be approximated by a function $v_i \in V_i$ such that $\|u - v_i\|_{L^2(\Omega\cap\omega_i)} \leq \epsilon_1(i)$, and $\|\nabla(u - v_i)\|_{L^2(\Omega\cap\omega_i)} \leq \epsilon_2(i)$ hold. Then the function $u_{\mathrm{ap}} := \sum_i \varphi_i v_i \in V \subset H^1(\Omega)$ satisfies*

$$\begin{aligned} \|u - u_{\mathrm{ap}}\|_{L^2(\Omega)} &\leq \sqrt{M} C_\infty \left(\textstyle\sum_i \epsilon_1^2(i)\right)^{1/2}, \\ \|\nabla(u - u_{\mathrm{ap}})\|_{L^2(\Omega)} &\leq \sqrt{2M} \left(\textstyle\sum_i \left(\frac{C_\nabla}{\operatorname{diam}(\omega_i)}\right)^2 \epsilon_1^2(i) + C_\infty^2 \epsilon_2^2(i)\right)^{1/2}. \end{aligned} \tag{4.5}$$

Proof. See [3, 4].

Due to this theorem we can use the PUM as an h-version, a p-version, or even an hp-version. From (4.5) we can see that the approximation property of the approximation space V may be improved via the reduction of the diameters of the cover patches $\{\omega_i\}$ (the insertion of particles $\{x_i\}$) or via the enhancement of the approximation qualities of the local spaces V_i (the increment of the polynomial order p_i). These refinement strategies are independent, thus we may use both at the same time. Assume that we have a local error estimate $\epsilon_1(i) \leq c_i(\operatorname{diam}(\omega_i))^{\nu_i} p_i^{-\mu_i} \|u\|_{L^2(\Omega\cap\omega_i)}$ for some μ_i, ν_i. Then the error estimates (4.5) of Theorem 4.3 take the form

$$\|u - u_{\mathrm{ap}}\|_{L^2(\Omega)} \leq M C_\infty \max_i \left\{ c_i(\operatorname{diam}(\omega_i))^{\nu_i} p_i^{-\mu_i} \right\} \|u\|_{L^2(\Omega)},$$

demonstrating the hp-like behavior of the method.

Since the PUM only employs a scattered data set, an adaptive h-refinement (by the insertion of new particles $\{x_i\}$) does not have to cope with problems caused by the grid-structure like hanging nodes, etc. Furthermore, there are no compatibility restrictions on the local spaces V_i. Thus we may also employ an adaptive p-refinement on the local spaces V_i. Hence the development of an appropriate error estimator (or at least indicator) is of great interest and is a main topic for future research.

Theorem 4.3 gives an error estimate for u_{ap} which is the linear combination of the locally optimal solutions v_i. The Galerkin method, however, constructs a different approximate solution u_{G}. Here, we do not use the local information from V_i only but rather all information from the spaces V_j with $\omega_j \cap \omega_i \neq \emptyset$. Therefore we sometimes may expect an even better convergence than Theorem 4.3 implies.

Note that the partition of unity $\{\varphi_i\}$ with φ_i from (4.2) and the space V in (4.4) fulfill Definitions 4.1 and 4.2. Consequently, we obtain the approximation properties and error estimates as stated in Theorem 4.3 for our approximation space V. In the following we use the space V in a Galerkin discretization of our elliptic subproblem (2.3).

4.2. Galerkin method with the PUM space. We want to solve elliptic boundary value problems of the type

$$\begin{aligned} Lu &= f \quad \text{in } \Omega \subset \mathbb{R}^d, \\ Bu &= g \quad \text{on } \partial\Omega, \end{aligned} \tag{4.6}$$

where L is a symmetric partial differential operator of second order and B expresses suitable boundary conditions.

Here, the following two major questions come up: How can we efficiently evaluate the integrals which arise in the Galerkin discretization for the stiffness matrix and the right-hand side? And how can we deal with boundary conditions properly?

4.2.1. Construction of appropriate Gauss quadrature formulas. In the following let $a(\cdot,\cdot)$ be the continuous and elliptic bilinear form induced by L on $H^1(\Omega)$. We discretize the partial differential equation using Galerkin's method. Then, we have to compute the stiffness matrix

$$A = (a_{ij}) \quad \text{with } a_{ij} = a(v_j, v_i)$$

and the right-hand side vector

$$\hat{f} = (f_i) \quad \text{with } f_i = \int_\Omega f v_i \,,$$

where $\{v_i\}$ is a basis of the PUM space V.

The partition of unity functions $\{\varphi_i\}$ are (in general) not of the same shape due to the use of a scattered particle set $\{x_i\}$ and the varying overlap of the cover patches $\{\omega_i\}$. Therefore the trial and test functions $\varphi_i\psi_k^i$ also are not of the same shape and we cannot simply use a transformation onto a reference element to compute the integrals (as is done in the FEM). Furthermore, we cannot employ a straightforward integration scheme to compute the integrals because of the low order global continuity of the trial and test functions $\varphi_i\psi_k^i$ and their derivatives.

Assume that the functions $\{\varphi_i\psi_k^i\}$ form a basis of V. If we restrict ourselves for reasons of simplicity to the case $L = -\Delta$ we have to compute the integrals $\int_\Omega \varphi_i\psi_k^i f$ for the right-hand side and the integrals $\int_\Omega \nabla(\varphi_i\psi_k^i)\nabla(\varphi_j\psi_l^j)$ for the stiffness matrix. Recall that φ_i is defined by (4.2). Now we carry out the differentiation. With the notation

$$\mathcal{G}_i := \left(\nabla W_i \sum_m W_m - W_i \sum_m \nabla W_m \right) ,$$

we end up with the integrals

$$\begin{aligned} &\int_\Omega \left(\frac{1}{\sum_m W_m}\right)^4 \mathcal{G}_i\psi_k^i \mathcal{G}_j\psi_l^j \,, \quad \int_\Omega \left(\frac{1}{\sum_m W_m}\right)^2 W_i\nabla\psi_k^i W_j\nabla\psi_l^j \,, \\ &\int_\Omega \left(\frac{1}{\sum_m W_m}\right)^3 \mathcal{G}_i\psi_k^i W_j\nabla\psi_l^j \,, \quad \int_\Omega \left(\frac{1}{\sum_m W_m}\right)^3 W_i\nabla\psi_k^i \mathcal{G}_j\psi_l^j \end{aligned} \tag{4.7}$$

for the stiffness matrix and the integrals

$$\int_\Omega \frac{1}{\sum_m W_m} W_i\psi_k^i f \tag{4.8}$$

for the right-hand side. Since we use piecewise polynomial weights W_i, the functions $\mathcal{G}_i$ may have quite a number of jumps of significant size within their support ω_i. Therefore, the integrals (4.7) should not be computed by a simple quadrature formula which does not respect these discontinuities.

All of the integrals (4.7) and (4.8) are of the form $\int_\Omega (\sum_m W_m)^{-p} \chi$, which leads to the idea of constructing Gauss quadrature formulas with respect to the weight functions $(\sum_m W_m)^{-p}$. Since the functions χ have support $\Omega \cap \omega_i$ or even $\Omega \cap \omega_i \cap \omega_j$, the construction of global quadrature formulas on Ω with respect to each of these weights is not advisable. However, the construction of Gauss quadratureformulas for

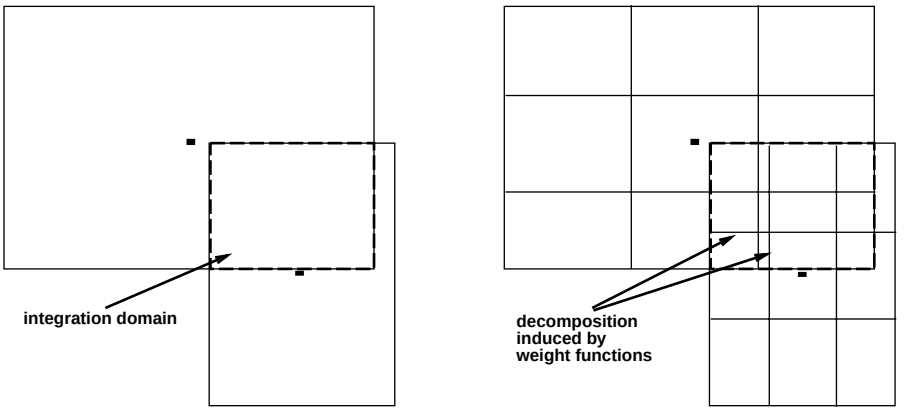


FIGURE 4.3. *Integration domain $\Omega_{ij} = \omega_i \cap \omega_j$ (left). Decomposition $\{C_n\}$ of the integration domain Ω_{ij} via the subdivision induced by the weight functions W_i and W_j (right). Here, the weights are tensor products of quadratic B-splines.*

each integration domain $\Omega \cap \omega_i \cap \omega_j$ and each integration weight is rather expensive. Therefore, we developed a different approach which is an improvement of the integration scheme introduced in [68]. Here, a decomposition of the integration domain induced by the employed shape functions was proposed. More recently a similar yet simpler approach was proposed by Dolbow and Belytschko in [25]. They also make use of the tensorproduct structure of the supports of their shape functions but they construct a *global* decomposition of the domain Ω. In our approach, we decompose each *local* integration domain $\Omega \cap \omega_i \cap \omega_j$ independent of all ω_k with $\omega_k \cap (\Omega \cap \omega_i \cap \omega_j) = \emptyset$. Furthermore, the number of cells of such a local decomposition[6] is almost minimal.

We exploit the tensor product structure of the weight functions W_i and the cover patches ω_i to decompose the integration domain into subpatches with simple geometry on which the functions φ_i are *rational*. Here, we also use the fact that the employed weights W_i are tensor products of B-splines, i.e., they are piecewise polynomial. Consider the integration domain $\Omega_{ij} = \omega_i \cap \omega_j \subset \Omega$. The intersection Ω_{ij} of two cover patches ω_i, ω_j which are tensor products of intervals is also a tensor product of intervals; see Figure 4.3 (left). Furthermore, the employed weight functions W_k are tensor products of B-splines of order l, i.e., they are piecewise polynomials of degree l. Therefore, these weight functions W_k induce a subdivision of each cover patch ω_k into $(l+1) \times (l+1)$ subpatches $\{\omega_{kp}\}$ on which $W_k\big|_{\omega_{kp}}$ is polynomial. Furthermore, these subpatches $\{\omega_{kp}\}$ are also tensor products of intervals. With the help of these subpatches $\{\omega_{ip}\}$, $\{\omega_{jp}\}$ we can define a decomposition $\{C_n\}$ of Ω_{ij}; see Figure 4.3 (right). On the cells C_n of this decomposition we have that $W_i\big|_{C_n}$ and $W_j\big|_{C_n}$ are polynomials of degree l, but all other $W_m\big|_{C_n}$ may still be piecewise polynomial only. Therefore, we refine the decomposition $\{C_n\}$ by further subdividing the cells C_n with the help of the $\{\omega_{kp}\}$ subpatches for all k with $\Omega_{ij} \cap \omega_k \neq \emptyset$; see Figure 4.4 (left). The cells $\hat{C}_n$ of this decomposition may be L-shaped; see Figure 4.4 (left). Such an L-shaped cell $\hat{C}_n$ is further subdivided into two rectangular cells $\tilde{C}_n^1$, $\tilde{C}_n^2$ (see Figure 4.4 (right)) to allow the use of a tensor product quadrature scheme on each cell of

[6]The decomposition is closely connected to the cover construction. The decomposition may be computed during the construction of the cover with only a slight increase of computational work.

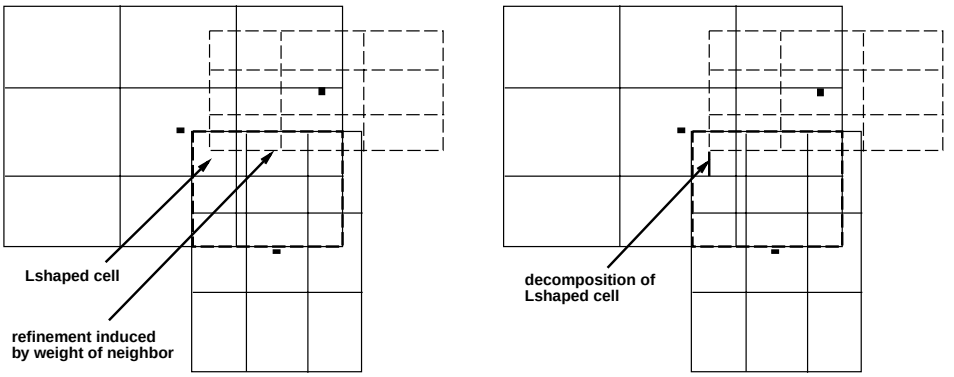


FIGURE 4.4. *Refinement of the decomposition* $\{C_n\}$ *of the integration domain* Ω_{ij} *via the subdivision induced by the weight function* W_k *(tensor product of quadratic B-splines) of one neighboring particle* x_k *(left). Further decomposition of L-shaped cells of this refinement step (right).*

the decomposition. The resulting decomposition[7] $\{\tilde{C}_n\}$ consists of rectangular cells $\tilde{C}_n$ on which all weight functions $W_m|_{\tilde{C}_n}$ are polynomials of degree l. Therefore, the functions $\mathcal{G}_i$ and the integration weights $(\sum_m W_m)^{-p}$ are rational on these cells. Due to the restrictions imposed in section 4.1.4 on the partition of unity $\{\varphi_i\}$ and on the cover $\{\omega_i\}$, the functions $(\sum_m W_m)^{-p}$ are nonsingular on the cells $\tilde{C}_n$. Hence, we can use any standard numerical integration[8] scheme to compute the integrals on the cells $\tilde{C}_n$.

According to Strang's second lemma we have to integrate all entries of the stiffness matrix sufficiently accurately to maintain the convergence order of the discretization error. The shape functions $\varphi_i\psi_k^i$ with $\psi_k^i \neq 1$ are smoother than the partition of unity function φ_i, i.e., a function $\varphi_i\psi_k^i$ is the sum of a polynomial and a rational function whereas φ_i is a rational function only. In a naive approach we could try to exploit this smoothness to reduce the computational costs of the numerical integration of the integrals involving $\varphi_i\psi_k^i$ with $\psi_k^i \neq 1$ by using a quadrature scheme which has a lower order than the quadrature scheme we used for the integrals involving the function φ_i only. But this might lead to the loss of the wanted convergence rate of the discretization error since we now compute the entries of the stiffness matrix with varying accuracy only.

Therefore, we have to find a different way to reduce the computational costs of the numerical integration which allows for a uniform error bound in k and l, i.e.,

$$\left\| \int_\Omega \nabla(\varphi_i\psi_k^i)\nabla(\varphi_j\psi_l^j) - I_\Omega^h \left(\nabla(\varphi_i\psi_k^i)\nabla(\varphi_j\psi_l^j) \right) \right\| \leq Ch^q \quad \text{for all } k, l\,.$$

In any case, we have to evaluate the functions φ_i and their derivatives $\nabla\varphi_i$ in the quadrature points of a sufficiently high order scheme to compute the integrals of the

[7]Note, that this approach is not restricted to domains Ω which are unions of rectangles but rather applicable to general domains. For integration domains $\omega_i \cap \omega_j \cap \Omega$ with $\omega_i \cap \omega_j \not\subset \Omega$ we apply this construction to the fictitious integration domain $\omega_i \cap \omega_j$. The cells $\tilde{C}_n$ with $\tilde{C}_n \cap \partial\Omega \neq \emptyset$ of this decomposition $\{\tilde{C}_n\}$ are then further subdivided using a polygonal approximation to the boundary $\partial\Omega$ of the computational domain Ω. Hence the final cell decomposition for integrals close to the boundary of general domains consists of rectangles and triangles.

[8]Since we use polynomials as local spaces V_i, we may even compute the integrals analytically, i.e., by symbolic computation, if we consider, for example, the Laplacian or more general operators involving polynomial coefficient functions [68].

nonsmooth functions φ_i. Furthermore, it is quite clear that an evaluation of φ_i and $\nabla\varphi_i$ is far more expensive than an evaluation of a local basis function ψ_k^i and its derivatives $\nabla\psi_k^i$. Taking this and the product structure of the approximation space into account we can reduce the computational costs by using the already computed values of the functions φ_i, φ_j and their derivatives for the evaluation of the integrals

$$\int_\Omega \nabla(\varphi_i\psi_k^i)\nabla(\varphi_j\psi_l^j) \quad \text{for all } k,l\,. \tag{4.9}$$

We carry out the differentiation in (4.9) and use the product structure of the shape functions $\{\varphi_i\psi_k^i\}$ to obtain the following integrals

$$\int_\Omega \nabla\varphi_i\nabla\varphi_j\psi_k^i\psi_l^j + \nabla\varphi_i\varphi_j\psi_k^i\nabla\psi_l^j + \varphi_i\nabla\varphi_j\nabla\psi_k^i\psi_l^j + \varphi_i\varphi_j\nabla\psi_k^i\nabla\psi_l^j \tag{4.10}$$

for all k, l. Now we have to integrate products of partition of unity functions φ_i, φ_j, their gradients $\nabla\varphi_i$, $\nabla\varphi_j$, local basis functions ψ_k^i, ψ_l^j and gradients $\nabla\psi_k^i$, $\nabla\psi_l^j$ of local basis functions. Therefore, we can reuse the computed values of φ_i, φ_j and their gradients $\nabla\varphi_i$, $\nabla\varphi_j$ to compute the integrals (4.10) for all k, l. Here, we evaluate the partition of unity functions φ_i and its gradient $\nabla\varphi_i$ only once per quadrature point for the complete local approximation space V_i, consequently reducing the costs but still computing all integrals with the same order of error. Hence, we have to consider the maximal polynomial degrees p_i, p_j of the local functions $\{\psi_k^i\}$, $\{\psi_l^j\}$ and the regularity of the functions φ_i, φ_j for the selection of the order q of the numerical integration scheme on the cells $\tilde{C}_n$.

Note that a further reduction of the costs of the integration can be achieved by using so-called sparse grid integration schemes [30, 63] instead of tensor product formulas on the subpatches.

4.2.2. Boundary conditions and Lagrangian multipliers. Our PUM shape functions $\{\varphi_i\psi_k^i\}$ are noninterpolatory since the partition of unity functions $\{\varphi_i\}$ are (in general) noninterpolatory, i.e., $\varphi_i(x_j) \neq \delta_{ij}$. Furthermore, the usage of local approximation spaces V_i with $\dim(V_i) > 1$ generates an approximation space $V = \sum_i \varphi_i V_i$ with more degrees of freedom than interpolation nodes $\{x_i\}$. Thus, we have to cope with the problem of how to fulfill boundary conditions.

First consider (4.6) with Neumann boundary conditions $Bu = \partial u/\partial n_L = g$ on $\Gamma := \partial\Omega$. We learn from the variational formulation

$$F(v) = \frac{1}{2}a(v,v) - (f,v) - \int_\Gamma gv d\Gamma \to \min\{v \in H^1(\Omega)\} \tag{4.11}$$

that the trial functions v only have to be from the definition space $H^1(\Omega)$ of the differential operator L in its weak form. The functions v do not have to fulfill any additional condition such as boundary values. Thus, the basis of a finite-dimensional subspace V of $H^1(\Omega)$ used to approximate the solution of (4.11) may be compiled of any function $v \in H^1(\Omega)$ and does not need to be interpolatory. Hence, we may directly use our functions $\varphi_i\psi_k^i$ as trial and test functions in the Galerkin procedure without problems.

However, Dirichlet boundary conditions $u = g$ on Γ explicitly impose the values of the solution u on the boundary Γ. Thus, the trial space of the usual weak formulation

$$\text{Find } u \in H^1(\Omega) \cap \{u = g \text{ on } \Gamma\} : a(u,v) = (f,v) \text{ for all } v \in H_0^1(\Omega) \tag{4.12}$$

is not the complete space $H^1(\Omega)$. Therefore, any finite-dimensional space V used to approximate the solution of (4.12) has to consist of functions $\tilde{u}$ fulfilling the boundary condition $\tilde{u} = g$ on Γ. This is usually achieved in the FEM by interpolation of the boundary value $\tilde{u} = \tilde{g}$. Since our shape functions are noninterpolatory, this approach cannot be pursued.

For Dirichlet problems, we use a result from the theory of Lagrange multipliers which states that a solution of (4.12) creates a stationary point of

$$F(v,q) = \frac{1}{2}a(v,v) - (f,v) - \int_\Gamma q(v-g)d\Gamma\,. \tag{4.13}$$

In this formulation the functions v do not have to satisfy the boundary conditions $v = g$ on Γ explicitly, since the additional term enforces the boundary conditions. The respective weak form of (4.13) is as follows: Find $(u,q) \in H^1(\Omega) \times H^{-1/2}(\Gamma)$ with

$$\begin{array}{rcrcl} a(u,v) & + & (\gamma(v),p) & = & (f,v) \text{ for all } v \in H^1(\Omega), \\ (\gamma(u),q) & & & = & (g,q) \text{ for all } q \in H^{-1/2}(\Gamma), \end{array} \tag{4.14}$$

where $\gamma : H^1(\Omega) \to H^{1/2}(\Gamma)$ is the trace operator. We can use the shape functions $\{\varphi_i\psi_k^i\}$ from the construction presented in section 4.1 for the discretization of the *saddle point problem* (4.14) since it only employs the space $H^1(\Omega)$ as trial and test space instead of the restricted spaces $H^1(\Omega) \cap \{u = g \text{ on } \Gamma\}$ as trial and $H_0^1(\Omega)$ as test space for problem (4.12). Thus, there is no need for interpolatory basis functions. The Lagrange multiplier approach in the finite element context can be found in [2, 15], for applications with wavelets, see [43], and within the fictitious domain and mortar element approach compare [52, 59].

Now we have to construct an appropriate finite dimensional subspace $Q \subset H^{-1/2}$ for the discretization of (4.14). Here, we use the particles $\{x_i \in \overline{\Omega}\}$ from (4.1) to construct a new particle distribution $\{x_j^\Gamma\}$ on the boundary Γ. The particles $\{x_i \in \overline{\Omega}\}$ with $\omega_i \cap \Gamma \neq \emptyset$ are projected in the direction of the outer normal onto the boundary Γ and give a new set of particles $\{x_j^\Gamma\}$, which live on the boundary Γ. Then, following the construction described in the previous sections, we set up the finite dimensional PUM space Q on the boundary using the PUM approach: From the newly generated particles $\{x_j^\Gamma\}$ we construct a cover $\{\omega_j^\Gamma\}$ of the $(d-1)$-dimensional manifold Γ, define a partition of unity using Shepard's approach, and choose polynomials on the patches $\{\omega_j^\Gamma\}$ to define a PUM space Q on the boundary. Note that this procedure has to be done with respect to the interior PUM space V. Necessary for the existence and uniqueness of a solution of the saddle point problem

$$\begin{array}{rcrcll} a(u,v) & + & (\gamma(v),p) & = & (f,v) & \text{for all } v \in V \subset H^1(\Omega)\,, \\ (\gamma(u),q) & & & = & (g,q) & \text{for all } q \in Q \subset H^{-1/2}(\Gamma) \end{array} \tag{4.15}$$

is the fulfillment of the discrete Ladyzhenskaya–Babuška–Brezzi (LBB) condition [19]

$$\inf_{q\in Q, q\neq 0} \sup_{v\in V, v\neq 0} \frac{|(\gamma(v),q)|}{\|v\|_V \|q\|_Q} \geq c_0\,. \tag{4.16}$$

According to the results in [15] the boundary space Q has to have a resolution coarse enough compared with the resolution of V in the interior to fulfill the discrete LBB condition (4.16). This coarser resolution of the space Q may be achieved by properly[9]

[9]We propose the use of a particle distribution with fewer particles but approximately the same density distribution as the complete particle distribution $\{x_j^\Gamma\}$.

thinning out the particle distribution, i.e., using only an appropriate subset of $\{x_j^\Gamma\}$, or by the choice of lower order polynomials on the boundary than in the interior. Both of these approaches have shown good results in our numerical experiments; see [68].

4.3. Solution of the discrete saddle point problem. Now we have to solve a linear system with saddle point structure, i.e.,

$$\begin{pmatrix} A & B^t \\ B & 0 \end{pmatrix} \begin{pmatrix} u \\ q \end{pmatrix} = \begin{pmatrix} f \\ g \end{pmatrix}, \tag{4.17}$$

where $B \in \mathbb{R}^{\dim(Q)} \times \mathbb{R}^{\dim(V)}$, $q \in \mathbb{R}^{\dim(Q)}$, $u \in \mathbb{R}^{\dim(V)}$ and $A \in \mathbb{R}^{\dim(V)} \times \mathbb{R}^{\dim(V)}$ is symmetric and positive definite and thus invertible. Block Gaussian elimination directly leads to the Schur complement system

$$BA^{-1}B^T q = BA^{-1}f - g. \tag{4.18}$$

The unique solvability of (4.17) is given when (4.16) holds.

A possible approach for the solution of (4.17) is now to iteratively solve (4.18) by a (preconditioned) conjugate gradient method and then to obtain u by backward substitution, i.e., $u = A^{-1}(f - B^T q)$. A realization of this method with a linear iteration is the (preconditioned) Uzawa algorithm; see [14, 47]. It reads

$$\begin{aligned} u_{i+1} &= u_i + A^{-1}(f - (Au_i + B^T q_i)), \\ q_{i+1} &= q_i + \omega_B C_B (Bu_{i+1} - g), \end{aligned} \tag{4.19}$$

where ω_B is an iteration parameter and C_B is a preconditioner for the Schur complement $BA^{-1}B^T$. The drawback of this method is that the action of A^{-1} must be computed in each iteration step, which makes the method expensive.

If we now replace A^{-1} by some steps of an iterative method, most preferable a multigrid method or a multilevel preconditioner, we employ only an approximation C_A to A^{-1} and obtain the so-called inexact Uzawa algorithm [16, 17]

$$\begin{aligned} u_{i+1} &= u_i + \omega_A C_A (f - (Au_i + B^T q_i)), \\ q_{i+1} &= q_i + \omega_B C_B (Bu_{i+1} - g) \end{aligned} \tag{4.20}$$

with iteration parameters ω_A, ω_B. Altogether, in this algorithm only the action of the matrices A, B, B^T and preconditioners C_A, C_B onto vectors must be programmed.

Here, we can exploit the block-structure of the matrices A and B, which is induced by the product structure of the shape functions $\varphi_i \psi_k^i$. The block corresponding to all the φ_i's can be solved efficiently with an algebraic multigrid solver (AMG) [34, 66, 67] and this solver may further be used as a preconditioner on the diagonal blocks corresponding to $\psi_k^i \neq 1$. Altogether, this procedure gives an efficient preconditioner C_A for the inexact Uzawa algorithm.

Now we have to find an efficient iterative solution method or preconditioner C_B for the Schur complement $BA^{-1}B^T$. Currently we use only $C_B = I$; therefore the iterative solution of (4.17) is not yet optimal, i.e., the condition number of the system matrix still depends on $h \simeq N^{-1/d}$. In the future, effective Schur complement preconditioners C_B have to be constructed to make our gridless method competitive. An approach could be the generalization of the methods developed in [18, 43] to matrices from gridless discretizations.

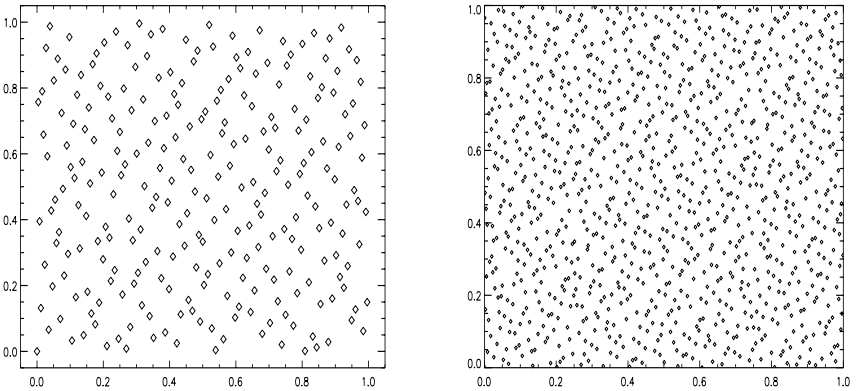


FIGURE 5.1. *Particle distributions with* 256 *points (left) and* 1024 *points (right) generated via a* $(2,3)$ *Halton sequence.*

5. Numerical experiments. In this section we present the results of our numerical experiments. We apply the particle-partition of unity method to a linear advection problem, several elliptic problems, the heat-equation, and an instationary convection-diffusion problem.

5.1. Hyperbolic problem. In the following we consider a standard test case for linear advection problems, the so-called Molenkamp problem [72]:

$$\frac{\partial u}{\partial t} + \begin{pmatrix} -y \\ x \end{pmatrix} \nabla u = 0 \text{ in } \Omega := (-1,1)^2 \times (0,T) \tag{5.1}$$

with an initial value $u(x,y,0) = \exp(-100((x-\frac{1}{4})^2+(y-\frac{1}{4})^2))$. For the exact solution $u(x,y,2l\pi) = u(x,y,0)$ for $l \in \mathbb{N}$ holds and mass $\int_\Omega u$ and energy $\|u\|_{L^2}^2$ are conserved.

We discretize (5.1) using the particle method described in section 3 with the shape functions from the particle-PUM. For the resolution of the initial value we choose a simple block-structure approach. Since $u(x,y,0)$ almost vanishes outside of some $\hat{\Omega} \subset \Omega$, we need only a few particles $\{\xi_i^B\}$ in $\Omega \setminus \hat{\Omega}$ for the construction of the shape functions. We distribute these particles $\{\xi_i^B\}$ in $\Omega \setminus \hat{\Omega}$ using a $(2,3)$ Halton sequence[10] (see Figure 5.1) with about 200 points, and another $(2,3)$ Halton sequence with 256 particles $\{\xi_i^I\}$ is used to resolve the initial value $u(x,y,0)$ in $\hat{\Omega}$ (see Figure 5.2). Note that a further adaptation of the initial particle distribution can be achieved following the ideas of [35].

Some of the particles $\{\xi_i\}$ leave the domain during the computation since we propagate the particles along their corresponding characteristic. To cope with the particle loss, we embed the domain Ω in a larger domain $\tilde{\Omega}$ and distribute a set of background particles $\{\xi_i^B\}$ rather in $\tilde{\Omega}\setminus\hat{\Omega}$ than $\Omega\setminus\hat{\Omega}$. We move all particles $\{\xi_i^B\}\cup\{\xi_i^I\}$ along their characteristic, but a PUM shape function is constructed in those particles $\xi_i \in \{\xi_i^B\} \cup \{\xi_i^I\}$ with $\xi_i \in \Omega$ only. Hence, some particles $\{\xi_i\}$ still leave the domain,

[10]Halton sequences are pseudo Monte Carlo sequences, which are used in sampling and numerical integration [62]. Consider $n \in \mathbb{N}_0$ given as $\sum_j n_j p^j = n$ for some prime p. We can define the transformation H_p from $\mathbb{N}_0$ to $[0,1]$ with $n \mapsto H_p(n) = \sum_j n_j p^{-j-1}$. Then, the (p,q) Halton sequence with N points is defined as $\{(H_p(n), H_q(n))\}$, where $n = 0, \ldots, N-1$.

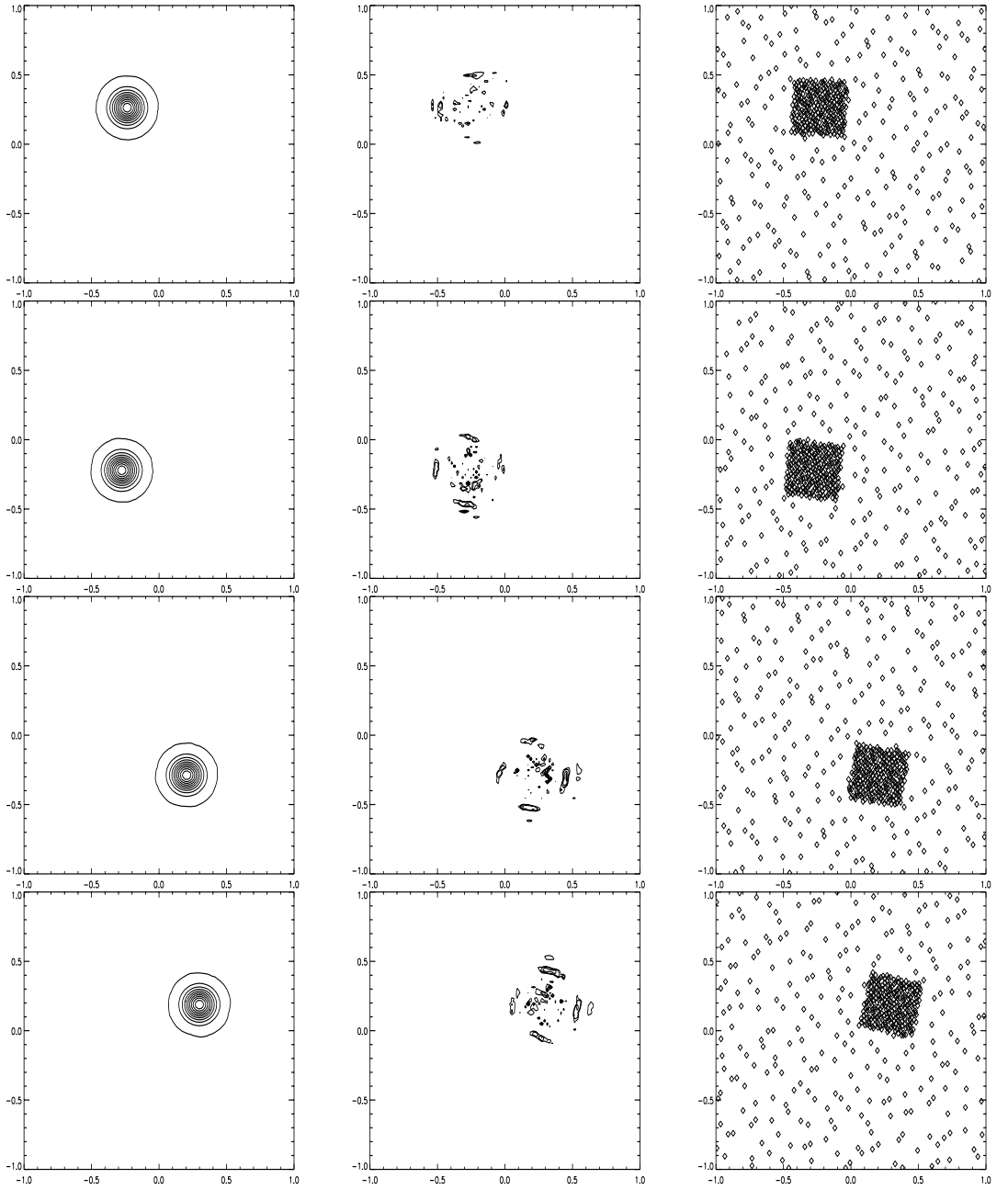


FIGURE 5.2. *Isolines* $(0.005, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9)$ *of the computed solution (left), isolines* $(0.0005, 0.001, 0.002, 0.003, 0.004, 0.005)$ *of the error (middle), and particle distributions (right) for* (5.1) *using the Runge–Kutta method at times* $(6\Delta t, 12\Delta t, 18\Delta t, 24\Delta t)$ *with* $\Delta t = 2\pi/25$.

TABLE 5.1

Errors in the L^∞-norm, oscillations $u_{\mathrm{ap}}^{\mathrm{osc}}$, mass d_{M} and energy d_{E} defects for the Molenkamp problem (5.1) using the Euler method, t and Δt given in $2\pi/50$.

Δt	$\|e\|_{L^\infty}$	$u_{\mathrm{ap}}^{\mathrm{osc}}$	d_{M}	d_{E}	$\|e\|_{L^\infty}$	$u_{\mathrm{ap}}^{\mathrm{osc}}$	d_{M}	d_{E}	$\|e\|_{L^\infty}$	$u_{\mathrm{ap}}^{\mathrm{osc}}$	d_{M}	d_{E}
	$t=1$				$t=2$				$t=4$			
4									4.17_{-1}	1.21_{-3}	2.03_{-1}	2.03_{-1}
2					1.15_{-1}	1.59_{-3}	5.99_{-2}	6.00_{-2}	2.27_{-1}	2.41_{-3}	1.16_{-1}	1.16_{-1}
1	2.93_{-2}	1.23_{-3}	1.57_{-2}	1.57_{-2}	5.89_{-2}	1.40_{-3}	3.11_{-2}	3.11_{-2}	1.17_{-1}	2.06_{-3}	6.13_{-2}	6.13_{-2}
	$t=8$				$t=16$				$t=32$			
8	9.62_{-1}	1.34_{-3}	5.05_{-1}	5.05_{-1}	1.00_{0}	3.44_{-3}	7.55_{-1}	7.55_{-1}	1.00_{0}	-1.66_{-1}	9.40_{-1}	9.43_{-1}
4	7.42_{-1}	6.83_{-4}	3.65_{-1}	3.65_{-1}	9.90_{-1}	1.68_{-3}	5.97_{-1}	5.97_{-1}	1.00_{0}	9.89_{-3}	8.38_{-1}	8.38_{-1}
2	4.37_{-1}	1.23_{-3}	2.19_{-1}	2.19_{-1}	7.69_{-1}	1.00_{-3}	3.90_{-1}	3.90_{-1}	9.94_{-1}	3.00_{-3}	6.28_{-1}	6.28_{-1}
1	2.31_{-1}	1.23_{-3}	1.19_{-1}	1.19_{-1}	4.44_{-1}	9.31_{-4}	2.24_{-1}	2.24_{-1}	7.77_{-1}	2.84_{-3}	3.97_{-1}	3.97_{-1}
	$t=64$				$t=128$				$t=256$			
8	1.00_{0}	-9.40_{-1}	9.96_{-1}	1.00_{0}	1.00_{0}	-1.00_{0}	1.00_{0}	1.00_{0}	1.00_{0}	-1.00_{0}	1.00_{0}	1.00_{0}
4	1.00_{0}	-4.28_{-1}	9.74_{-1}	9.82_{-1}	1.00_{0}	-9.91_{-1}	9.99_{-1}	1.00_{0}	1.00_{0}	-1.00_{0}	1.00_{0}	1.00_{0}
2	1.00_{0}	4.86_{-3}	8.62_{-1}	8.62_{-1}	1.00_{0}	-7.87_{-1}	9.81_{-1}	9.93_{-1}				
1	9.96_{-1}	5.34_{-3}	6.37_{-1}	6.37_{-1}								

but there also are some particles $\{\xi_i^B\}$ (re)entering Ω. Therefore, we have an almost constant number of particles in Ω and $\hat{\Omega}$ over time.

We use a stretch parameter $\alpha = 1.25$, the well-known hat function, and complete quadratic Legendre polynomials V_i^C for the construction of the shape functions in this experiment. Hence, the approximation space has about 2700 degrees of freedom, of which we have 1500 within the approximate support $\hat{\Omega}$. As ODE solver in this experiment we use a simple Euler and a fourth order Runge–Kutta method. The increase of computational work by applying a Runge–Kutta method instead of the Euler method is negligible since we only have more evaluations of the velocity field which is independent of the solution of (5.1).

We solve the mass matrix problem using a preconditioned CG method with a tolerance of 10^{-8}. The error $u - u_{\mathrm{ap}}$ in the L^∞-norm and the oscillations

$$u_{\mathrm{ap}}^{\mathrm{osc}} = \begin{cases} \min u - \min u_{\mathrm{ap}} & : \ |\min u_{\mathrm{ap}} - \min u| > |\max u - \max u_{\mathrm{ap}}|, \\ \max u_{\mathrm{ap}} - \max u & : \ |\min u_{\mathrm{ap}} - \min u| \le |\max u - \max u_{\mathrm{ap}}| \end{cases}$$

(both computed on a 500×500 grid), the mass defect $d_{\mathrm{M}} = 1 - \int_\Omega u_{\mathrm{ap}} / \int_\Omega u$, and the energy defect $d_{\mathrm{E}} = 1 - \|u_{\mathrm{ap}}\|_{L^2}^2 / \|u\|_{L^2}^2$ for the computed solution u_{ap} at various times t for several values of Δt are given in Table 5.1 for the Euler method. The corresponding values for the Runge–Kutta method are displayed in Table 5.2. From these numbers we see the anticipated convergence behavior in the L^∞-norm, i.e., $O(\Delta t)$ for the Euler and $O(\Delta t^4)$ for the Runge–Kutta method.

We compute an approximate solution u_{ap} in every time step k by projecting a transformed former solution onto the current approximation space,

$$u_{\mathrm{ap}}^k = \Pi^k \left(u_{\mathrm{ap}}^{k-1} \circ \tilde{T}^{\Delta t} \right). \tag{5.2}$$

Hence, for the defects in mass d_{M} and energy d_{E} we get the equivalences

$$d_{\mathrm{M}}^k = 1 - \frac{\int_\Omega \Pi^k \left(u_{\mathrm{ap}}^{k-1} \circ \tilde{T}^{\Delta t} \right)}{\int_\Omega u^0}, \qquad d_{\mathrm{E}}^k = 1 - \frac{\int_\Omega \| \Pi^k \left(u_{\mathrm{ap}}^{k-1} \circ \tilde{T}^{\Delta t} \right) \|^2}{\int_\Omega \|u^0\|^2}, \tag{5.3}$$

where $u(\cdot, k\Delta t) = u^k = u^0 \circ T^{k\Delta t}$ and $T^{k\Delta t}$ is the transformation induced by the exact characteristic at $k\Delta t$ with $|\det J_{T^{k\Delta t}}| = 1$. Not only the transformation $\tilde{T}^{\Delta t}$

TABLE 5.2

Errors in the L^∞-norm, oscillations $u_{\mathrm{ap}}^{\mathrm{osc}}$, mass d_{M}, and energy d_{E} defects for the Molenkamp problem (5.1) using a fourth order Runge–Kutta method, t and Δt given in $2\pi/50$.

Δt	$\|e\|_{L^\infty}$	$u_{\mathrm{ap}}^{\mathrm{osc}}$	d_{M}	d_{E}	$\|e\|_{L^\infty}$	$u_{\mathrm{ap}}^{\mathrm{osc}}$	d_{M}	d_{E}	$\|e\|_{L^\infty}$	$u_{\mathrm{ap}}^{\mathrm{osc}}$	d_{M}	d_{E}
	$t=1$				$t=2$				$t=4$			
4									5.14_{-3}	1.72_{-3}	-2.23_{-4}	-2.20_{-4}
2					4.62_{-3}	1.61_{-3}	-3.91_{-6}	-5.88_{-7}	4.31_{-3}	1.84_{-3}	-9.12_{-6}	9.98_{-7}
1	4.40_{-3}	1.41_{-3}	5.56_{-7}	4.27_{-6}	5.25_{-3}	1.57_{-3}	2.30_{-7}	7.57_{-6}	5.05_{-3}	1.68_{-3}	-2.40_{-7}	1.58_{-5}
	$t=8$				$t=16$				$t=32$			
8	3.16_{-2}	7.62_{-4}	-1.30_{-2}	-1.30_{-2}	6.18_{-2}	1.44_{-3}	-2.62_{-2}	-2.62_{-2}	1.21_{-1}	1.97_{-3}	-5.32_{-2}	-5.32_{-2}
4	4.30_{-3}	1.70_{-3}	-4.48_{-4}	-4.40_{-4}	8.13_{-3}	1.97_{-3}	-8.94_{-4}	-8.77_{-4}	8.48_{-3}	1.69_{-3}	-1.79_{-3}	-1.75_{-3}
2	4.08_{-3}	1.41_{-3}	-1.51_{-5}	1.32_{-6}	4.60_{-3}	1.73_{-3}	-3.05_{-5}	5.67_{-6}	4.72_{-3}	1.20_{-3}	-5.65_{-5}	7.97_{-6}
1	3.80_{-3}	1.05_{-3}	4.40_{-7}	2.72_{-5}	6.20_{-3}	1.36_{-3}	1.40_{-6}	5.61_{-5}	6.96_{-3}	1.84_{-3}	2.65_{-6}	1.03_{-4}
	$t=64$				$t=128$				$t=256$			
8	2.40_{-1}	1.02_{-3}	-1.09_{-1}	-1.09_{-1}	4.63_{-1}	4.93_{-4}	-2.30_{-1}	-2.30_{-1}	7.99_{-1}	2.63_{-3}	-5.14_{-1}	-5.13_{-1}
4	1.53_{-2}	1.14_{-3}	-3.58_{-3}	-3.51_{-3}	3.05_{-2}	1.99_{-3}	-7.17_{-3}	-7.05_{-3}	5.90_{-2}	2.72_{-3}	-1.44_{-2}	-1.42_{-2}
2	8.49_{-3}	2.45_{-3}	-1.08_{-4}	1.06_{-5}	9.90_{-3}	3.14_{-3}	-2.08_{-4}	-5.46_{-6}				
1	1.02_{-2}	3.29_{-3}	5.13_{-6}	1.90_{-4}								

but also the projection Π^k, i.e., the approximation space, depend on the applied ODE solver since the particles which define the approximation space are propagated forward along the characteristic. Therefore, we have two sources for a time-dependent error component in (5.3), $|\det J_{\tilde{T}^{\Delta t}}|$ and Π^k. A convergence behavior similar to that of the applied ODE solver can be perceived from the measured defects $d_{\mathrm{M/E}}$. As one can expect, for both ODE solvers we have $d_{\mathrm{M}} = d_{\mathrm{E}}$ if the defects are large compared with the spatial resolution, i.e., the displayed defects $d_{\mathrm{M/E}}$ are significant only if the local errors can be resolved (beyond oscillations) by the approximation space. For the Euler method we have a loss of energy and mass over time which is due to the fact that the backward Euler integration contracts the support of the approximate solution, i.e., the characteristics are approximated by a contracting spiral. For the particle distribution, though, we approximate the characteristic by forward integration. Hence, the path of a single particle is a widening spiral and the particles are propagated away from the support of the solution. The particles may even leave the domain of interest whereas the approximate solution will move to the center with a contracted support. Therefore, the approximate solution may vanish once its support is contracted beyond the spatial resolution of the approximation space (see Table 5.1). Hence, we have an additional loss of spatial resolution due to the difference in forward-backward time integration. This error affects the quality of the projection Π^k which is the main source for the oscillations $u_{\mathrm{ap}}^{\mathrm{osc}}$. The displayed oscillations $u_{\mathrm{ap}}^{\mathrm{osc}}$ are also a measure for peak attenuation since $u_{\mathrm{ap}}^{\mathrm{osc}} = \min u - \min u_{\mathrm{ap}}$ in all our experiments.

In Figure 5.2 the isolines of the approximate solution, the isolines of the error, and the corresponding particle distribution at times $(6\Delta t, 12\Delta t, 18\Delta t, 24\Delta t)$ with $\Delta t = 2\pi/25$ are displayed. We clearly see that the spatial degrees of freedom follow the evolution of the solution. From these figures we cannot see a difference in the forward-backward time integration for the Runge–Kutta method, i.e., the particles and the solution are propagated along the "same" path. This can also be observed from the oscillations displayed in Table 5.2 since $u_{\mathrm{ap}}^{\mathrm{osc}}$ is (almost) independent of Δt and t.

From the displayed isolines we notice the "conservation" of the peak of the solution. Moreover, we see that the overall error is dominated by the error in space rather than in time for small values of Δt. Further experiments showed the antici-

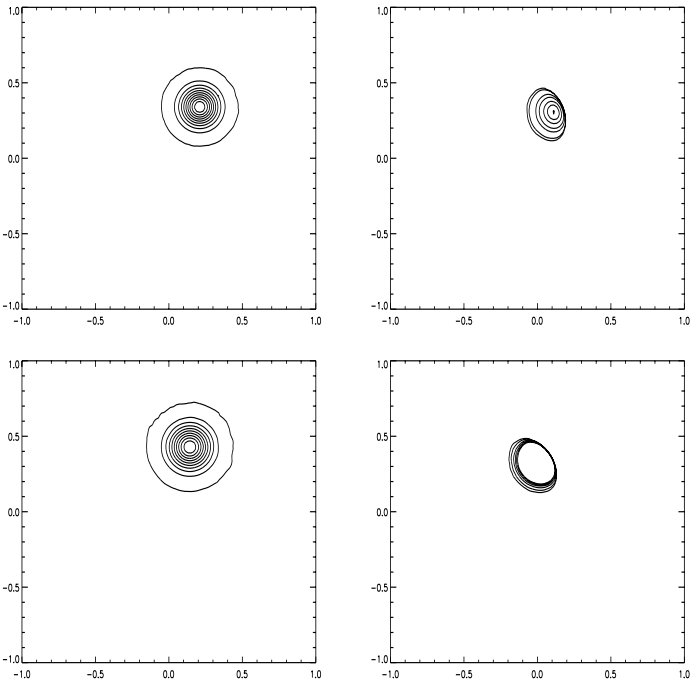


FIGURE 5.3. *Isolines* $(0.005, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9)$ *of the computed solution (left) and isolines* $(0.005, 0.01, 0.05, 0.1, 0.2, 0.3, 0.4)$ *of the error (right) for* (5.1) *using a fourth order Runge–Kutta method after* 3 *(*$t = 19\Delta t$*, top) and* 6 *(*$t = 38\Delta t$*, bottom) revolutions with* $\Delta t = 4\pi/25$.

pated "third order" convergence in space (see section 5.2). Here, the error is located in a small region around the approximate support $\hat{\Omega}$ where the resolution of the approximation space is significantly less than in $\hat{\Omega}$. Hence, the error could probably be reduced by using an initial particle distribution which is more adapted to the initial value. The isolines of the computed solution after (slightly more than) three and six full revolutions using the Runge–Kutta method with $\Delta t = 4\pi/25$ are displayed in Figure 5.3. Again, we see the conservation of the peak. But here we can also observe the mass and energy defect. The diameter of every isoline grows, i.e., we have an increase in mass and energy over time. Furthermore, the isolines of the corresponding errors are displayed which show the dominance of the error in time rather than in space for large values of Δt.

5.2. Elliptic problems. In the following we consider the model problem

$$-\nabla A \nabla u + bu = f \text{ in } \Omega := (0,1)^2 \tag{5.4}$$

with suitable boundary conditions $Bu = g$ on $\Gamma := \partial\Omega$. We discretize this elliptic problem using our particle-PUM as described in section 4. The particle distribution we use is generated via either a Halton sequence or a graded uniform distribution.

According to the results of [3, 4] (see section 4.1.4), we can estimate the global error with respect to $\max_i \operatorname{diam}(\omega_i)$ only. We can expect erratic fluctuations in the convergence rates if we measure the convergence based on such error estimates since we use a pseudo Monte Carlo sequence to generate the particle distribution. Furthermore, we may get a convergence behavior of the global space V which is superior to the one of the local spaces V_i due to the overlapping of the cover patches; see section 4.1.4.

Therefore, we measure convergence rates ρ with respect to the number of degrees of freedom $\mathsf{dof} = \sum_{i=1}^{N} \dim(V_i)$, i.e., we propose an algebraic error estimate

$$\|u - \tilde{u}\| = O(\mathsf{dof}^{\rho}).$$

We compute ρ from the measured errors $\|u - \tilde{u}_l\|$ and $\|u - \tilde{u}_{l-1}\|$ on two successive levels l and $l-1$ via the relation

$$\rho = \frac{\log\left(\frac{\|e_l\|}{\|e_{l-1}\|}\right)}{\log\left(\frac{\mathsf{dof}_l}{\mathsf{dof}_{l-1}}\right)}, \quad \text{where } e_l = u - \tilde{u_l}. \tag{5.5}$$

These convergence rates ρ can be translated into the well-accepted h^{α} notation in the case of a uniform particle distribution and $\dim(V_i) = \mathsf{const}$. Here, we have $h \sim N^{-1/d} \sim \mathsf{dof}^{-1/d}$. Therefore we get the relation $-\alpha/d \sim \rho$. For example, the analogue of a convergence of the order $O(h^2)$ in the L^2-norm is achieved if we measure $\rho_{L^2} = -1$ for a 2D problem.

Example 1 (h-version of particle-PUM, smooth solution). Our first elliptic example is the simple Helmholtz equation

$$-\Delta u + u = f \text{ in } \Omega = (0,1)^2 \tag{5.6}$$

with Neumann boundary conditions $\partial u/\partial n = g$ on $\Gamma = \partial\Omega$. We choose f and g such that the continuous solution to the problem is given as

$$u(x,y) = \arctan\left(100\left(\frac{x+y}{\sqrt{2}} - 0.8\right)(x - x^2)(y - y^2)\right). \tag{5.7}$$

The weight function W_i in this experiment is the quadratic B-spline, the stretch parameter $\alpha = 1.5$, the local approximation spaces V_i are the linear Legendre polynomials, and the underlying particle distribution is the pseudo Monte Carlo distribution generated via a $(2,3)$ Halton sequence. Therefore, we expect to measure convergence rates $\rho_{L^2} \sim -1$ in the L^2-norm and $\rho_{H^1} \sim -0.5$ in the H^1-norm. The solution (5.7) and the error for the experiment with 256 particles are displayed in Figure 5.4.

Table 5.3 and Figure 5.5 give the error in different norms and the associated convergence rates for varying particle numbers. We clearly see a rate of -1 in the L^2-norm and L^∞-norm and a rate of -0.5 for the H^1-norm (i.e., a convergence of the order $O(h^2)$ in the L^2-norm and L^∞-norm and a convergence of the order $O(h)$ in the H^1-norm). These results demonstrate that the particle-PUM based on a pseudo Monte Carlo particle distribution matches the convergence behavior of the h-version of the finite element method on a uniform grid. Thus, our method incorporates the advantages of a particle method but preserves at the same time the usual convergence properties of the finite element method.

Example 2 (h-version of particle-PUM, singular solution). In our second experiment with an elliptic problem we consider the Laplace equation

$$-\Delta u = f \text{ in } \Omega = (0,1) \times (-0.5, 0.5) \tag{5.8}$$

with Dirichlet boundary conditions $u = g$ on $\Gamma = \partial\Omega$. We choose f and g such that

$$u(x,y) = u(z) = \mathrm{Re}(z^{1/2}) = (x^2 + y^2)^{1/4} \cos\left(\frac{\arctan(\frac{x}{y})}{2}\right) \tag{5.9}$$

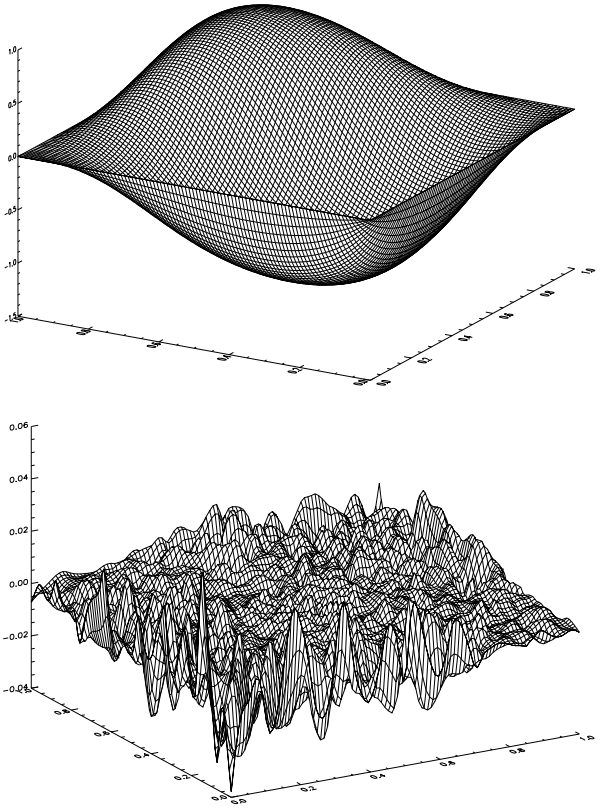


FIGURE 5.4. *Graph of solution* (5.7) *and graph of error for Example* 1 *using* 256 *particles.*

TABLE 5.3
Errors and convergence rates for Example 1.

N	p	dof	$\|e\|_{L^\infty}$	ρ_{L^∞}	$\|e\|_{L^2}$	ρ_{L^2}	$\|e\|_{H^1}$	ρ_{H^1}
64	1	192	1.318_{-1}	$***$	2.819_{-2}	$***$	9.826_{-1}	$***$
256	1	768	4.002_{-2}	-0.860	5.472_{-3}	-1.183	4.056_{-1}	-0.638
1024	1	3072	1.119_{-2}	-0.919	1.629_{-3}	-0.874	2.223_{-1}	-0.434
4096	1	12288	3.004_{-3}	-0.949	3.833_{-4}	-1.044	1.044_{-1}	-0.535
16384	1	49152	7.797_{-4}	-0.973	9.439_{-5}	-1.011	5.230_{-2}	-0.499

is the continuous solution to the problem. Besides a uniform particle distribution we also use a graded particle sequence (see Figure 5.8) to cope with the singularity of the solution (5.9). Note that all particles ξ_i of the uniform distribution are interior points of the domain Ω, i.e., $\xi_i \in \Omega \setminus \partial\Omega$. We use the same weight functions and local approximation spaces as in Example 1 and a stretch parameter $\alpha = 1.25$ for the uniform and the graded particle distribution. The Dirichlet boundary conditions are enforced by Lagrangian multipliers which are constructed according to section 4.2.2. Here, we are using the constant function only as local approximation spaces for the multiplier space Q_h to have a coarser resolution on the boundary than in the interior so that the discrete LBB condition (4.16) holds. We measure a convergence rate ρ_{L^∞} of about -0.35 for the uniform particle distribution (see Table 5.4). This is slightly better than the well-known convergence of the order $O(h^{1/2})$ of the finite element method in the L^∞-norm for this problem. Since the pointwise convergence of

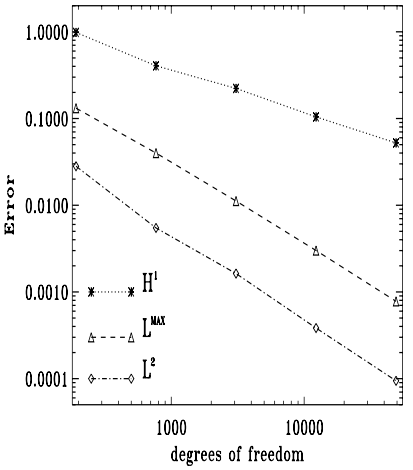


FIGURE 5.5. *Convergence history for Example* 1.

TABLE 5.4
Errors and convergence rates for Example 2 *for the uniform particle distribution (left) and the graded particle distribution (right).*

$\|e\|_{L^\infty}$	ρ_{L^∞}	$\|e\|_{L^2}$	ρ_{L^2}	N	p	dof	$\|e\|_{L^\infty}$	ρ_{L^∞}	$\|e\|_{L^2}$	ρ_{L^2}
7.639_{-2}	$***$	1.889_{-3}	$***$	64	1	192	8.059_{-2}	$***$	3.196_{-3}	$***$
5.251_{-2}	-0.270	7.171_{-4}	-0.699	256	1	768	5.293_{-2}	-0.303	9.593_{-4}	-0.868
3.391_{-2}	-0.315	2.653_{-4}	-0.717	1024	1	3072	3.001_{-2}	-0.409	2.860_{-4}	-0.873
1.875_{-2}	-0.427	9.727_{-5}	-0.724	4096	1	12288	1.473_{-3}	-0.513	9.603_{-5}	-0.787

the finite element method in points sufficiently far away from the singularity is of the order $O(h^{3/2})$, we can expect a convergence rate ρ_{L^2} in the L^2-norm less than -0.75. We measure a convergence rate ρ_{L^2} which is close to -0.75 for the uniform particle distribution. Figure 5.6 shows the solution (5.9) and the error for 256 particles of the graded particle distribution. From Figure 5.7 and Table 5.4 we see that the use of a graded particle distribution improves the convergence behavior of the method.

Now, we measure convergence rates ρ_{L^2} and ρ_{L^∞} of about -0.8 in the L^2-norm and -0.5 in the L^∞-norm. Since the grading we employ to the particle distribution (see Figure 5.8) is not optimal, we are not able to completely recover $\rho_{L^2,L^\infty} \sim -1$. Here, error estimators for adaptive particle refinement must be developed in the future to fully exploit the capabilities of the method.

The results from Examples 1 and 2 demonstrate that the particle-PUM does work on irregular or scattered data with roughly the same convergence behavior as the finite element method on a uniform grid. Furthermore, Example 2 indicates that we might also resolve singularities of the solution by local refinement, i.e., by increasing the particle density close to the singularity. Thus, we can achieve a similar convergence as we would obtain with an adaptive finite element method but can circumvent the geometric constraints we have to face there (angle conditions, hanging nodes, proper local grid refinement, quasi-uniformity of the grid, etc.). With the particle-PUM we may insert particles (almost) anywhere near the singularity without worrying much about the neighboring particles and their respective patch size.

Example 3 (p-version of particle-PUM, smooth solution). We again consider

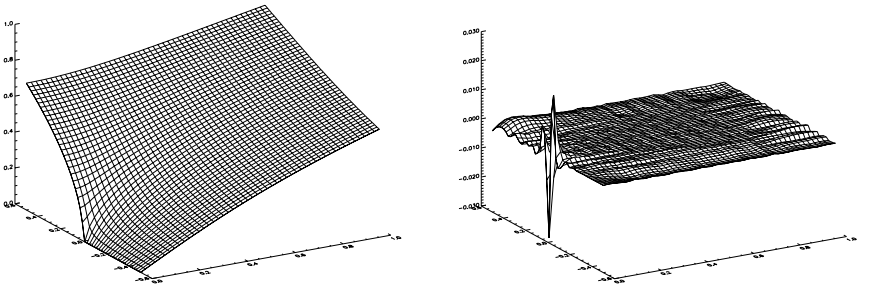


FIGURE 5.6. *Graph of solution* (5.9) *and graph of error for Example* 2 *using* 256 *particles.*

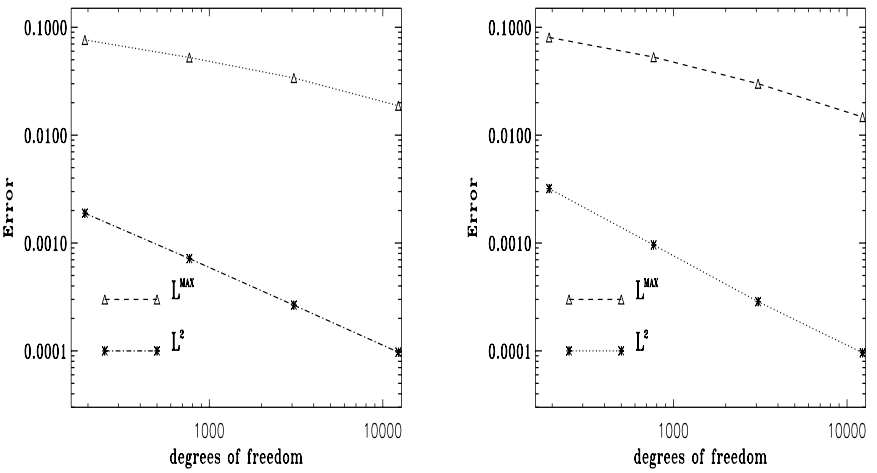


FIGURE 5.7. *Convergence history for Example* 2 *for the uniform particle distribution (left) and the graded particle distribution (right).*

the continuous problem from Example 1, but we will now apply the p-version of our particle-PUM. Here, the particle distribution is fixed, and we use 256 particles generated via a $(2, 3)$ Halton sequence and a stretch parameter $\alpha = 1.5$. The weight function used in this experiment is the well-known hat function. Furthermore, we use the complete polynomials V_i^C as local approximation spaces with degrees $p_i = p$ and p ranging from one to five. We choose tensor products of Legendre polynomials as local basis functions.

For analytic solutions u we may estimate the error $u - \tilde{u}$ on every cover patch ω_i using the Taylor-series expansion (see section 4.1.4). We obtain the local estimate

$$\|u - \tilde{u}\| = O\left(\|D^{(p+1)}u\| \frac{h^{p+1}}{(p+1)!}\right) \tag{5.10}$$

for an approximation $\tilde{u}$ of degree p, which leads to an exponential error estimate

$$\|u - \tilde{u}\| = O\left(\exp(-b\sqrt{\mathsf{dof}_p})\right) \tag{5.11}$$

on every cover patch ω_i. In our first experiment with the p-version we choose f and

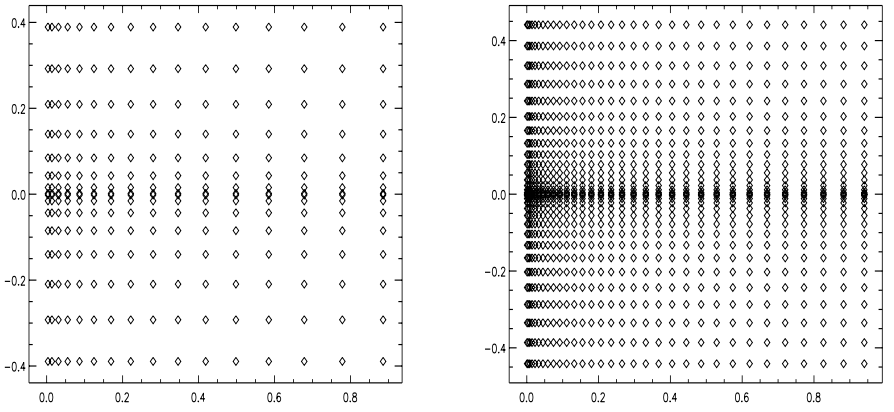


FIGURE 5.8. *Graded particle distribution via transformation:* $(x, y) \mapsto (x^2, \pm 2y^2)$.

g such that the continuous solution to the problem is given as

$$u(x, y) = \exp(4(x + y)). \tag{5.12}$$

This solution is sufficiently smooth for (5.11) to hold. Therefore, we expect to see an exponential convergence in this experiment. The plots of the measured errors against the number of degrees of freedom are displayed in Figure 5.9 (left). Their gradients ρ as defined in (5.5) are given in Table 5.5. These results clearly show the expected exponential convergence behavior.

Example 4 (p-version of particle-PUM, limited smoothness of solution). In a second experiment with the p-version of our particle-PUM we choose f and g such that the continuous solution to the problem is given by

$$u(x, y) = (x^2 + y^2)^{5/4}. \tag{5.13}$$

Here, the estimate (5.11) does not hold since (5.13) is not analytic. Therefore, we may not expect an exponential convergence behavior of the p-version of our particle-PUM in this experiment.

We use the same particle distribution, the same cover, and the same local approximation spaces V_i as in Example 3 with p now ranging from one to eight. Figure 5.10 shows the solution (5.13) and the error for $p = 5$. The measured errors are displayed in Figure 5.9 (right). The corresponding rates ρ and the measured errors are given in Table 5.6. From these results we clearly see an algebraic convergence behavior of the p-version of our particle-PUM for problems with solutions with limited smoothness properties. A similar behavior can be observed when we employ a finite element discretization with the p-version on a fixed uniform grid. In fact, a convergence rate of $2 \cdot 5/4$ ($\rho_{H^1} = -2.5$) in the H^1-norm is well known for the classical p-version of the FEM [5] for this problem and we (almost) recover this convergence behavior with our particle-PUM.

This experiment again demonstrates that the p-version of the particle-PUM based on a pseudo Monte Carlo particle distribution converges as well as the corresponding p-version of the finite element method.

In summary, the results of our numerical experiments with the h-version and p-version of our particle-PUM applied to elliptic problems clearly show that the method possesses the advantages of a particle method. At the same time, it exhibits a convergence behavior similar to that of the respective version of the finite element method.

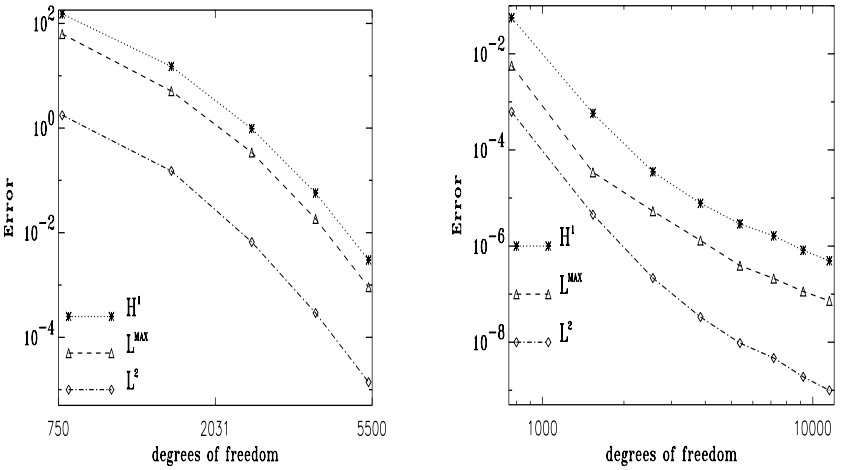


FIGURE 5.9. *Convergence history for Example* 3 *(left), Example* 4 *(right).*

TABLE 5.5
Errors and convergence rates for Example 3.

N	p	dof	$\|e\|_{L^\infty}$	ρ_{L^∞}	$\|e\|_{L^2}$	ρ_{L^2}	$\|e\|_{H^1}$	ρ_{H^1}
256	1	768	6.292_1	$***$	1.761_0	$***$	1.514_2	$***$
256	2	1536	5.094_0	-3.627	1.512_{-1}	-3.542	1.507_1	-3.329
256	3	2560	3.423_{-1}	-5.286	6.628_{-3}	-6.122	9.738_{-1}	-5.362
256	4	3840	1.837_{-2}	-7.207	2.920_{-4}	-7.700	5.732_{-2}	-6.986
256	5	5376	9.130_{-4}	-8.921	1.392_{-5}	-9.044	3.007_{-3}	-8.761

5.3. Parabolic problem. In the following we consider the heat-equation

$$\frac{\partial u}{\partial t} - \Delta u = 0 \text{ in } \Omega := (-0.5, 0.5)^2 \times (0, T) \tag{5.14}$$

with vanishing Neumann boundary conditions $\partial u/\partial n = 0$ on $\Gamma := \partial\Omega$ and initial value $u(x, y, 0) = \exp(-100(x^2+y^2))$ in $\Omega := (-0.5, 0.5)^2$. For this parabolic model problem we use a random walk process as described in section 2 to simulate the diffusive transport on the particle positions. Furthermore, we reduce the parabolic problem to a sequence of elliptic problems via a standard implicit Eulerian time discretization which gives the usual $O(\Delta t)$ convergence behavior in time. We employ the shape functions from the PUM based on the particle positions after the random walk process to discretize the elliptic problem in every time step. Here, the weight functions W_i are the hat functions. The stretch parameter in this experiment is $\alpha = 1.25$. As local approximation spaces V_i we use the complete quadratic polynomials V_i^C. We use a block-structure approach using an approximate support $\hat{\Omega}$ of the initial value as described in section 5.1 to resolve the initial value. The particle distributions and isolines of the solution at times $(5\Delta t, 15\Delta t, 25\Delta t, 35\Delta t)$ with $\Delta t = 0.001$ are displayed in Figure 5.11. Here, we clearly see the diffusive particle transport and the reduction of the total number of particles over time. The diffusive particle transport adapts the distribution of the spatial degrees of freedom appropriately to the evolution of the solution. The solution gets smoother over time due to the diffusion. Here, the solution can be resolved to the same accuracy as the initial value with fewer unknowns, i.e., a successively reduced number of particles is sufficient to give a good approximation on Ω. We start with 469 particles (see Figure 5.12) and after 35 time steps there are

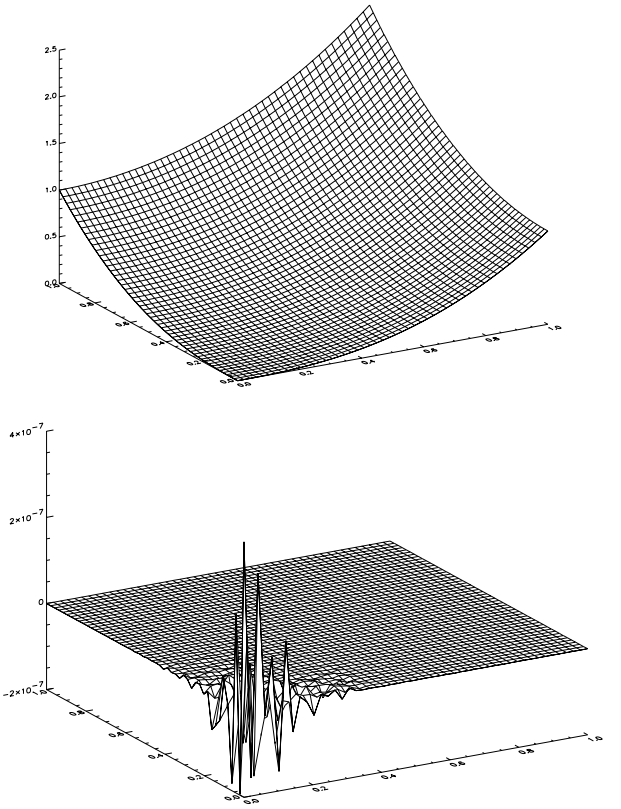


FIGURE 5.10. *Graph of solution* (5.13) *and graph of error for Example* 4 *with* $p = 5$.

TABLE 5.6
Errors and convergence rates for Example 4.

N	p	dof	$\|e\|_{L^\infty}$	ρ_{L^∞}	$\|e\|_{L^2}$	ρ_{L^2}	$\|e\|_{H^1}$	ρ_{H^1}
256	1	768	5.694_{-3}	$***$	6.208_{-4}	$***$	4.073_{-1}	$***$
256	2	1536	3.429_{-5}	-7.375	4.516_{-6}	-7.103	5.614_{-2}	-6.605
256	3	2560	5.335_{-6}	-3.642	2.165_{-7}	-5.947	5.765_{-4}	-5.479
256	4	3840	1.311_{-6}	-3.461	3.346_{-8}	-4.605	3.510_{-5}	-3.726
256	5	5376	3.931_{-7}	-3.580	9.608_{-9}	-3.708	7.747_{-6}	-2.926
256	6	7168	2.116_{-7}	-2.153	4.696_{-9}	-2.489	2.894_{-6}	-1.999
256	7	9216	1.141_{-7}	-2.460	1.908_{-9}	-3.583	8.173_{-7}	-2.743
256	8	11520	7.254_{-8}	-2.028	9.972_{-10}	-2.909	4.935_{-7}	-2.261

only 374 left (see Figure 5.11). But the number of particles within the approximate support $\hat{\Omega}$ (while $\hat{\Omega} \subset \Omega$) is almost constant over time, i.e., the solution in every time step is represented by an almost constant number of degrees of freedom in space. This is due to the Monte Carlo simulation of the diffusion onto the particle distribution and the Neumann boundary conditions. Overall, we only reduce the computational work by applying the Monte Carlo step, but the spatial resolution in every time step is sufficiently high to give a good approximation of the solution to (5.14). Due to use of quadratic polynomials V_i^C we have a "third order" convergence in space.

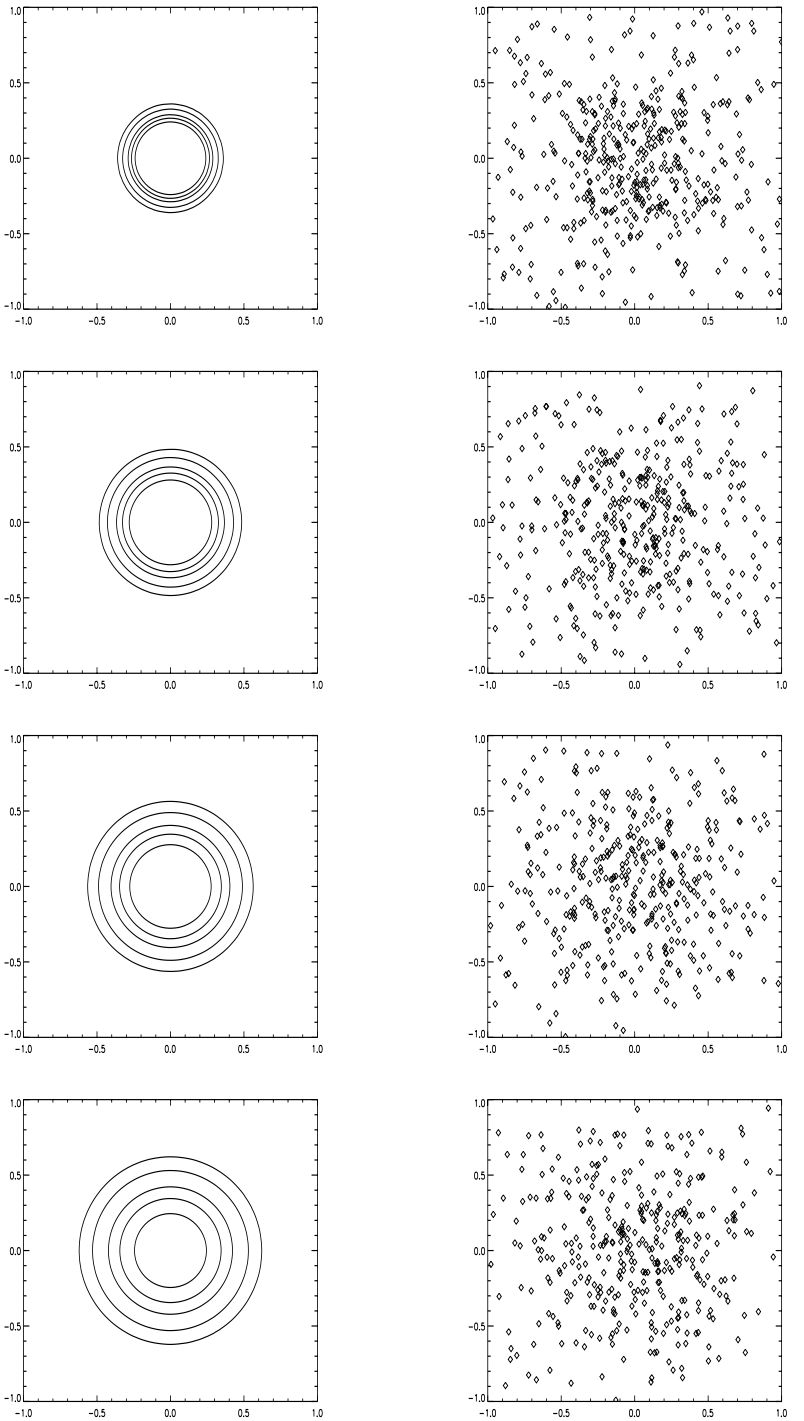


FIGURE 5.11. *Particle distributions and isolines* $(0.005, 0.01, 0.02, 0.03, 0.045)$ *of the approximate solution for* (5.14) *at times* $(5\Delta t, 15\Delta t, 25\Delta t, 35\Delta t)$ *with* $\Delta t = 10^{-3}$.

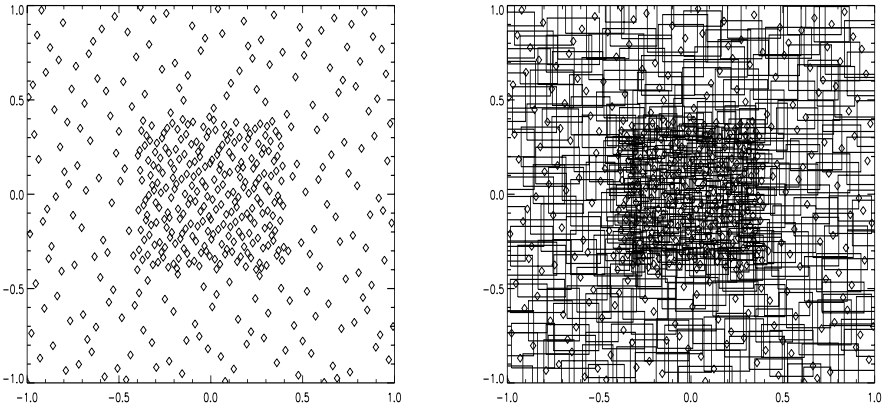


FIGURE 5.12. *Initial particle distribution (left) and the constructed cover (right) for* (5.14).

5.4. Convection-diffusion problem. In the following we consider the model problem

$$\frac{\partial u}{\partial t} - \frac{1}{125}\Delta u + \frac{1}{\sqrt{x^2+y^2}} \begin{pmatrix} -y \\ x \end{pmatrix} \nabla u = 0 \text{ in } \Omega := (-1,1)^2 \times (0,T) \tag{5.15}$$

with vanishing Neumann boundary conditions $\partial u/\partial n = 0$ on $\Gamma := \partial\Omega$ and initial value $u(x,y,0) = \exp(-100((x-\frac{1}{4})^2 + (y-\frac{1}{4})^2))$.

We discretize (5.15) via the particle method presented in section 2. First, we split the operator $\frac{\partial}{\partial t} - \nu\Delta + v\nabla$ into a hyperbolic operator $\frac{\partial}{\partial t} + v\nabla$ and a parabolic operator $\frac{\partial}{\partial t} - \nu\Delta$. Then, we apply the corresponding particle method to the resulting hyperbolic problem (see section 3, section 5.1) with a fourth order Runge–Kutta method, and to the remaining parabolic problem (see section 2, section 5.3). Again, we use $\alpha = 1.25$, hat functions as weight functions W_i for the construction of the partition of unity $\{\varphi_i\}$ and a block-structure approach to resolve the initial value. As local approximation spaces V_i we use the complete quadratic polynomials V_i^C.

The particle distributions and isolines of the solution at $(5\Delta t, 10\Delta t, 15\Delta t, 20\Delta t)$ with $\Delta t = 2\pi/50$ are displayed in Figure 5.13. Here, we see that the diffusive particle movement together with the particle movement induced by the Lagrangian discretization of the convective terms of (5.15) causes the particles to follow the solution over time. Thus, we have roughly the same number of degrees of freedom for the resolution of the solution in every time step. Note that the Lagrangian treatment of the convection part circumvents the stability problems we might experience with a finite element discretization for the convective terms of (5.15). Further experiments showed the expected discretization error of "third order" in space and the expected $O(\Delta t)$ discretization error in time which is due to the first order implicit Eulerian time discretization for the parabolic subproblem.

6. Concluding remarks. We presented a meshless Lagrangian discretization method for instationary convection-diffusion problems. The method combines a particle approach and a meshless method for the generation of shape functions and the adaptation of the distribution of nodes (particles). We gave the details of the implementation and results of our numerical experiments. They showed that the convergence properties of the h-version and p-version of our particle-PUM for elliptic problems are comparable to those of the h-version and p-version of the FEM, respectively.

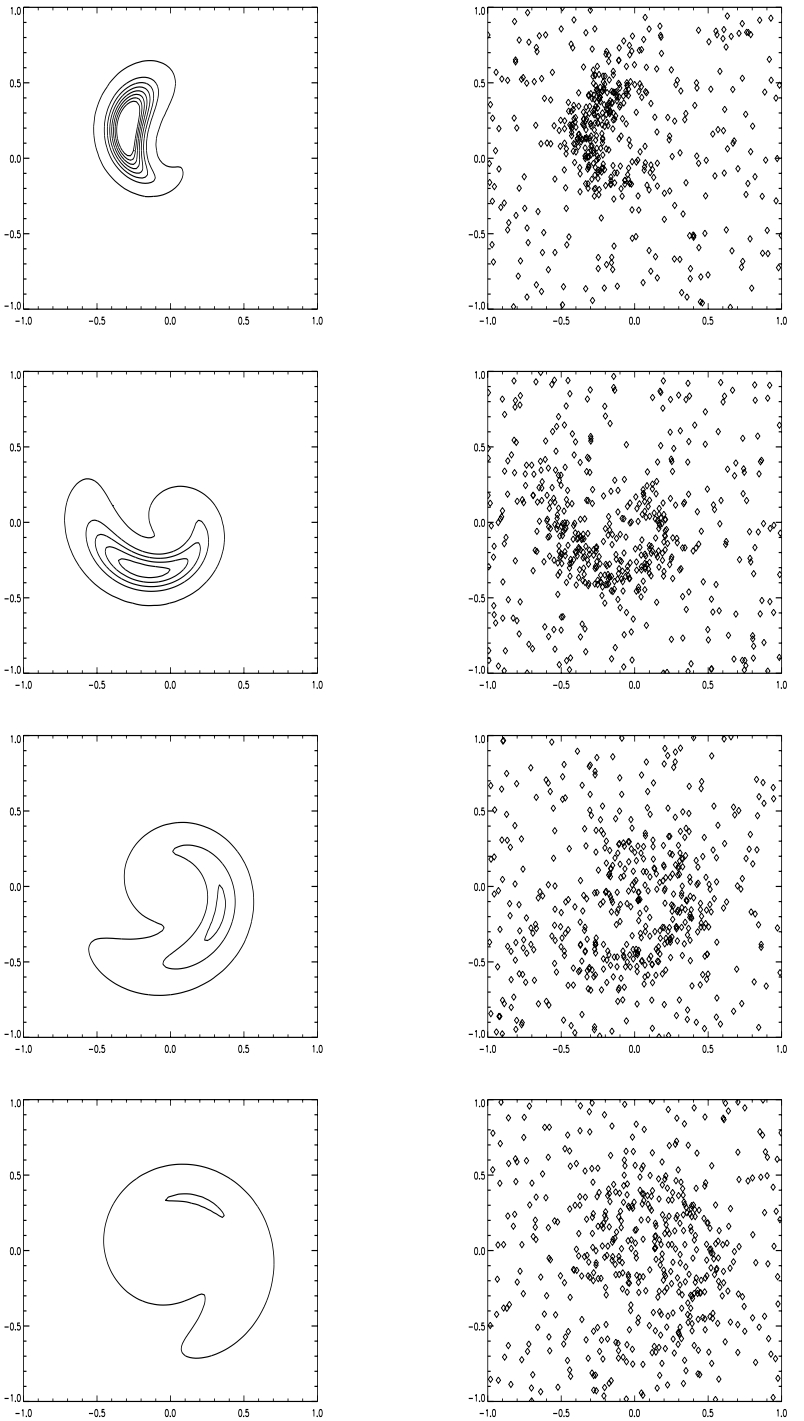


FIGURE 5.13. *Particle distributions and isolines* $(0.01, 0.05, 0.075, 0.1, 0.125, 0.15, 0.175, 0.2)$ *of the approximate solution for* (5.15) *at times* $(5\Delta t, 10\Delta t, 15\Delta t, 20\Delta t)$ *with* $\Delta t = 2\pi/50$.

The convergence behavior in time of the proposed method for hyperbolic problems corresponds to the convergence behavior of the employed ODE solver. Furthermore, a fewer number of degrees of freedom in space than with a uniform semi-Lagrangian method or even Euler method may be used since the degrees of freedom follow the solution over time. Hence, we do not have to resolve the complete computational domain with the same accuracy. Adaptive semi-Lagrangian or Euler methods would have to apply refinement and coarsening in every time step whereas our method will transform an adaptively refined particle distribution for the initial value into an adaptively refined particle distribution in every time step without applying explicit refinement and coarsening.

Up to now, the method is in an experimental state and is surely not yet competitive to existing grid-based methods. The efficiency of the method has to be improved. The most room for improvement is during the construction of a cover, the integration of the shape functions and the iterative solution of the resulting linear system, especially in the case of an elliptic problem with Dirichlet boundary conditions where we have to solve a saddle point problem. Furthermore, an adaptive refinement, i.e., the proper insertion of particles (h-version) as well as the local increment of the degree of the polynomials used (p-version) or both together (hp-version), needs the development of reliable local error indicators and estimators. Moreover, an adaptive control of the time step size, the use of local time stepping methods and the choice of more sophisticated operator splittings could further improve the performance of the method for time-dependent problems.

In summary, much additional work remains to be done, but the results presented in this paper demonstrate the promising properties of the particle-PUM and encourage its further development.

REFERENCES

[1] H. Babovsky, *Die Boltzmann–Gleichung*, Teubner, Stuttgart, 1998.

[2] I. Babuška, *The finite element method with Lagrangian multipliers*, Numer. Math., 20 (1973), pp. 179–192.

[3] I. Babuška and J. M. Melenk, *The partition of unity finite element method: Basic theory and applications*, Comput. Methods Appl. Mech. Engrg., 139 (1996), pp. 289–314.

[4] I. Babuška and J. M. Melenk, *The partition of unity method*, Internat. J. Numer. Methods Engrg., 40 (1997), pp. 727–758.

[5] I. Babuška and M. Suri, *The p and h-p versions of the finite element method, basic principles and properties*, SIAM Rev., 36 (1994), pp. 578–632.

[6] M. J. Baines, *Moving Finite Elements*, Monographs on Numerical Analysis, Oxford Science Publications, London, 1994.

[7] R. E. Bank, *PLTMG: A Software Package for Solving Elliptic Partial Differential Equations. Users' Guide* 7.0, Frontiers in Appl. Math. 15, SIAM, Philadelphia, 1994.

[8] P. Bastian, *Parallele Adaptive Mehrgitterverfahren*, Teubner, Stuttgart, 1996.

[9] P. Bastian, K. Birken, K. Johannsen, S. Lang, K. Eckstein, N. Neuss, H. Rentz-Reichert, and C. Wieners, *UG—A flexible software toolbox for solving partial differential equations*, Comput. Visual. Sci., 1 (1997), pp. 27–40.

[10] R. Beck, B. Erdmann, and R. Roitzsch, *Kaskade* 3.0 *Users Guide*, Tech. Report 95-11, Konrad-Zuse-Zentrum für Informationstechnik, Berlin, 1995.

[11] T. Belytschko, Y. Krongauz, D. Organ, M. Fleming, and P. Krysl, *Meshless methods: An overview and recent developments*, Comput. Methods Appl. Mech. Engrg., 139 (1996), pp. 3–47.

[12] T. Belytschko, Y. Y. Lu, and L. Gu, *Element-free galerkin methods*, Internat. J. Numer. Methods Engrg., 37 (1994), pp. 229–256.

[13] M. J. Berger and J. Oliger, *An adaptive mesh refinement for hyperbolic partial differential equations*, J. Comput. Phys., 53 (1984), pp. 484–512.

[14] D. Braess, *Finite Elemente*, 2nd ed., Springer, Berlin, Heidelberg, 1997.

[15] J. H. Bramble, *The Lagrange multiplier method for Dirichlet's problem*, Math. Comput., 37 (1981), pp. 1–11.
[16] J. H. Bramble, J. E. Pasciak, and A. T. Vassilev, *Analysis of the inexact Uzawa algorithm for saddle point problems*, J. Numer. Anal., 34 (1997), pp. 1072–1092.
[17] J. H. Bramble, J. E. Pasciak, and A. T. Vassilev, *Uzawa type algorithms for nonsymmetric saddle point problems*, Math. Comp., 69 (2000), pp. 667–689.
[18] J. H. Bramble, J. E. Pasciak, and P. S. Vassilevski, *Computational scales of Sobolev norms with application to preconditioning*, Math. Comp., 69 (2000), pp. 463–480.
[19] F. Brezzi and M. Fortin, *Mixed and Hybrid Finite Element Methods*, Springer, Berlin, Heidelberg, 1991.
[20] W. Dai and P. Woodward, *Extension of the piecewise parabolic method to multidimensional ideal magnetohydrodynamic*, J. Comput. Physics, 115 (1994), pp. 485–514.
[21] P. J. Davis, *Interpolation and Approximation*, Dover, New York, 1975.
[22] L. Demkowicz, J. T. Oden, W. Rachowicz, and O. Hardy, *Toward a universal HP adaptive finite element strategy, Part* 1. *Constrained approximation and data structure*, Comput. Methods Appl. Mech. Engrg., 77 (1989), pp. 79–112.
[23] P. Deufelhard, P. Leinen, and H. Yserantant, *Concepts of an adaptive hierarchical finite element code*, IMPACT Comput. Sci. Engrg., 1 (1989), pp. 3–35.
[24] G. A. Dilts, *Moving-least-square-particle hydrodynamics* I: *Consistency and stability*, Internat. J. Numer. Methods Engrg., 44 (1999), pp. 1115–1155.
[25] J. Dolbow and T. Belytschko, *Numerical integration of the Galerkin weak form in meshfree methods*, Comput. Mech., 23 (1999), pp. 219–230.
[26] C. A. M. Duarte, *A Review of Some Meshless Methods to Solve Partial Differential Equations*, Tech. Report 95-06, TICAM, University of Texas, 1995.
[27] C. A. M. Duarte and J. T. Oden, *HP clouds—A meshless method to solve boundary value problems*, Numer. Methods Partial Differential Equations, 12 (1996), pp. 673–705.
[28] C. Franke and R. Schaback, *Convergence orders of meshless collocation methods using radial basis functions*, Adv. Comput. Math., 8 (1998), pp. 381–399.
[29] C. Franke and R. Schaback, *Solving partial differential equations by collocation using radial basis functions*, Appl. Math. Comput., 93 (1998), pp. 73–82.
[30] T. Gerstner and M. Griebel, *Numerical integration using sparse grids*, Numer. Algebra, 18 (1998), pp. 209–232.
[31] R. A. Gingold and J. J. Monaghan, *Smoothed particle hydrodynamics: Theory and application to non-spherical stars*, Mon. Not. R. Astr. Soc., 181 (1977), pp. 375–389.
[32] R. A. Gingold and J. J. Monaghan, *Kernel estimates as a basis for general particle methods in hydrodynamics*, J. Comput. Physics, 46 (1982), pp. 429–453.
[33] R. T. Glassey, *The Cauchy Problem in Kinetic Theory*, SIAM, Philadelphia, 1996.
[34] T. Grauschopf, M. Griebel, and H. Regler, *Additive multilevel-preconditioners based on bilinear interpolation, matrix dependent geometric coarsening and algebraic multigrid coarsening for second order elliptic PDEs*, Appl. Numer. Math., 23 (1997).
[35] E. Hlawka and R. Mück, *Über eine Transformation von gleichverteilten Folgen* II, Computing, 9 (1972), pp. 127–138.
[36] J. Hoschek and D. Lasser, *Grundlagen der geometrischen Datenverarbeitung*, B. G. Teubner, Stuttgart, 1992.
[37] K. Hvistendahl Karlsen, K. Brusdal, H. K. Dahle, S. Evje, and K.-A. Lie, *The corrected operator splitting approach applied to a nonlinear advection-diffusion problem*, Comput. Methods Appl. Mech. Engrg., 167 (1998), pp. 239–260.
[38] K. Hvistendahl Karlsen and N. H. Risebro, *An operator splitting method for nonlinear convection-diffusion equations*, Numer. Math., 77 (1997), pp. 365–382.
[39] C. Johnson, *Numerical solution of partial differential equations by the finite element method*, Cambridge University Press, Cambridge, UK, 1987.
[40] E. J. Kansa, *Multiquadratics—a scattered data approximation scheme with applications to computational fluid-dynamics*—I. *Surface approximations and partial derivative estimates*, Comput. Math. Appl., 19 (1990), pp. 127–145.
[41] E. J. Kansa, *Multiquadratics—a scattered data approximation scheme with applications to computational fluid-dynamics*—II. *Solutions to parabolic, hyperbolic and elliptic partial differential equations*, Comput. Math. Appl., 19 (1990), pp. 147–161.
[42] D. E. Knuth, *The Art of Computer Programming, Vol.* 3, *Searching and Sorting,* 2nd ed., Addison–Wesley, Reading, MA, 1998.
[43] A. Kunoth, *Multilevel preconditioning—appending boundary conditions by Lagrange multipliers*, Adv. Comput. Math., 4 (1995), pp. 145–170.
[44] P. Lancaster, *Moving weighted least-squares methods*, in Polynomial and Spline Approxi-

mation, B. N. Sahney, ed., NATO Adv. Stud. Series C 49, Dordrecht, Boston, London, 1979.
[45] P. Lancaster and K. Salkauskas, *Surfaces generated by moving least squares methods*, Math. Comput., 37 (1981), pp. 141–158.
[46] J. Lang, *Adaptive finite element methods for reaction-diffusion equations*, Appl. Numer. Math., (1998), pp. 105–116.
[47] U. Langer and W. Queck, *Preconditioned Uzawa-Type Iterative Methods for Solving Mixed Finite Element Equations*, Wissenschaftliche Schriften 3/1987, Technische Universität Karl–Marx–Stadt, 1987.
[48] D. Lanser and J. G. Verwer, *Analysis of Operator Splitting for Advection-Diffusion-Reaction Problems*, J. Comp. Appl. Math., 111 (1999), pp. 201–216.
[49] T. Liska and J. Orkisz, *The finite difference method at arbitrary irregular grids and its application in applied mechanics*, Comput. Struct., 11 (1980), pp. 83–95.
[50] W. K. Liu, S. Jun, and Y. F. Zhang, *Reproducing kernel particle methods*, Internat. J. Numer. Methods Fluids, 20 (1995), pp. 1081–1106.
[51] L. B. Lucy, *A numerical approach to the testing of the fission hypothesis*, Astronomical J., 82 (1977), pp. 1013–1024.
[52] Y. Maday, C. Mavriplis, and A. T. Patera, *Nonconforming mortar element methods: Application to spectral discretizations*, in Proceedings of the Second International Symposium Domain Decomposition Methods for Partial Differential Equations, T. F. Chan, R. Glowinski, J. Périaux, and O. B. Widlund, eds., SIAM, Philadelphia, 1989, pp. 392–418.
[53] K. Miller, *Moving finite elements* II, SIAM J. Numer. Anal., 18 (1981), pp. 1033–1057.
[54] K. Miller and R. N. Miller, *Moving finite elements* I, SIAM J. Numer. Anal., 18 (1981), pp. 1019–1032.
[55] J. J. Monaghan, *An introduction to SPH*, Comput. Phys. Comm., 48 (1977), pp. 89–96.
[56] J. J. Monaghan, *Why particle methods work*, SIAM J. Sci. Statist. Comput., 3 (1982), pp. 422–433.
[57] K. Nanbu, *Direct simulation scheme derived from the Boltzmann equation*, J. Phys. Soc. Japan, 49 (1980), pp. 20–49.
[58] K. Nanbu, *Theoretical basis on the direct simulation Monte Carlo method*, in Rarefied Gas Dynamics, Vol. 1, V. Boffi and C. Cercignani, eds., Teubner, Stuttgart, 1986.
[59] S. V. Nepomnyaschikh, *Decomposition and fictitious domain methods for elliptic boundary value problems*, in Proceedings of the Fifth International Symposium on Domain Decomposition Methods for Partial Differential Equations, T. F. Chan, D. E. Keyes, G. A. Meurant, J. S. Scroggs, and R. G. Voigt, eds., SIAM, Philadelphia, 1992.
[60] H. Neunzert, A. Klar, and J. Struckmeier, *Particle Methods: Theory and Applications*, Tech. Report 95-153, Arbeitsgruppe Technomathematik, Universität Kaiserslautern, 1995.
[61] H. Neunzert and J. Struckmeier, *Particle methods for the Boltzmann equation*, Acta Numerica, (1995), pp. 417–457.
[62] H. Niederreiter, *Random Number Generation and Quasi-Monte Carlo Methods*, SIAM, Philadelphia, 1992.
[63] E. Novak and K. Ritter, *High dimensional integration of smooth functions over cubes*, Numer. Math., 75 (1996), pp. 79–97.
[64] J. T. Oden, T. Strouboulis, and P. Devloo, *Adaptive finite element methods for the analysis of inviscid compressible flow: Part* I. *Fast refinement/unrefinement and moving mesh methods for unstructured meshes*, Comput. Methods Appl. Mech. Engrg., 59 (1986), pp. 327–362.
[65] E. S. Oran and J. P. Boris, *Numerical Simulation of Reactive Flow*, Elsevier, New York, 1987.
[66] J. Ruge and K. Stüben, *Efficient solution of finite difference equations by algebraic multigrid*, Arbeitspapiere 89, GMD, Bonn, 1984.
[67] J. Ruge and K. Stüben, *Algebraic multigrid*, Arbeitspapiere 210, GMD, Bonn, 1986.
[68] M. A. Schweitzer, *Ein Partikel-Galerkin-Verfahren mit ansatzfunktionen der partition of unity method*, Diplomarbeit, Institut für Angewandte Mathematik, Universität Bonn, 1997.
[69] D. Shepard, *A two-dimensional interpolation function for irregularly spaced data*, in Proceedings 1968 ACM Nat. Conf., ACM, New York, 1968, pp. 517–524.
[70] G. Strang, *On the construction and comparison of difference schemes*, J. Numer. Anal., 5 (1968), pp. 506–517.
[71] J. W. Swegle, S. W. Attaway, F. J. Mello, and D. L. Hicks, *An Analysis of the Smoothed Particle Hydrodynamics*, Tech. Report SAND93-2513 UC-705, Sandia National Laboratories, Los Alamos, NM, 1994.
[72] C. B. Vreugdenhil and B. Koren, eds., *Numerical Methods for Advection-Diffusion Prob-*

lems, Notes Numer. Fluid Mechanics 45, Vieweg, Braunschweig, Wiesbaden, 1993.
[73] H. Wendland, *Meshless Galerkin methods using radial basis functions*, Math. Comput., 68 (1999), pp. 1521–1531.
[74] H. Yserantant, *A new class of particle methods*, Numer. Math., 76 (1997), pp. 87–109.
[75] H. Yserantant, *A particle model of compressible fluids*, Numer. Math., 76 (1997), pp. 111–142.
[76] *Special Issue on Meshless Methods*, Comput. Methods Appl. Mech. Engrg., 139 (1996).

SIAM J. SCI. COMPUT.
Vol. 22, No. 3, pp. 891–916
© 2000 Society for Industrial and Applied Mathematics

A PARALLEL SCHWARZ METHOD FOR A CONVECTION-DIFFUSION PROBLEM*

M. GARBEY†, YU. A. KUZNETSOV‡, AND YU. V. VASSILEVSKI§

Abstract. This paper describes a parallel convection-diffusion solver which may be used as part of a Navier–Stokes solver for three-dimensional channel flow at moderately large Reynolds numbers [S. Turek, *Efficient Solvers for Incompressible Flow Problems: An Algorithmic Approach in View of Computational Aspects*, Springer-Verlag, 1999]. The solver uses a multiplicative Schwarz domain decomposition with overlapping subdomains to solve singularly perturbed convection-diffusion equations where convection is dominant. Upwind finite differences are used for the spatial discretization. The algorithm uses special features of the singularly perturbed convection-diffusion operator. The error due to a local perturbation of the boundary conditions decays extremely fast, in the upwind as well as the crosswind direction, so the overlap in the domain decomposition can be kept to a minimum. The algorithm parallelizes well and is particularly suited for applications in three dimensions. Results of two- and three-dimensional numerical experiments are presented.

Key words. convection-diffusion, overlapping domain decomposition, maximum principle, upwind finite differences, damping factor

AMS subject classifications. Primary, 35B25, 65N05; Secondary, 35J25, 35J40

PII. S1064827598335854

1. Introduction.

The importance of efficient convection-diffusion solvers is difficult to overestimate. Apart from their direct use in modeling schemes, for example, in pollution propagation, they constitute an essential part of several numerical techniques for the Navier–Stokes equations [8, 22]. In this paper, we present a new parallel convection-diffusion solver, which may be incorporated in a Navier–Stokes solver for three-dimensional channel flow at moderately large Reynolds number [21]. We focus on algorithmic (rather than approximation) issues for the convection-diffusion equations (adaptive grids, boundary layers, etc.). It is well known that the performance of an algorithm for linear singular perturbation problems depends not so much on the actual value of the small perturbation parameter as on its value relative to the mesh size [5]. In this paper we focus therefore on the analysis of the discrete operator, taking this relative value into account.

To solve a singularly perturbed convection-diffusion equation with strongly dominant convection, we apply the multiplicative Schwarz method for domain decomposition with overlapping subdomains. The algorithm is based on special features of the singularly perturbed convection-diffusion operator [20]. In particular, we take advantage of an anisotropic propagation of the perturbation associated with any local perturbation of the boundary conditions. In the case of a constant transport vector, a pointwise perturbation causes the perturbation to propagate only in the downwind direction and vanish rapidly both in the *upwind* and *crosswind* directions. Figure 1 (upper row) illustrates this phenomenon for a pointwise perturbation of homogeneous

*Received by the editors February 16, 1998; accepted for publication (in revised form) June 2, 1999; published electronically August 31, 2000.

http://www.siam.org/journals/sisc/22-3/33585.html

†Université Claude Bernard Lyon-1, CDCSP, ISTIL-Bâtiment 101, 43 Bd du 11 Novembre 1918, 69622 Villeurbanne Cedex, Lyon, France (garbey@cdcsp.univ-lyon1.fr).

‡Department of Mathematics, University of Houston, Houston, TX 77204-3476 (kuz@math.uh.edu).

§Institute of Numerical Mathematics, Russian Academy of Sciences, 8 ul. Gubkina, 117333 Moscow, Russia (vasilevs@labnumat.inm.ras.ru).

Dirichlet boundary conditions at the midpoints of the left, upper, and right boundaries of the unit square. For an arbitrary transport vector, this is no longer the case. Directional variations of the transport vector in the vicinity of a pointwise perturbation of the boundary conditions tend to smear-out the solution perturbation. Figure 1 (bottom row) illustrates this phenomenon for a strong vortex near a pointwise perturbation of the boundary conditions.

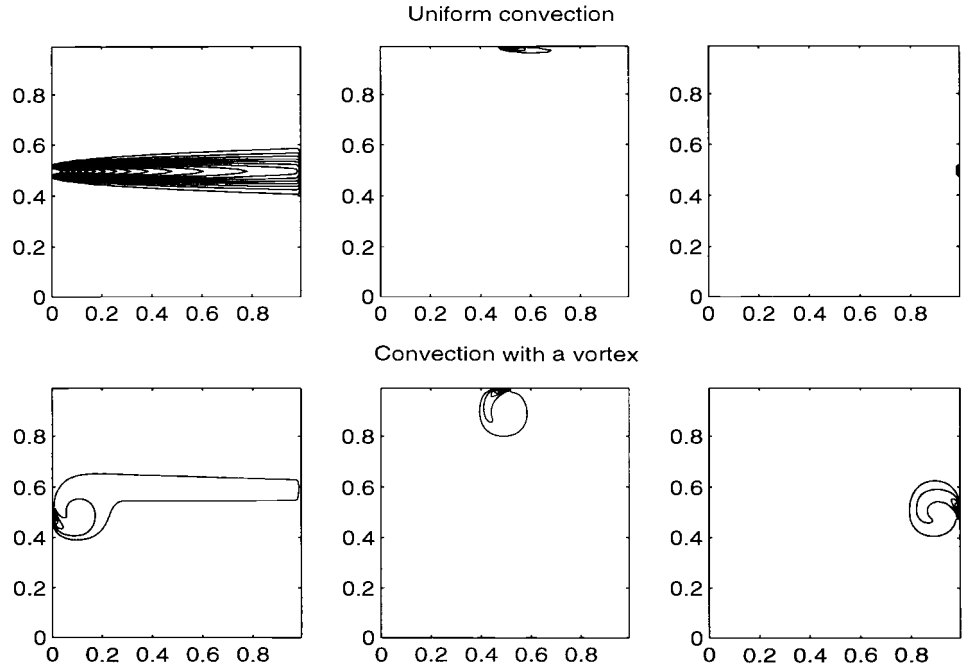


FIG. 1. *Solutions to problems with a pointwise perturbation of homogeneous Dirichlet boundary conditions.*

We propose a parallel algorithm for the solution of singularly perturbed convection-diffusion problems described by the operator

$$\mathcal{L} = -\varepsilon\Delta + B^1(x_1, x_2) \cdot \frac{\partial}{\partial x_1} + B^2(x_1, x_2) \cdot \frac{\partial}{\partial x_2},$$

where $\vec{B} = (B^1, B^2)^T$, with B^1 positive in the main part of the domain and B^2 of the same order as B^1 or smaller.

Problems with transport vectors $\vec{B}$ with *small deviations* from the prescribed downwind direction (B^2 small compared to B^1) are solved by the following two-level Schwarz method. At the outer level, we split the original domain into overlapping crosswind strips (slices). The classical Schwarz method in the downwind direction is the outer part of the algorithm. If the trace of B^1 on the artificial interfaces is positive and bounded below by a positive constant B_0 of order one, very few outer iterations are needed to reach a prescribed accuracy. The fast convergence is a consequence of the upwind decay. It is natural to use a Schwarz method for the inner iterations. We split each strip-subdomain into overlapping boxes and define the inner iterations using a version of either the multiplicative or the additive Schwarz method. The novel feature of the version of the Schwarz method selected for the inner iterations is

that the choice of overlapping domain decomposition is based on the fast crosswind decay phenomenon which is present when the trace of B^2 is smaller than the trace of B^1 on the artificial interfaces. At the level of the inner iterations, we have natural parallelism, since each of the box-subdomains can be treated by its own processor. Actually, one can consider a family of parallel methods by varying the type of solver within the strip-subdomain. It is very important that the number of iterations between the box-subdomains with odd and even indices may be very small (1 or 2) and small overlappings practically do not affect the fast convergence of the algorithm.

For problems with a transport vector with *large deviations* from the prescribed downwind direction we introduce an adaptive two-level Schwarz method based on the automatically adapted first-stage domain decomposition. The first-stage splitting is chosen in such a way that the B^1-component of the transport vector is strictly positive on the artificial interfaces. For instance, if the transport vector contains a strong stagnant vortex in some region, the strip-subdomain has to be chosen sufficiently "thick" to contain the vortex. Moreover, the number of iterations between the box-subdomains with odd and even indices has to be increased in those thick strip-subdomains that slow down the iterations. Nevertheless, we show that the method maintains a fast convergence rate. As long as the vertex zone is small compared to the size of the domain, the algorithm is efficient and readily parallelizable. The techniques are readily generalized to the three-dimensional case.

The theoretical analysis of the above algorithm is given in [6, 7]. It is performed on the discrete level. The properties of grid functions satisfying certain systems of algebraic equations constitute the core of the analysis. The approximation scheme, which uses finite differences, must satisfy the discrete maximum principle.

Numerical results for the additive version of the two-stage Schwarz method applied to singularly perturbed convection-diffusion problem are presented in [10]. For domain-decomposition algorithms based on discrete upwind decaying phenomenon, we refer to [9, 19]. Other results concerning domain-decomposition methods for singularly perturbed elliptic equations can be found in [1, 4, 5, 2, 12, 13, 14]. The decay of the discrete Green's function has been analyzed in [11, 15] for the case of streamline upwinding Petrov–Galerkin discretization. For a comprehensive study of differential solution properties, we refer to [20, 3]. Parallel multigrid techniques relevant to the problem of interest are considered in [17, 18].

The remainder of the paper is organized as follows. In section 2 we consider the simplest singularly perturbed convection-diffusion operator with a constant transport vector. First, we investigate the upwind and crosswind error propagation. Second, we assemble the results for the propagation errors to analyze the convergence property of the two-level Schwarz method. Section 3 is devoted to the convection-diffusion operator with a variable transport vector. We present an adaptive version of the two-level Schwarz method for solving the discrete problem and discuss the convergence of the method. Section 4 gives implementation details for the adaptive technique for splitting the computational domain and providing fast convergence and efficient parallelization of the algorithm. In section 5, we present the results of the numerical experiments. In sections 5.1 and 5.2, we consider two- and three-dimensional problems and report on the numerical results in terms of the convergence and parallel performance.

2. Convection-diffusion problem with a constant transport vector. We consider a two-dimensional convection-diffusion equation with constant coefficients in

the unit square:

$$\begin{aligned} \mathcal{L}u &= f \qquad \text{in } \Omega = (0;1)\times(0;1), \\ u &= g \qquad \text{on } \Gamma_D, \\ \frac{\partial u}{\partial \nu} &= 0 \qquad \text{on } \Gamma_N, \end{aligned} \tag{2.1}$$

where

$$\mathcal{L} = -\varepsilon\Delta + \frac{\partial}{\partial x_1}, \tag{2.2}$$

$\varepsilon \equiv \text{const} \in (0;1)$, $\Gamma_N = \{x \equiv (x_1, x_2) : x \in \partial\Omega, x_1 = 1, 0 < x_2 < 1\}$, $\Gamma_D = \partial\Omega \setminus \Gamma_N$, and ν is the outer conormal vector to Γ_N. We assume that the functions g and f are sufficiently smooth, nonnegative, and bounded from above by 0.5. The above restrictions on g and f are dictated only by simplicity of presentation. We introduce the Neumann boundary condition for the sake of generality; the case $\Gamma_N = \emptyset$ is covered by the analysis as well.

Let Ω_h be a square grid with the step size $h = 1/n$, where n is a positive integer. Then the system of the finite difference equations can be presented in the following form:

$$\begin{aligned} \mathcal{L}_h\, u_{ij} &= f_{ij}, \qquad x_{ij} \in \Omega \cup \Gamma_N, \\ u_{ij} &= g_{ij}, \qquad x_{ij} \in \partial\Omega \setminus \Gamma_N, \end{aligned} \tag{2.3}$$

where $x_{ij} = (ih, jh)$, $i, j = 0, \ldots, n$. Here

$$\mathcal{L}_h\, u_{ij} = \begin{cases} \dfrac{\varepsilon}{h^2}\left[4u_{ij} - u_{i+1j} - u_{i-1j} - u_{ij-1} - u_{ij+1}\right] + \dfrac{u_{ij} - u_{i-1j}}{h}, & x_{ij} \in \Omega, \\ \dfrac{\varepsilon}{h^2}\left[3u_{ij} - u_{i-1j} - u_{ij-1} - u_{ij+1}\right] + \dfrac{u_{ij} - u_{i-1j}}{h}, & x_{ij} \in \Gamma_N; \end{cases} \tag{2.4}$$

that is, the convection term is approximated by the upwind finite differences. In what follows, the term "upwind" stands for the opposite direction to the first coordinate axis and "crosswind" stands for the orthogonal direction.

We note that grid operator (2.4) satisfies the well-known maximum principle [16]. Taking into account restrictions on f and g and applying the grid maximum principle we get

$$\max_{x_{ij} \in \bar{\Omega}} u_{ij} \le 1.$$

2.1. Upwind and crosswind error propagation. Let n_a and n_b be positive integers such that $n_a < n_b \le n$. We set $a = n_a \cdot h$ and $b = n_b \cdot h$ and consider a grid problem (Figure 2)

$$\begin{aligned} \mathcal{L}_h\, v_{ij} &= f_{ij} \qquad \text{in } \Omega_b = (0;b)\times(0;1), \\ v_{ij} &= g_{b,j} \qquad \text{on } \Gamma_b = \{x : x_1 = b, 0 < x_2 < 1\}, \\ v_{ij} &= g_{ij} \qquad \text{on } \partial\Omega_b \setminus \Gamma_b, \end{aligned} \tag{2.5}$$

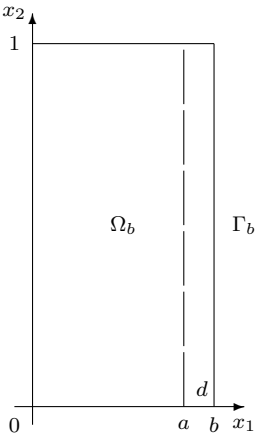


FIG. 2. *Strip subproblem in* Ω_b.

where

$$g_{b,j} = (1 - j \cdot h) \cdot g(b, 0) + j \cdot h \cdot g(b, 1). \tag{2.6}$$

Our goal is to find for given n_a and $\delta \in (0; 1)$ a value of n_b, possibly smaller to satisfy the inequality

$$\max_{\substack{0 \le i \le n_a \\ 0 \le j \le n}} |v_{ij} - u_{ij}| \le \delta. \tag{2.7}$$

LEMMA 2.1. *Let* $d = h \cdot (n_b - n_a)$. *Then* (2.7) *holds true for any* d *which satisfies the inequality*

$$d \ge h \, \frac{\ln(1/\delta)}{\ln q_1}, \qquad q_1 = 1 + \frac{h}{\varepsilon}. \tag{2.8}$$

Proof. The grid function $w_{ij} = v_{ij} - u_{ij}$ is the solution of the grid problem

$$\begin{aligned} \mathcal{L}_h \, w_{ij} &= 0 && \text{in } \Omega_b, \\ w_{ij} &= 0 && \text{on } \partial\Omega \setminus \Gamma_b, \\ w_{ij} &= g_{b,j} - u_{ij} && \text{on } \Gamma_b. \end{aligned}$$

Using the grid maximum principal we can show that

$$|w_{ij}| \le w_i, \qquad x_{ij} \in \bar{\Omega},$$

where the grid function

$$w_i = \frac{q_1^i - 1}{q_1^{n_b} - 1}, \qquad i = 0, \ldots, n_b, \qquad q_1 = 1 + \frac{h}{\varepsilon},$$

is the solution to the finite difference system

$$\frac{\varepsilon}{h^2}[2w_i - w_{i-1} - w_{i+1}] + \frac{1}{h}(w_i - w_{i-1}) = 0, \qquad i = 1, \ldots, n_b - 1,$$
$$w_0 = 0, \qquad w_{n_b} = 1.$$

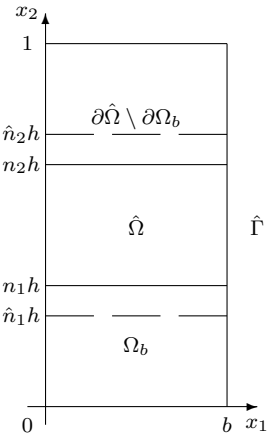


FIG. 3. *Box subproblem.*

Thus,

$$w_i \le q_1^{i-n_b}. \qquad \square$$

Remark 1. We see that in the case $\varepsilon \sim h^\alpha, \alpha > 1$ (that is, $\varepsilon \ll h$), $\ln\left(1+\frac{h}{\varepsilon}\right) \simeq (\alpha-1)\ln h^{-1}$. Therefore, the overlap required by (2.8) is as much as a few mesh steps:

$$d \ge h\frac{\ln(1/\delta)}{(\alpha-1)\ln h^{-1}}.$$

In the opposite case $h \ll \varepsilon$ we rewrite (2.8) as

$$d \ge \varepsilon\,\frac{\ln(1/\delta)}{\ln \kappa_1}, \quad \kappa_1 = (1+h/\varepsilon)^{\varepsilon/h},$$

and note that $\kappa_1 \to e$ as $h/\varepsilon \to 0$. The last estimate coincides with the differential estimate [20]. $\square$

Now, we consider an approximation to the solution of the problem (2.5) by the solution of an auxiliary problem.

We assume that n_1 and n_2, $n_1 \le n_2$, are given nonnegative integers, and $\hat{n}_1$ and $\hat{n}_2$ are also nonnegative integers such that $\hat{n}_1 \le n_1$ and $n_2 \le \hat{n}_2 \le n$. We consider a finite difference problem (Figure 3)

$$\begin{aligned}
\mathcal{L}_h \hat{v}_{ij} &= f_{ij} \qquad \text{in } \hat{\Omega} = (0;b)\times(\hat{n}_1 h; \hat{n}_2 h),\\
\hat{v}_{ij} &= g_{b,j} \qquad \text{on } \hat{\Gamma} = \partial\hat{\Omega} \cap \Gamma_b,\\
\hat{v}_{ij} &= \begin{cases} g_{ij} & \text{on } \partial\hat{\Omega} \cap \partial\Omega_b \setminus \Gamma_b,\\ 1 & \text{on } \partial\hat{\Omega} \setminus \partial\Omega_b. \end{cases}
\end{aligned} \tag{2.9}$$

The grid function $g_{b,j}$ is defined in (2.6).

Our goal is to find for given n_1, n_2, and $\delta \in (0;1)$ values of $\hat{n}_1$ and $\hat{n}_2$ as close as possible to the values of n_1 and n_2, respectively, such that the estimate

$$\max_{\substack{0\le i\le n_b\\ n_1\le j\le n_2}} |v_{ij} - \hat{v}_{ij}| \le \delta \tag{2.10}$$

holds true.

LEMMA 2.2. *To provide estimate* (2.10)*, choose*

$$\begin{aligned} \hat{n}_1 &= \max\{0; n_1 - n_b - [n^*]\}, \\ \hat{n}_2 &= \min\{n; n_2 + n_b + [n^*]\}, \end{aligned} \tag{2.11}$$

where $[n^*]$ *is the integer part of* n^* *and*

$$n^* = \frac{\ln 2/\delta}{\ln q_2}, \qquad q_2 = 1 + \frac{h}{2\varepsilon}. \tag{2.12}$$

Proof. The error grid function $w_{ij} = v_{ij} - \hat{v}_{ij}$ satisfies the equations

$$\begin{aligned} \mathcal{L}_h w_{ij} &= 0 \qquad \text{in } \hat{\Omega}, \\ w_{ij} &= \begin{cases} 0 & \text{on } \partial\hat{\Omega} \cap \partial\Omega_b, \\ v_{ij} - 1 & \text{on } \partial\hat{\Omega} \setminus \partial\Omega_b. \end{cases} \end{aligned}$$

It follows from the grid maximum principle that the solution grid function to (2.5) satisfies the inequality (due to the assumptions $0 \le f \le 0.5$ in Ω and $0 \le g \le 0.5$ on $\partial\Omega$)

$$\max_{\substack{0 \le i \le n_b \\ 0 \le j \le n}} v_{ij} \le 1.$$

Hence,

$$\max_{\partial\hat{\Omega}} |w_{ij}| \le 1.$$

We consider an additional grid problem

$$\begin{aligned} \mathcal{L}_h \hat{w}_{ij} &= 0 \qquad \text{in } \hat{\Omega}, \\ \hat{w}_{ij} &= \begin{cases} 0 & \text{on } \partial\hat{\Omega} \cap \partial\Omega_b, \\ 1 & \text{on } \partial\hat{\Omega} \setminus \partial\Omega_b. \end{cases} \end{aligned} \tag{2.13}$$

It is obvious that $|w_{ij}| \le \hat{w}_{ij}$ in $\hat{\Omega} \cup \partial\hat{\Omega}$.

For the sake of simplicity we consider the situation when $\hat{n}_2 < n$ and $\hat{n}_1 = 0$. Then (2.13) can be presented by

$$\begin{aligned} \mathcal{L}_h \hat{w}_{ij} &= 0 \qquad \text{in } \hat{\Omega}, \\ \hat{w}_{ij} &= 1 \qquad \text{on } \hat{\Gamma}_2 = \{x : 0 < x_1 < b, \qquad x_2 = \hat{n}_2 h\}, \\ \hat{w}_{ij} &= 0 \qquad \text{on } \partial\hat{\Omega} \setminus \hat{\Gamma}_2. \end{aligned} \tag{2.14}$$

We replace (2.14) by the same sort of the finite difference problem in the smaller domain $G = \{x : x \in \hat{\Omega}, x_1 + x_2 < (\hat{n}_2 + 1)h\}$:

$$\begin{aligned} \mathcal{L}_h z_{ij} &= 0 \qquad \text{in } G, \\ z_{ij} &= 0 \qquad \text{on } \partial G \cap \partial\hat{\Omega}, \\ z_{ij} &= 1 \qquad \text{on } \partial G \setminus \partial\hat{\Omega}. \end{aligned} \tag{2.15}$$

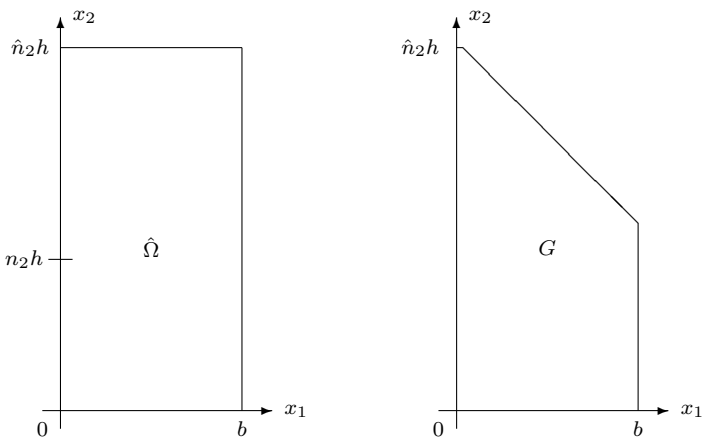


FIG. 4. *An example of domains* $\hat{\Omega}$ *and* G.

According to the grid maximum principle the following estimate holds true:

$$\hat{w}_{ij} \le z_{ij}, \qquad x \in G.$$

An example of the domains $\hat{\Omega}$ and G is presented in Figure 4.

Let us define the values

$$z_l = \max_{i+j=l} z_{ij}, \qquad l = 1, \dots, \hat{n}_2 + 1.$$

We fix some l and write (2.15) for the grid node where $z_l = z_{ij}$. Negative terms we substitute by their low estimates and get from (2.15) the system of inequalities [4]

$$-\frac{2\varepsilon}{h^2}\left[z_{l-1} + z_{l+1} - 2z_l\right] + \frac{z_l - z_{l-1}}{h} \le 0, \qquad l = 2, \dots, \hat{n}_2,$$

with the boundary conditions

$$z_1 = 0, \qquad z_{\hat{n}_2+1} = 1.$$

Consider the system

$$-\frac{2\varepsilon}{h^2}\left[\hat{z}_{l-1} + \hat{z}_{l+1} - 2\hat{z}_l\right] + \frac{\hat{z}_l - \hat{z}_{l-1}}{h} = 0, \qquad l = 2, \dots, \hat{n}_2, \tag{2.16}$$

$$\hat{z}_1 = 0, \qquad \hat{z}_{\hat{n}_2+1} = 1.$$

The grid maximum principle results are

$$z_l \le \hat{z}_l, \qquad l = 1, \dots, \hat{n}_2.$$

Applying the analysis from the proof of Lemma 2.1 to system (2.16) we get estimates

$$\hat{z}_l \le q_2^{-(\hat{n}_2+1-l)}, \qquad l = 1, \dots, \hat{n}_2,$$

where

$$q_2 = 1 + h/2\varepsilon.$$

Thus, to provide estimate (2.10) we can choose the value of $\hat{n}_2$ equal to the integer part of $\hat{n}_2^* + 1$, where $\hat{n}_2^*$ is the solution of the equation

$$\delta/2 = q_2^{-(\hat{n}_2^*+1-n_b-n_2)},$$

while $\hat{n}_1$ is equal to the integer part of $\hat{n}_1^*$, where $\hat{n}_1^*$ is the solution of the equation

$$\delta/2 = q_2^{-(-\hat{n}_1^*-1-n_b+n_1)}. \qquad \square$$

Remark 2. We see that in the case $2\varepsilon \sim h^\alpha, \alpha > 1$ (i.e., $\varepsilon \ll h$), $\ln\left(1+\frac{h}{\varepsilon}\right) \simeq (\alpha-1)\ln h^{-1}$. Therefore, the overlap required by (2.11) is as much as n_b plus a few mesh steps:

$$n^\star \geq \frac{\ln(1/\delta)}{(\alpha-1)\ln h^{-1}}.$$

In the opposite case $h \ll \varepsilon$ we rewrite (2.12),

$$n^* h = 2\varepsilon\,\frac{\ln 2/\delta}{\ln \kappa_2}, \quad \kappa_2 = (1+h/2\varepsilon)^{2\varepsilon/h},$$

and note that $\kappa_2 \to e$ as $h/\varepsilon \to 0$. The width of additional overlap n^*h is proportional to ε (cf. Remark 1) though the overall crosswind overlap is still of order of $n_b h$. Numerical experiments clearly point out that the term $n_b h$ is superfluous. Below we consider two cases when the term $n_b h$ may be omitted. $\square$

In the asymptotic case $\varepsilon \ll h$ estimates (2.11)–(2.12) may be considerably enhanced [7] as shown in the next lemma.

LEMMA 2.3. *Let $\varepsilon \ll h$. To provide estimate* (2.10)*, choose*

$$\begin{aligned} \hat{n}_1 &= \max\{0; n_1 - [n^*]\}, \\ \hat{n}_2 &= \min\{n; n_2 + [n^*]\}, \end{aligned} \tag{2.17}$$

where $[n^*]$ is the integer part of n^* and

$$n^* = \frac{\ln 2/\delta}{\ln \mu}, \quad \mu = \frac{h}{\varepsilon} \gg 1. \tag{2.18}$$

Proof. We take advantage of the initial steps of the proof of Lemma 2.2 and start the analysis of the solution of the problem (2.14).

We fix n_b and denote $p = n_b - 1$ and define a set of the lines $P_l = \{x = (x_1, x_2) : x_1 + p x_2 = l,\ l = 1, 2, \ldots, p(\hat{n}_2 + 1)\}$. Let

$$z_l = \max_{x_{ij} \in P_l} \hat{w}_{ij}, \qquad l = 1, \ldots, p(\hat{n}_2 + 1),$$

where x_{ij} stands for any grid node in $\hat{\Omega}$. Due to the discrete maximum principle

$$z_l = 0,\ l = 1, \ldots, p, \qquad z_l = 1,\ l = p\hat{n}_2 + 1, \ldots, p(\hat{n}_2 + 1). \tag{2.19}$$

From (2.14) we pass to a system of inequalities

$$-\frac{\varepsilon}{h^2}\left[z_{l-p} + z_{l-1} + z_{l+1} + z_{l+p} - 4z_l\right] + \frac{z_l - z_{l-1}}{h} \leq 0, \quad l = p+1, \ldots, p\hat{n}_2,$$

equipped with the boundary conditions (2.19). We consider a solution $\hat{z}_l$ to the system

$$-\frac{\varepsilon}{h^2}\left[\hat{z}_{l-p}+\hat{z}_{l-1}+\hat{z}_{l+1}+\hat{z}_{l+p}-4\hat{z}_l\right]+\frac{\hat{z}_l-\hat{z}_{l-1}}{h}=0, \quad l=p+1,\dots,p\hat{n}_2, \tag{2.20}$$

subjected to

$$\hat{z}_l=0,\ l=1,\dots,p, \qquad \hat{z}_l=1,\ l=p\hat{n}_2+1,\dots,p(\hat{n}_2+1). \tag{2.21}$$

Due to the discrete maximum principle

$$z_l\le\hat{z}_l, \qquad l=p+1,\dots,p\hat{n}_2.$$

Let R_i, $i=1,\dots,2p$, be the roots of the polynomial

$$Q(\hat{z})=-1-(\mu+1)\hat{z}^{p-1}+(\mu+4)\hat{z}^p-\hat{z}^{p+1}-\hat{z}^{2p}$$

with $\mu\equiv\frac{h}{\varepsilon}$. We will show that for the case $\mu\gg1$

$$|R_{2p}|=\max_{1\le i\le 2p}|R_i|\approx\mu^{\frac{1}{p}}.$$

Indeed, the formal asymptotic expansion yields

$$\mu|R_{2p}|^p\approx|R_{2p}|^{2p}$$

or

$$\mu^{p\beta+1}\approx\mu^{2p\beta}, \tag{2.22}$$

provided that

$$|R_{2p}|\approx\mu^{\beta}. \tag{2.23}$$

(2.22) and (2.23) result in

$$|R_{2p}|\approx\mu^{\frac{1}{p}}, \quad \mu\gg1. \tag{2.24}$$

To justify the formal asymptotic expansion, we analyze $|\,|R_{2p}|-\mu^{\frac{1}{p}}\,|$ for $p>2$, $\mu\gg1$:

$$|\,|R_{2p}|-\mu^{\frac{1}{p}}\,|\le C\frac{|Q(R_{2p})-Q(\mu^{\frac{1}{p}})|}{|Q'(\mu^{\frac{1}{p}})|}\le C\frac{|0+\mu^{2-\frac{1}{p}}|}{|-p\mu^{2-\frac{1}{p}}|}\le C\frac{1}{p}.$$

Thus, (2.24) is valid.

To evaluate $\hat{z}_l$, we consider the sequence

$$\tilde{z}_l=\frac{R_{2p}^l+\bar{R}_{2p}^l-(R_{2p}+\bar{R}_{2p})}{R_{2p}^{p\hat{n}_2+1}+\bar{R}_{2p}^{p\hat{n}_2+1}-(R_{2p}+\bar{R}_{2p})}, \quad l=1,\dots,p(\hat{n}_2+1).$$

$\tilde{z}_l$ are real numbers and satisfy to (2.20), while

$$\tilde{z}_1=0,\ \tilde{z}_l>0,\ l=2,\dots,p, \quad \tilde{z}_{p\hat{n}_2+1}=1,\ \tilde{z}_l>1,\ l=p\hat{n}_2+2,\dots,p(\hat{n}_2+1).$$

For $e_l\equiv\tilde{z}_l-\hat{z}_l$ we have

$$-\frac{\varepsilon}{h^2}\left[e_{l-p}+e_{l-1}+e_{l+1}+e_{l+p}-4e_l\right]+\frac{e_l-e_{l-1}}{h}=0,\quad l=p+1,\dots,p\hat{n}_2,$$

$$e_1=0,\ e_l>0,\ l=2,\dots,p,\qquad e_{p\hat{n}_2+1}=0,\ e_l>0,\ l=p\hat{n}_2+2,\dots,p(\hat{n}_2+1).$$

Due to the discrete maximum principle

$$e_l\geq 0,\quad l=p+1,\dots,p\hat{n}_2,$$

that is to say

$$\hat{z}_l\leq\tilde{z}_l,\quad l=p+1,\dots,p\hat{n}_2.$$

Hence,

$$\hat{z}_l\leq\delta$$

when

$$\frac{R_{2p}^l+\bar{R}_{2p}^l-(R_{2p}+\bar{R}_{2p})}{R_{2p}^{p\hat{n}_2+1}+\bar{R}_{2p}^{p\hat{n}_2+1}-(R_{2p}+\bar{R}_{2p})}\leq\delta. \tag{2.25}$$

Since $|R_{2p}|\approx\mu^{\frac{1}{p}}\gg 1$, (2.25) is equivalent to

$$\mu^{\frac{l-(p\hat{n}_2+1)}{p}}\leq\delta$$

or

$$\frac{l-1}{p}\leq\hat{n}_2-\frac{\ln\delta^{-1}}{\ln\mu}.$$

Taking into account the slope of the lines P_l we get the estimate

$$\hat{w}_{ij}\leq\delta,\qquad j<\hat{n}_2-\frac{\ln\delta^{-1}}{\ln\mu}.\qquad\square$$

For certain grids the values $h(\hat{n}_1-n_1)$, $h(n_2-\hat{n}_2)$ may be reduced without assumption $\varepsilon\ll h$, estimate (2.10) being still valid. Consider $\hat{\Omega}=(0,b)\times(0,c)$ and a rectangular grid with mesh steps $h_1=b/n_{x_1}$, $h_2=c/n_{x_2}$ and the grid problem

$$\begin{aligned}\bar{\mathcal{L}}_h\,\hat{w}_{ij}&=0 &&\text{in }\hat{\Omega},\\ \hat{w}_{ij}&=1 &&\text{on }\hat{\Gamma}_2=\{x:0<x_1<b,\ x_2=c\},\\ \hat{w}_{ij}&=0 &&\text{on }\partial\hat{\Omega}\setminus\hat{\Gamma}_2,\end{aligned}$$

where

$$\bar{\mathcal{L}}_h\,\hat{w}_{ij}=\frac{\varepsilon}{h_1^2}\left[2\hat{w}_{ij}-\hat{w}_{i+1j}-\hat{w}_{i-1j}\right]+\frac{\varepsilon}{h_2^2}\left[2\hat{w}_{ij}-\hat{w}_{ij+1}-\hat{w}_{ij-1}\right]+\frac{\hat{w}_{ij}-\hat{w}_{i-1j}}{h_1}.$$

LEMMA 2.4. *There exists ratio h_2/h_1 such that for h_1 sufficiently small*

$$\begin{aligned}&\hat{w}_{ij}\leq\delta\ \textit{on}\ \Omega_\delta=(0,b)\times(0,c_\delta),\\ &c_\delta=c-2\sqrt{\varepsilon b\ln 1/\delta}.\end{aligned} \tag{2.26}$$

Proof (see [7]). By analogy with the proof of Lemma 2.2 we get

$$\hat{w}_{ij} \le \delta \text{ on } \Omega_\delta = (0,b) \times (0, c_\delta),$$
$$c_\delta = c - h_2 \frac{\ln 1/\delta}{\ln q} - b\frac{h_2}{h_1},\ q = 1 + \frac{h_1}{\varepsilon(h_1^2/h_2^2 + 1)}.$$

We can maximize Ω_δ by optimizing c_δ provided that h_1 is small. It is maximal when $h_2 = h_1\sqrt{\frac{\varepsilon \ln 1/\delta}{b}}$:

$$c_\delta = c - 2b\frac{h_2}{h_1} = c - 2\sqrt{\varepsilon b \ln 1/\delta}. \qquad \square$$

We see that for specific grids one can satisfy (2.10) with an extension $2\sqrt{\varepsilon b \ln 2/\delta}$. Since we can choose h_1 vanishing we come to the conclusion that the differential problem has the same property. Consequently, up to a pointwise approximation error, the discrete problem posed on square grids yields (2.26) as well. In particular, (2.26) is valid in the case $h << \varepsilon$. For the case of $b \approx 1$ results similar to (2.26) are reported in [11, 15], where authors dealt with the streamline upwind Petrov–Galerkin (SUPG) discretization.

2.2. Two-level Schwarz method. Let $\{n_{a,k}\}$ and $\{n_{b,k}\}$ be strongly monotone sequences of integers such that $0 = n_{a,1} < n_{a,2} < \cdots < n_{a,m} < n$ and $n_{a,k+1} < n_{b,k} \le n$, $k = 1, \ldots, m-1$, $n_{b,m} = n$, $a_k = n_{a,k} \cdot h$, $a_{m+1} = 1$, and $b_k = n_{b,k} \cdot h$, $k \le n$. We partition the domain $\Omega = (0;1) \times (0;1)$ into m overlapping strip subdomains $\Omega_k = (a_k; b_k) \times (0;1)$ and each strip Ω_k we partition into p overlapping subdomains $\Omega_{k,s} = (a_k; b_k) \times (\hat{n}_1^{(s)}h; \hat{n}_2^{(s)}h)$ (Figure 5). Here $\hat{n}_1^{(s)}$ and $\hat{n}_2^{(s)}$ are chosen for given $n_1^{(s)}$ and $n_2^{(s)}$, respectively, using the formulae (2.11)–(2.12).

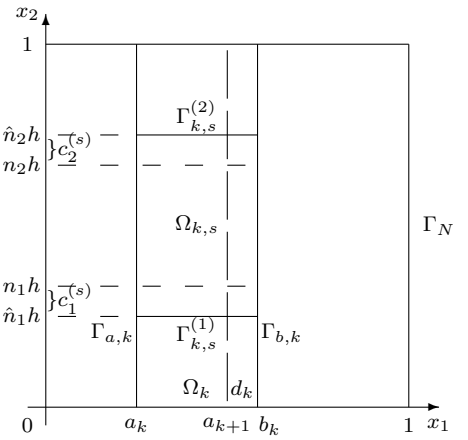


FIG. 5. *Two-level domain decomposition.*

Let u_{ij}^{t-1} be a current guess to u_{ij}. A new approximation u_{ij}^t to the solution grid function u_{ij} of the problem (2.3) will be found with the multiplicative Schwarz method based on the two-level overlapping domain decomposition

$$\Omega = \bigcup_{k=1}^{m} \bigcup_{s=1}^{p} \Omega_{k,s}. \tag{2.27}$$

One complete iteration step includes solving a sequence of subproblems in the strip subdomains Ω_k, $k = 1, \ldots, m$, (outer Schwarz loop) and each kth strip subproblem consists of p subproblems in the subdomains $\Omega_{k,s}$, $s = 1, \ldots, p$. In order to specify the inner iterates within each strip subdomain we introduce the following notations.

Let

$$\Gamma_{a,k} = \{x,\ x_1 = a_k,\ 0 < x_2 < 1\},\ \Gamma_{b,k} = \{x,\ x_1 = b_k,\ 0 < x_2 < 1\}, \quad k = 1, \ldots, m,$$

$$\Gamma_{k,s}^{(1)} = \{x : a_k < x_1 < b_k, x_2 = \hat{n}_1^{(s)} h\},\ \Gamma_{k,s}^{(2)} = \{x : a_k < x_1 < b_k, x_2 = \hat{n}_2^{(s)} h\}, \tag{2.28}$$

and let the value of k be fixed ($k \geq 1$) and u_{ij}^t be given on $\Gamma_{a,k}$ ($u_{ij}^t = g_{ij}$ on $\Gamma_{a,1}$). For $k < m$ we denote the subproblem

$$\begin{aligned} \mathcal{L}_h \hat{u}_{ij}^{(s)} &= f_{ij} && \text{in } \Omega_{k,s}, \\ \hat{u}_{ij}^{(s)} &= u_{ij}^t && \text{on } \partial\Omega_{k,s} \cap \Gamma_{a,k}, \\ \hat{u}_{ij}^{(s)} &= u_{ij}^{t-1} && \text{on } \partial\Omega_{k,s} \setminus \Gamma_{a,k}, \end{aligned}$$

by

$$\mathcal{A}_{k,s}^t \hat{u}_{ij}^{(s)} = f_{ij},$$

and assuming $\hat{u}_{ij}^{(s-1)}, \hat{u}_{ij}^{(s+1)}$ are given on $\Gamma_{k,s}^{(1)}, \Gamma_{k,s}^{(2)}$, we denote the subproblem

$$\begin{aligned} \mathcal{L}_h \hat{u}_{ij}^{(s)} &= f_{ij} && \text{in } \Omega_{k,s}, \\ \hat{u}_{ij}^{(s)} &= u_{ij}^t && \text{on } \partial\Omega_{k,s} \cap \Gamma_{a,k}, \\ \hat{u}_{ij}^{(s)} &= u_{ij}^{t-1} && \text{on } \partial\Omega_{k,s} \cap \Gamma_{b,k}, \\ \hat{u}_{ij}^{(s)} &= \hat{u}_{ij}^{(s-1)} && \text{on } \Gamma_{k,s}^{(1)},\ s > 1, \\ \hat{u}_{ij}^{(s)} &= \hat{u}_{ij}^{(s+1)} && \text{on } \Gamma_{k,s}^{(2)},\ s < p, \end{aligned}$$

by

$$\mathcal{M}_{k,s}^t \hat{u}_{ij}^{(s)} = f_{ij}.$$

For $k = m$ we substitute the nonhomogeneous Dirichlet boundary condition on $\Gamma_{b,m}$ by the homogeneous Neumann boundary condition on $\Gamma_{b,m} = \Gamma_N$.

Now we are able to state the two-stage Schwarz method (Figure 6).

ALGORITHM. Given the current guess u_{ij}^{t-1} and positive integers T_k, $k = 1, \ldots, m$:

For $k = 1, \ldots, m$ find the restriction of u_{ij}^t onto $(a_k; a_{k+1}] \times (0; 1)$ by the following inner iterations:

1. Set $l = 0$. For all even s solve $\mathcal{A}_{k,s}^t \hat{u}_{ij}^{(s)} = f_{ij}$.
2. Do while ($l \leq T_k$)
   Set $l = l + 1$.
   If l is odd then
   for all odd s solve $\mathcal{M}_{k,s}^t \hat{u}_{ij}^{(s)} = f_{ij}$
   else
   for all even s solve $\mathcal{M}_{k,s}^t \hat{u}_{ij}^{(s)} = f_{ij}$
   End do

3. Define the grid function u_{ij}^t on $[a_k, a_{k+1}] \times (0;1)$ by

$$u_{ij}^t = \hat{u}_{ij}^{(s)}, \quad x \in [a_k, a_{k+1}] \times [n_1^{(s)}h; n_2^{(s)}h], \quad s = 1, \ldots, p.$$

The above algorithm is rather flexible. By varying the value of T_k one can formulate different types of the approximate solvers in the crosswind strips Ω_k:

- Different T_k generate different numbers of the multiplicative Schwarz iterations between the subdomains $\Omega_{k,s}$ with odd and even indexes s.
- Instead of the condition $l > T_k$ the stopping criterion might be a decrease of the residual for the strip problem.
- Additive solvers may be formulated:
  1. For all s solve $\mathcal{A}_{k,s}^t \hat{u}_{ij}^{(s)} = f_{ij}$.
  2. Define the grid function u_{ij}^t on $[a_k, a_{k+1}] \times (0;1)$ by

$$u_{ij}^t = \hat{u}_{ij}^{(s)}, \quad x \in [a_k, a_{k+1}] \times [n_1^{(s)}h; n_2^{(s)}h], \quad s = 1, \ldots, p.$$

To evaluate the convergence rate of the above multiplicative Schwarz method, we have to estimate the maximum norm of the grid function $u_{ij} - u_{ij}^t$. For simplicity we present the result related to the multiplicative version with $T_k = 1$, $k = 1, \ldots, m$. Hereinafter, we denote by d_k the downwind overlap $b_k - a_{k+1}$ and by $c_1^{(s)}$, $c_2^{(s)}$ the crosswind overlaps $(\hat{n}_1^{(s)} - n_1^{(s)})h$, $(n_2^{(s)} - \hat{n}_2^{(s)})h$, respectively, and define b as $\max\limits_{1 \le k \le m} (b_k - a_k)$.

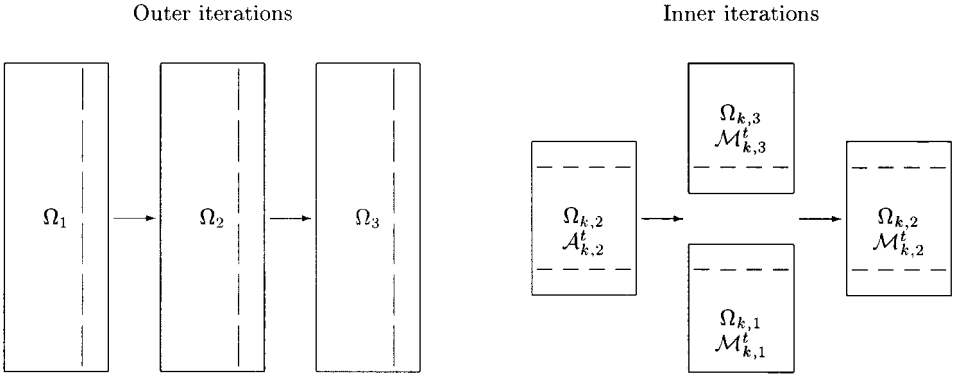


FIG. 6. *Two-stage Schwarz method.*

THEOREM 2.5. *Let* $d \equiv \min\limits_{1 \le k \le m-1} d_k$,

$$c_1^{(s)} \equiv c_2^{(s)} = b + \left[\frac{d}{h} \frac{\ln q_1}{\ln q_2} \right] h, \quad q_1 = 1 + \frac{h}{\varepsilon}, \quad q_2 = 1 + \frac{h}{2\varepsilon}, \tag{2.29}$$

and let a nonnegative grid function u_{ij}^{t-1} *be given in* $\bar{\Omega}$ *such that*

$$\max_{\Omega_h} |u_{ij} - u_{ij}^{t-1}| \le 1. \tag{2.30}$$

Then

$$\max_{\Omega_h} |u_{ij} - u_{ij}^t| \le 3m q_1^{-d/h} \max_{\Omega_h} |u_{ij} - u_{ij}^{t-1}|. \tag{2.31}$$

Proof. Let $r_{t-1} = \max_{\Omega_h} |u_{ij} - u_{ij}^{t-1}|$. To estimate $|u_{ij} - u_{ij}^{t}|$, we consider the following strip equations $(1 \le k < m)$(Figure 5):

$$\begin{cases} \mathcal{L}_h w_{ij}^{(k,s)} &= 0 & \text{in } \Omega_{k,s},\ s \textbf{ even}, \\ w_{ij}^{(k,s)} &= \delta_{k-1} & \text{on } \partial\Omega_{k,s} \cap \Gamma_{a,k}, \\ w_{ij}^{(k,s)} &= 0 & \text{on } \partial\Omega_{k,s} \cup \partial\Omega, \\ w_{ij}^{(k,s)} &= r_{t-1} & \text{on } \partial\Omega_{k,s} \setminus (\Gamma_{a,k} \cup \partial\Omega), \end{cases}$$

$$\begin{cases} \mathcal{L}_h w_{ij}^{(k,s)} &= 0 & \text{in } \Omega_{k,s},\ s \textbf{ odd}, \\ w_{ij}^{(k,s)} &= \delta_{k-1} & \text{on } \partial\Omega_{k,s} \cap \Gamma_{a,k}, \\ w_{ij}^{(k,s)} &= r_{t-1} & \text{on } \partial\Omega_{k,s} \cap \Gamma_{b,k}, \\ w_{ij}^{(k,s)} &= w_{ij}^{(k,s-1)} & \text{on } \Gamma_{k,s}^{(1)}, \\ w_{ij}^{(k,s)} &= w_{ij}^{(k,s+1)} & \text{on } \Gamma_{k,s}^{(2)}, \end{cases}$$

where $\Gamma_{k,s}^{(1)}$ and $\Gamma_{k,s}^{(2)}$ are defined in (2.28), $\delta_0 = 0$, $\delta_{k-1} \equiv \text{const} \in (0;1)$. Let us fix the value of $k, 1 \le k < m$, and introduce the grid function $w_{ij}^{(k)}$ as the solution of the system

$$\begin{cases} \mathcal{L}_h w_{ij}^{(k)} &= 0 & \text{in } \Omega_k, \\ w_{ij}^{(k)} &= \delta_{k-1} & \text{on } \Gamma_{a,k}, \\ w_{ij}^{(k)} &= r_{t-1} & \text{on } \Gamma_{b,k}, \\ w_{ij}^{(k)} &= 0 & \text{on } \partial\Omega \setminus (\Gamma_{a,k} \cup \Gamma_{b,k}), \end{cases}$$

where $\Omega_k, \Gamma_{a,k}$, and $\Gamma_{b,k}$ are defined as in (2.28).

Then the grid functions

$$\tilde{w}_{ij}^{(k,s)} = w_{ij}^{(k,s)} - w_{ij}^{(k)}, \quad s = 1, \dots, p,$$

satisfy the equations

$$\begin{cases} \mathcal{L}_h \tilde{w}_{ij}^{(k,s)} &= 0 & \text{in } \Omega_{k,s},\ s \textbf{ even}, \\ \tilde{w}_{ij}^{(k,s)} &= 0 & \text{on } \partial\Omega_{k,s} \cap \partial\Omega_k, \\ \tilde{w}_{ij}^{(k,s)} &= r_{t-1} - w_{ij}^{(k)} & \text{on } \partial\Omega_{k,s} \setminus \partial\Omega_k, \end{cases}$$

$$\begin{cases} \mathcal{L}_h \tilde{w}_{ij}^{(k,s)} &= 0 & \text{in } \Omega_{k,s},\ s \textbf{ odd}, \\ \tilde{w}_{ij}^{(k,s)} &= 0 & \text{on } \partial\Omega_{k,s} \cap \partial\Omega_k, \\ \tilde{w}_{ij}^{(k,s)} &= \tilde{w}_{ij}^{(k,s-1)} & \text{on } \Gamma_{k,s}^{(1)},\ s > 1, \\ \tilde{w}_{ij}^{(k,s)} &= \tilde{w}_{ij}^{(k,s+1)} & \text{on } \Gamma_{k,s}^{(2)} r,\ s < p. \end{cases}$$

By virtue of the fact that

$$b + \left[\frac{d}{h}\frac{\ln q_1}{\ln q_2}\right] h = b + \left[\frac{\ln \frac{1}{q_1^{-d/h}}}{\ln q_2}\right] h$$

and Lemma 2.2, the values of $\hat{n}_1^{(s)}$ and $\hat{n}_2^{(s)}$ are chosen to satisfy the estimates

$$\max_{\text{even } s} \max_{\substack{a_k \le x_{1,i} \le b_k \\ n_1^{(s)} \cdot h \le x_{2,j} \le n_2^{(s)} \cdot h}} |\tilde{w}_{ij}^{(k,s)}| \le 2q_1^{-d/h} r_{t-1}.$$

Then, using the maximum principle we get the estimation

$$\max_{\text{odd } s} \max_{\substack{a_k \le x_{1,i} \le b_k \\ n_1^{(s)} \cdot h \le x_{2,j} \le n_2^{(s)} \cdot h}} |\tilde{w}_{ij}^{(k,s)}| \le 2q_1^{-d/h} r_{t-1}.$$

Based on the discrete maximum principle and Lemma 2.1 one derives the estimation for $w_{ij}^{(k)}$:

$$\max_{\substack{a_k \le x_{1,i} \le a_{k+1} \\ 0 \le x_{2,j} \le 1}} |w_{ij}^{(k)}| \le \delta_{k-1} + q_1^{-d/h} r_{t-1}.$$

Hence,

$$\max_{\substack{a_k \le x_{1,i} \le a_{k+1} \\ n_1^{(s)} \cdot h \le x_{2,j} \le n_2^{(s)} \cdot h}} |w_{ij}^{(k,s)}| \le \delta_{k-1} + 3q_1^{-d/h} r_{t-1}, \quad s = 1, \ldots, p,$$

and

$$\max_{\substack{a_k \le x_{1,i} \le a_{k+1} \\ 0 \le x_{2,j} \le 1}} |u_{ij} - u_{ij}^t| \le \delta_{k-1} + 3q_1^{-d/h} r_{t-1}$$

hold true.

By recursion we have

$$\delta_k \le \delta_{k-1} + 3q_1^{-d/h} r_{t-1}$$

and

$$\delta_{k-1} \le 3(k-1) {q_1}^{-d/h} r_{t-1}, \quad k = 1, \ldots, m. \tag{2.32}$$

Applying the above technique and (2.32) for each strip, we have

$$\max_{\Omega_h} |u_{ij} - u_{ij}^t| \le 3mq_1^{-d/h} r_{t-1}. \qquad \square$$

COROLLARY 2.6. *Let* $\varepsilon \ll h$, $d \equiv \min\limits_{1 \le k \le m-1} d_k$,

$$c_1^{(s)} \equiv c_2^{(s)} = d, \tag{2.33}$$

and let a nonnegative grid function u_{ij}^{t-1} *be given in* $\bar{\Omega}$ *such that*

$$\max_{\Omega_h} |u_{ij} - u_{ij}^{t-1}| \le 1. \tag{2.34}$$

Then

$$\max_{\Omega_h} |u_{ij} - u_{ij}^t| \le 3mq_1^{-d/h} \max_{\Omega_h} |u_{ij} - u_{ij}^{t-1}|, \quad q_1 = \frac{h}{\varepsilon}. \tag{2.35}$$

Remark 3. A consequence of (2.35) is that in the case $\varepsilon \ll h$ the overlap between subdomains $\Omega_{k,s}$ of order of h provides a very good value of the damping factor $\frac{\max_{\Omega_h} |u_{ij}-u_{ij}^t|}{\max_{\Omega_h} |u_{ij}-u_{ij}^{t-1}|}$. □

3. Convection-diffusion problem with a variable transport vector. We proceed to the problem (2.1) with a variable convection:

$$\mathcal{L} = -\varepsilon\Delta + B^1(x_1, x_2) \cdot \frac{\partial}{\partial x_1} + B^2(x_1, x_2) \cdot \frac{\partial}{\partial x_2}. \tag{3.1}$$

We assume that the functions B^1 and B^2 are sufficiently smooth, their moduli being bounded from above by 1, and $B^1(1, x_2) > 0$. The discrete problem can be written in the form (2.3), where $\mathcal{L}_h$ stands for the upwind finite differences counterpart of $\mathcal{L}$. We note that having in mind the Navier–Stokes applications we confine our consideration by the transport vectors $\vec{B} = (B^1, B^2)^T$ typical for the channel problems. In these problems the flow is almost unidirectional and the leading component of $\vec{B}$ dominates the others everywhere except in recirculation zones. We will exploit this assumption in what follows.

We partition the domain $\Omega = (0; 1) \times (0; 1)$ into m overlapping strip subdomains $\Omega_k = (a_k; b_k) \times (0; 1)$ such that the transport vector $\vec{B}$ in the subdomains Ω_k, $k = 1, \ldots, m$, satisfies one of two conditions: either

$$B_{ij}^1 \ge B_0 > 0, \quad -B_{ij}^2 \le R \cdot B_0 \text{ in } \Omega_k, \tag{3.2}$$

with certain constants $0 < R < 1$, $0 < B_0$, or

$$B_{ij}^1 \ge B_0 > 0, \text{ in } \Omega_k \setminus (a_k; a_{k+1}) \times (0; 1). \tag{3.3}$$

The first condition (3.2) corresponds to the case when the transport vector has small deviations from the prescribed downwind direction. The second condition (3.3) corresponds to the more general case when the B^1-component of $\vec{B}$ is positive and bounded from below by a positive constant B_0 on the interface $\Gamma_{b,k}$ and as a consequence, in the rightmost narrow part of Ω_k, but $\vec{B}$ has large deviations from the prescribed downwind direction in Ω_k. In line with the above assumption, we split the set of all the indexes $k = 1, \ldots, m$, into two parts, $\mathcal{P}_s$ and $\mathcal{P}_v$, such that if (3.2) holds true in Ω_k, then $k \in \mathcal{P}_s$, otherwise $k \in \mathcal{P}_v$. Each strip Ω_k, $k = 1, \ldots, m$, we partition into p overlapping subdomains $\Omega_{k,s} = (a_k; b_k) \times (\hat{n}_1^{(s)} h; \hat{n}_2^{(s)} h)$, where $\hat{n}_1^{(s)}$ and $\hat{n}_2^{(s)}$ have to be chosen for given $n_1^{(s)}$ and $n_2^{(s)}$, respectively.

The two-level Schwarz algorithm is no different from the method presented in the previous section. We adapt only the sequence T_k governing the iterations. Let T be a large positive integer. We define the sequence T_k by

$$T_k = \begin{cases} T, & k \in \mathcal{P}_v, \\ 1, & k \in \mathcal{P}_s. \end{cases}$$

To evaluate the convergence rate of the above multiplicative Schwarz method, we assume that $T \gg 1$, which is to say that the errors due to the iterative solvers in

the strips Ω_k, $k \in \mathcal{P}_v$, are negligible. Hereafter, we denote by $\#\mathcal{P}_v$ the number of elements in $\mathcal{P}_v$ and redefine b as $\max_{k\in\mathcal{P}_s}(b_k - a_k)$.

THEOREM 3.1. *Let $\varepsilon \ll h$, $T \gg 1$, and let a nonnegative grid function u_{ij}^{t-1} be given in $\bar{\Omega}$ such that*

$$\max_{\Omega_h} |u_{ij} - u_{ij}^{t-1}| \le 1. \tag{3.4}$$

Let there exist such positive constants B_0 and $R < 1$ that in Ω_k, $k \in \mathcal{P}_v$, $\vec{B}$ satisfies (3.3) *and*

$$\min_{k\in\mathcal{P}_v} d_k \ge h + h\frac{\ln 2\#\mathcal{P}_v}{\ln q_3},\ q_3 = 1 + \frac{B_0 h}{\varepsilon},$$

and in Ω_k, $k \in \mathcal{P}_s$, $\vec{B}$ satisfies (3.2). *If*

$$d = \min_{k\in\mathcal{P}_s} d_k \ge h \cdot \frac{\ln 6m}{\ln q_3}, \tag{3.5}$$

and

$$c_2^{(s)} = c_1^{(s)} \ge b + \left[\frac{d}{h}\frac{\ln q_3}{\ln q_4}\right] h, \qquad q_4 = \frac{1 + \frac{B_0 h}{2\varepsilon}}{1 + R \cdot \frac{B_0 h}{2\varepsilon}}, \tag{3.6}$$

then

$$\max_{\Omega_h} |u_{ij} - u_{ij}^{t}| \le Q \max_{\Omega_h} |u_{ij} - u_{ij}^{t-1}|, \tag{3.7}$$

with certain constant $Q < 1$.

The proof is similar to the proof of the Theorem 2.5 and is based on the following lemmas [7].

LEMMA 3.2. *Consider the problem* (2.5)–(2.6). *Let $B_{ij}^1 \ge B_0 > 0$ in Ω_b. Then* (2.7) *holds true for any $d = h \cdot (n_b - n_a)$ which satisfies the inequality*

$$d \ge h\ \frac{\ln(1/\delta)}{\ln q_3}, \quad q_3 = 1 + \frac{B_0 h}{\varepsilon}. \tag{3.8}$$

LEMMA 3.3. *Consider the problem* (2.9). *Let*

$$B_{ij}^1 \ge B_0 > 0, \quad -B_{ij}^2 \le R \cdot B_0 \text{ in } \hat{\Omega}, \tag{3.9}$$

with a certain constant $0 < R < 1$. To provide estimate (2.10), *choose*

$$\begin{aligned} \hat{n}_1 &= \max\{0; n_1 - n_b - [n^*]\}, \\ \hat{n}_2 &= \min\{n; n_2 + n_b + [n^*]\}, \end{aligned} \tag{3.10}$$

where $[n^*]$ is the integer part of n^* and

$$n^* = \frac{\ln 2/\delta}{\ln q_4}, \quad q_4 = \frac{1 + \frac{B_0 h}{2\varepsilon}}{1 + R \cdot \frac{B_0 h}{2\varepsilon}}. \tag{3.11}$$

LEMMA 3.4. *Consider a grid problem*

$$\begin{aligned} \mathcal{L}_h\, v_{ij} &= 0 \qquad \text{in } \Omega_b = (0;b)\times(0;1), \\ v_{ij} &= 1 \qquad \text{on } \Gamma_b = \{x : x_1 = b, 0 < x_2 < 1\}, \\ v_{ij} &= 0 \qquad \text{on } \partial\Omega_b \setminus \Gamma_b. \end{aligned} \tag{3.12}$$

Let $\varepsilon \ll h$, $b = n_b h$, $\Omega_a = (0; n_a h)\times(0;1)$, *where* $0 < n_a < n_b$. *Furthermore, let*

$$|B^1_{ij}| \le 1,\ |B^2_{ij}| \le 1, \quad 0 < i < n_b,\ 0 < j < n, \tag{3.13}$$

while in the complement of Ω_a *to* Ω_b,

$$B^1_{ij} \ge B_0 > 0, \qquad n_a < i < n_b,\ 0 < j < n. \tag{3.14}$$

Then

$$\max_{\substack{0\le i\le n_a \\ 0\le j\le n}} |v_{ij}| \le \delta \tag{3.15}$$

holds true for any n_b *which satisfies the inequality*

$$n_b \ge n_a + 1 + \frac{\ln \delta^{-1}}{\ln q_5}, \quad q_5 = 1 + \frac{B_0 h}{\varepsilon}. \tag{3.16}$$

Remark 4. The convergence result of Theorem 3.1 is worse than that of Theorem 2.5: the damping factor in (3.7) is not estimated as in (2.31). The reasons are possible crosswind error propagations and nonemptiness of $\mathcal{P}_v$. □

Remark 5. The condition $\varepsilon \ll h$ may be substituted by $\max_{k\in\mathcal{P}_v}(a_{k+1} - a_k)/h \gg 1$, $h/\varepsilon > p_0 > 0$, Lemma 3.4 and Theorem 3.1 being still valid [7]. This means that fast convergence is assured either in the asymptotic case or in the case of wide enough strips Ω_k, $k \in \mathcal{P}_v$. □

4. Computer realization. As follows from Theorem 2.5 and Theorem 3.1, three decomposition parameters are the crucial factors affecting the convergence of the method. These are the crosswind overlaps $c_1^{(s)}$, $c_2^{(s)}$, the downwind overlap d_k of the decomposition into overlapping subdomains, and the position of interfaces. In the case of the variable transport vector the influence of the crosswind overlaps may be reduced by applying $T_k > 1$ Schwarz iterations within each crosswind strip instead of one iteration. In our experiments, even two multiplicative Schwarz iterations compensate well the deviations of B^2 from 0 and provide a strong crosswind damping factor. Hereinafter, we shall apply $T_k = 2$ Schwarz iterations within each crosswind strip Ω_k, $k \in \mathcal{P}_s$. However, the decomposition into overlapping crosswind strips Ω_k has to be adapted to the transport vector $\vec{B}$ in order to provide fast upwind error damping at each fractional step of the global outer Schwarz iteration. To be more precise, the trace of B^1 is bound to be strictly positive on the artificial interfaces: $B^1_{ij} \ge B_0 > 0$. This ensures the fast convergence of the method.

It remains now to present an adaptive technique providing the above decomposition. Given a transport vector $\vec{B} = (B^1, B^2)$, one estimates the fractional damping factors F_k associated with the strips Ω_k, $k = 1, \dots, m-1$. For each $F_k > \frac{1}{m-1}\sum_{i=1}^{m-1} F_i$, one merges the strips Ω_k and Ω_{k+1} and gets new decomposition into crosswind overlapping strips for which one applies the two-level Schwarz method with $T_k = 2, k \in \mathcal{P}_s$, $T_k = T \gg 1, k \in \mathcal{P}_v$.

The structure of the algorithm and features of its adaptive counterpart pose natural restrictions to the transport vector $\vec{B}$. First, the major part of the computational domain has to be occupied by the transport field $\vec{B}$ with the dominant direction coincident with the x_1-axis (for high convergence of the outer Schwarz iterations). Second, the remaining part of Ω, being projected to the x_1-axis, forms a set with a small one-dimensional measure (for minimizing the cost of many iterates in the strips $\Omega_k, k \in \mathcal{P}_v$). For instance, this is the case of a presence of several isolated strong vortices in a global downwind flow.

Now we discuss briefly the parallel implementation of the algorithm. Although it is possible to pipeline the Schwarz fractional steps associated with the global downwind propagation, we do not consider this opportunity since we deal with the fast convergent algorithm which makes the pipelining inefficient. In spite of the sequential realization of the downwind computations, we can parallelize the algorithm when solving the crosswind subproblems. More precisely, the i-processor treats subdomains $\Omega_{k,2i-1}$, $\Omega_{k,2i}$, and exchanges with the processors $i-1$, $i+1$, by the solution traces on $\partial\Omega_{k,s}$, $s = 2i-2, 2i+1$. In practical situations, instead of choosing certain large integers T one could perform iterations until the residual associated with the discrete problem in Ω_k reduces by a factor of 10. In our numerical experiments we fixed the number of iterations $T_k = T = 14$, $k \in \mathcal{P}_v$.

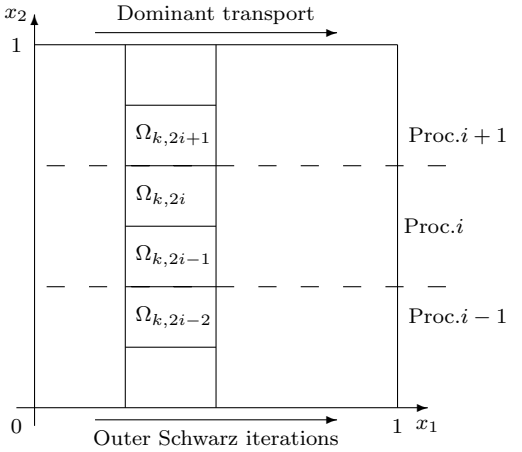


FIG. 7. *Data allocation in parallel implementation.*

Thus, given $P = p/2$ processors, the i-processor deals with data associated with the downwind strip $\cup_{k=1}^{m}$ $(\Omega_{k,2i-1} \cup \Omega_{k,2i})$, $i = 1, \ldots, P$ (Figure 7). We solve the subdomains problems exactly, taking advantage of a factorization technique. We motivate factorizing the respective matrices by the nature of the domain decomposition algorithm: first, we factorize matrices only at the stage of initialization; second, the algorithm is well suited to computation on a massively parallel computer which enables us to reduce considerably both the dimensions of the subproblems and the natural bandwidth of the matrices; third, with factorization we avoid the problem of an optimal stop criterion for any iterative solver. Since we solve the grid subproblems in $\Omega_{k,s}$ by a factorization technique, for the same grid and the same overlap the increase of P would decrease the arithmetical complexity of the global Schwarz iter-

ation. At the same time, with P growing the communications time does not vanish due to the latency time. These conflicting tendencies are presented in the iterations time measurements. Numerical experiments were carried out on a parallel computer DEC TruCluster.

In order to compare the proposed method to other techniques for parallel solution of the convection-diffusion problem, we count the arithmetical complexity of the algorithm implemented at P processors. Let $\mathcal{P}_v = \emptyset$ and let the numbers of grid nodes in the downwind and crosswind directions be $n_1 = 2Pl_1 + 1$ and $n_2 = \sum_{k=1}^{m} l_{2,k} + 1$, respectively. Let l_1, $l_{2,k}$, $k = 1, \ldots, m$, be small fixed integers. The values of n_1 and n_2 are varied according to the values of P, the number of processors, and m, the number of the crosswind strips. First, we estimate the arithmetical complexity of one iteration at one processor. The estimates of the bandwidth and the size of the matrix associated with subdomain $\Omega_{k,s}$ are $O(l_1)$ and $O(l_1 l_{2,k})$, in the case of rectangular grids. Thus, the factorization and the solution procedure will take $O(l_1^3 l_{2,k})$ (ops) and $O(l_1^2 l_{2,k})$ (ops), respectively. The arithmetical work per one iteration done by one processor is $O(\sum_{k=1}^{m} l_1^2 l_{2,k}) = O(l_1 \cdot l_1 n_2)$ (ops). Hence, the arithmetical complexity is of optimal order with respect to the number of nodes associated with the processor (for l_1 fixed). In the three-dimensional case we apply the above two-dimensional decomposition, and subdomains $\Omega_{k,s}$ become long narrow "bricks" with the long side associated with n_3 nodes. The subdomain matrix bandwidth becomes $O(l_1 l_{2,k})$ and the arithmetical work per one iteration done by one processor is $O(\sum_{k=1}^{m} l_1 l_{2,k} \cdot l_1 l_{2,k} n_3)$. Thus, in the case of a singularly perturbed convection-diffusion operator with small deviations of the transport vector from the prescribed downwind direction ($\mathcal{P} = \emptyset$), the arithmetical work done by each processor at each iteration is proportional to the number of unknowns associated with the processor. The estimate of the arithmetical complexity is therefore asymptotically unimprovable. As far as the rate of convergence of the method is concerned, it follows from (2.31) that theoretically the convergence rate is independent of the number of processors P and is proportional to the number of crosswind strips m. In our numerical experiments, however, the number of iterations is insensitive both to P and m, in the case of singular perturbation for the grid operator with $\varepsilon \ll h$.

5. Numerical experiments.

5.1. Two-dimensional problems. The set of numerical experiments presented here may be split into two parts related to different grid problems. First, we consider the simplest grid problem (2.3) and investigate the convergence and parallel properties of the two-level Schwarz method with $T_k = 2$, $k = 1, \ldots, m$, (Figures 8 and 9). The numerical results confirm the theoretical prediction given in Theorem 2.5. Then, we consider the case of $\vec{B}$ affecting the convergence of the above Schwarz method. It is the case of strong stagnant vortices; see Figure 10. In Figure 11 we show the comparative characteristics between the two-level Schwarz methods based on the uniform and adaptive domain decompositions. Although the standard Schwarz method demonstrates deterioration of the convergence rate, the adaptive version results in a fast convergent method, in accordance with Theorem 3.1. It is worth mentioning that the adaptive procedure is fully automatic and does not require from the user any knowledge of the algorithm features.

Given a domain $\Omega = (0, 1) \times (0, 1)$ and a square grid with a mesh step $h = 1/n$, we consider the problem (2.3) with

$$f_{ij} = \quad 0, \qquad\qquad x_{ij} \in \Omega \cup \Gamma_N,$$

$$g_{ij} = \begin{cases} \sin(\pi jh), & x_{ij} = (0, jh),\ 0 < j < n, \\ 0, & x_{ij} = (ih, 0),\ 0 \le i \le n, \\ 0, & x_{ij} = (ih, 1),\ 0 \le i \le n. \end{cases}$$

Given a number of processors $P = p/2$, we decompose Ω into $p \times p$ subdomains $\Omega_{k,s} = (a_k, b_k) \times (\hat{n}_1^{(s)} h, \hat{n}_2^{(s)} h)$,

$$\begin{array}{llll} a_k = n_{a,k} h, & n_{a,k} = \max((k-1)n/p - 1, 0), & k = 1, \ldots, p, \\ b_k = n_{b,k} h, & n_{b,k} = \min(kn/p + 1, n), & k = 1, \ldots, p, \\ \hat{n}_1^{(s)} = \max(n_1^{(s)} - 1, 0), & n_1^{(s)} = (s-1)n/p, & s = 1, \ldots, p, \\ \hat{n}_2^{(s)} = \min(n_2^{(s)} + 1, n), & n_2^{(s)} = sn/p, & s = 1, \ldots, p. \end{array} \tag{5.1}$$

Note that we use here equal subdomains, the overlap between them being equal to $2h$.

In Figure 8 we present the number of iterations and the iterations time needed to reduce the Euclidean norm of the residual by a factor of 10^6 starting from the trivial initial guess.

As follows from the results, $\varepsilon = 10^{-2}$ does not correspond to a discrete singular perturbation operator for considered values of h, since the number of iterations grows significantly with increasing P. The reason is that the above method does not incorporate a global coarse problem and does not take into account possible error propagation in upwind and crosswind directions due to high diffusion. By contrast, $\varepsilon \le 10^{-3}$ provides a fast convergence even with the minimal overlap $2h$. However, the iterations time has a tendency to stagnation with increasing P, particularly for small n. Such behavior is conditioned by the latency time of the message passing and the deterioration of the communication time with the number of processors P getting larger. This is clearly illustrated in Figure 9, where we show the elapsed time per iteration and the weight of the communications time in the iterations time.

Figure 11 presents the comparison of the iterations count and the time measurement related to application of the two-level method based on uniform and adaptive domain decompositions, to problem (2.3) with the same boundary conditions and the transport vector containing two strong vortices; see Figure 10. The convergence of the Schwarz iterations practically is not affected by the numbers of subdomains and processors, due to adaptive domain splitting.

5.2. Three-dimensional problems. We present three-dimensional numerical experiments in a way similar to the previous section. Given a domain $\Omega = (0,1)^3$ and a cubic grid with a mesh step $h = 1/n$, we consider the grid problem

$$\begin{aligned} \mathcal{L}_h\, u_{ijl} &= f_{ijl}, \qquad x_{ijl} \in \Omega \cup \Gamma_N, \\ u_{ijl} &= g_{ijl}, \qquad x_{ijl} \in \partial\Omega \setminus \Gamma_N, \end{aligned} \tag{5.2}$$

where $\mathcal{L}_h$ is the upwind finite difference's three-dimensional counterpart of (3.1),

$$x_{ijl} = (ih, jh, lh),\ i, j, l = 0, \ldots, n,\ \Gamma_N = \{x \equiv (x_1, x_2, x_3) : x_1 = 1, 0 < x_i < 1,\ i = 2, 3\},$$

$$f_{ijl} = \quad 0, \qquad\qquad x_{ijl} \in \Omega \cup \Gamma_N,$$

$$g_{ijl} = \begin{cases} \sin(\pi jh)\sin(\pi lh), & x_{ijl} = (0, jh, lh),\ 0 < j, l < n, \\ 0 & \text{otherwise.} \end{cases}$$

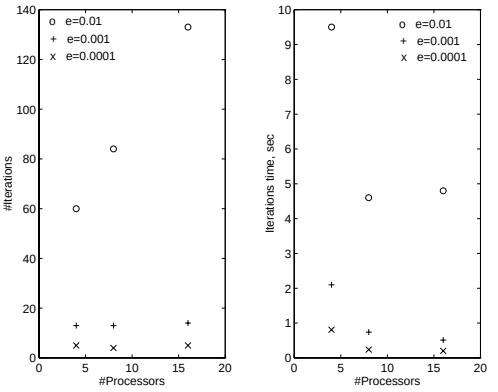


FIG. 8. *Number of iterations and iterations time,* $n = 256$.

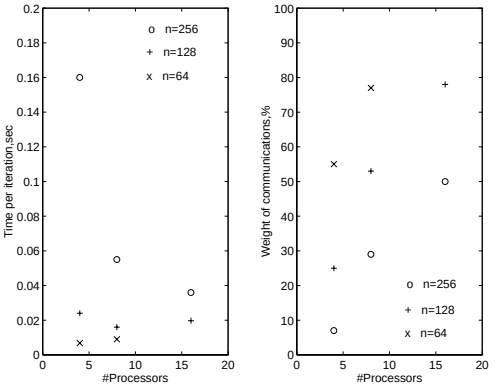


FIG. 9. *Elapsed time per iteration and communications weight.*

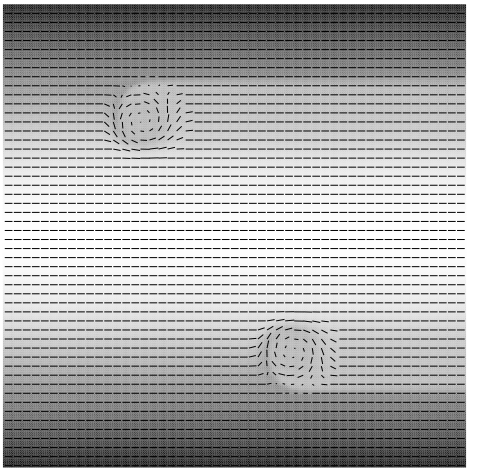


FIG. 10. *Solution of* (2.3) *in the case of strong stagnant vortices.*

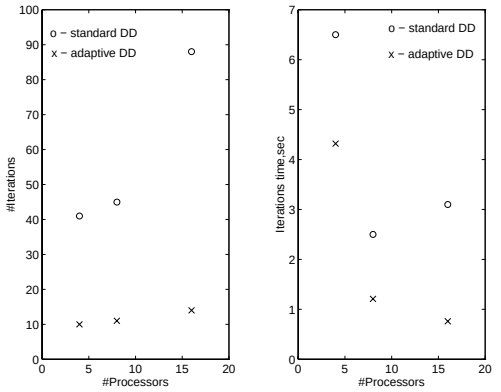


FIG. 11. *Uniform versus adaptive DD,* $\varepsilon = 10^{-3}$, $n = 256$.

TABLE 1
Number of iterations (iterations time, sec).

	Uniform DD, $\vec{B} = (1,0,0)^T$					Adaptive DD, $\vec{B}$ contains two vortices				
p	8	16		32		8	16		32	
P	4	8		16		4	8		16	
$\varepsilon \setminus n$	32	32	64	64	96	32	32	64	64	96
10^{-3}	4(0.31)	5(0.28)	5(1.7)	6(1.4)	7(4.3)	9(1.3)	6(0.55)	7(5.7)	8(3.2)	9(14.3)
10^{-4}	3(0.25)	3(0.17)	3(1.0)	3(0.7)	4(2.5)	9(1.3)	6(0.55)	7(5.7)	8(3.2)	9(14.3)

Given a number of processors $P = p/2$, we split Ω into $p \times p$ overlapping stretched subdomains $\Omega_{k,s} = (a_k, b_k) \times (\hat{n}_1^{(s)} h, \hat{n}_2^{(s)} h) \times (0,1)$, a_k, b_k, $\hat{n}_1^{(s)}$, $\hat{n}_2^{(s)}$ being defined in (5.1). The overlap between subdomains equals $2h$. Since the domain decomposition is two-dimensional, the three-dimensional Schwarz method is no different from the two-dimensional version. The stretched shape of the subdomains Ω_{ks} is well suited to the subdomain solver based on the factorization technique since it reduces significantly the natural bandwidth of the matrices to be factorized.

In Table 1 we display the number of the iterations and iterations time needed to reduce the Euclidean norm of the residual by a factor of 10^6 starting from the trivial initial guess. The uniform domain decomposition is applied to the problem with the constant transport vector, whereas the adaptive domain decomposition is extended for the solution of the problem with the transport vector containing two strong stagnant vortices; see Figure 12. Fast convergence is observed in both cases and the influence of the problem parameters such as ε, $h = n^{-1}$, P, is very small if not negligible. However, the data presented in this table do not allow us to judge the speed-up for a fixed grid problem. This being so, in Table 2 we display a scalability of the method in terms of the ratio of the time per iteration and the number of processors. We fix the grid subdomain and build up the number of unknowns by increasing the number of subdomains and the number of processors. Taking into account that the arithmetical work per processor is proportional to P, we exhibit the scalability in terms of the ratio of the time per iteration and the number of processors. If the parallel implementation is scalable, then the last value remains constant independent of the number of processors. Table 2 shows this is the case.

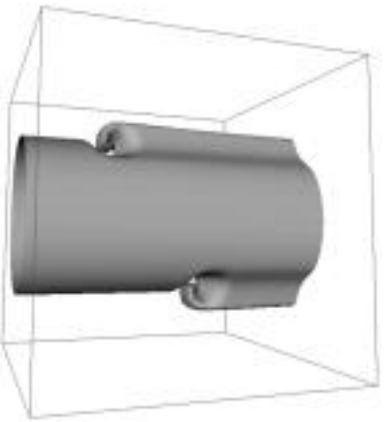


FIG. 12. *ISO-surface* $u = 0.685$ *for a three-dimensional solution of* (5.2), $\varepsilon = 10^{-4}$, *in the case of strong stagnant vortices.*

TABLE 2
Scalability, time per iteration/number of processors, $\cdot 10^{-2}$.

problem	Uniform DD				Adaptive DD			
P	4	8	12	16	4	8	12	16
grid	64×24^2	64×48^2	64×72^2	64×96^2	64×24^2	64×48^2	64×72^2	64×96^2
Scalability	2.3	2.5	2.5	2.1	4.6	4.8	5.4	5.9

6. Conclusion. We have presented the two-level Schwarz method for parallel solution of the discrete singularly perturbed convection-diffusion problems. To provide the fast convergence of the method in the case of convection fields with severe deviations from the prescribed constant direction, the adaptive domain decomposition is applied. The analysis of the method is presented. The method is readily parallelized. The results of parallel numerical experiments have shown the efficiency of the approach. Based on the above technique we develop a three-dimensional Navier–Stokes solver for channel problems.

Acknowledgment. The authors are very thankful to the Region Rhone Alpes for supporting this research.

REFERENCES

[1] H. BLUM, S. LISKY, AND R. RANNACHER, *A domain splitting algorithm for parabolic problems*, Computing, 49 (1992), pp. 11–23.

[2] Q.V. DINH, R. GLOWINSKI, AND J. PERIAUX, *Solving elliptic problems by domain decomposition methods with applications*, in Elliptic Problem Solvers II, G. Birkhoff and A. Schoenstadt, eds., Academic Press, Orlando, FL, 1984, pp. 395–426.

[3] W. ECKHAUS, *Asymptotic Analysis of Singular Perturbations*, North-Holland, Amsterdam, 1979.

[4] M. GARBEY, *A Schwarz alternating procedure for singular perturbation problems*, SIAM J. Sci. Comput., 17 (1996), pp. 1175–1201.

[5] M. GARBEY AND H.G. KAPER, *Heterogeneous domain decomposition method for singularly perturbed elliptic boundary value problems*, SIAM J. Numer. Anal., 34 (1997), pp. 1513–1544.

[6] M. Garbey and Yu.A. Kuznetsov, *Parallel Schwarz Algorithm for Equations with Singular-Perturbed Convection-Diffusion Operator*, Preprint 238, C.N.R.S.U.M.R. 5585, Université Claude Bernard Lyon 1, 1996.

[7] M. Garbey, Yu.A. Kuznetsov, and Yu.V. Vassilevski, *Parallel Schwarz Algorithm for a Problem with a Singular-Perturbed Convection-Diffusion Operator*, Preprint CDCSP-97-08, CDCSP, Université Claude Bernard Lyon 1, 1997; also available online from http://www.univ-lyon1.fr/cdcsp.

[8] J.-L. Guermond, *Un Résultat de Convergence d'Ordre Deux pour l'Approximation des Équations de Navier-Stokes par Projection Incrémentale*, C.R. Acad. Sci. Paris Sér. I Math., 325 (1997), pp. 1329–1332.

[9] F.K. Hebeker and Yu.A. Kuznetsov, *Unsteady Convection and Convection-Diffusion Problems via Overlapping Domain-Decomposition Methods*, Preprint 93-54 (SFB 359), University of Heidelberg, Germany, 1993.

[10] Y. Iliash, Y. Kuznetsov, and Y. Vassilevski, *Efficient Parallel Solvers for Two Dimensional Potential Flow and Convection-Diffusion Problems on Nonmatching Grids*, Report 357, Universität Augsburg, Germany, 1996.

[11] C. Johnson, A.H. Schatz, and L.B. Wahlbin, *Crosswind smear and pointwise errors in streamline diffusion finite element methods*, Math. Comp., 49 (1987), pp. 25–38.

[12] Yu.A. Kuznetsov, *New algorithms for approximate realization of implicit difference schemes*, Soviet J. Numer. Anal. Math. Modelling, 3 (1988), pp. 99–114.

[13] Yu.A. Kuznetsov, *Domain decomposition methods for unsteady convection-diffusion problems*, in Computing Methods in Applied Sciences and Engineering, R. Glowinski and A. Lichnewsky, eds., SIAM, Philadelphia, 1990, pp. 211–227.

[14] Yu.A. Kuznetsov, *Overlapping domain decomposition methods for FE-problems with elliptic singular perturbed operators*, in On Domain Decomposition Methods for Partial Differential Equations, Proceedings of the Fourth International Symposium, R. Glowinski, Yu.A. Kuznetsov, G. Meurant, J. Périaux, and O.B. Widlund, eds., SIAM, Philadelphia, 1991, pp. 223–241.

[15] K. Niijima, *Pointwise error estimates for a streamline diffusion finite element scheme*, Numer. Math., 56 (1990), pp. 707–719.

[16] J.M. Ortega, *Numerical Analysis: A Second Course*, SIAM, Philadelphia, 1990.

[17] C.W. Oosterlee and T. Washio, *An evaluation of parallel multigrid as a solver and a preconditioner for singularly perturbed problems*, SIAM J. Sci. Comput., 19 (1998), pp. 87–110.

[18] T. Washio and C.W. Oosterlee, *Flexible multiple semicoarsening for three-dimensional singularly perturbed problems*, SIAM J. Sci. Comput., 19 (1998), pp. 1646–1666.

[19] R. Rannacher and G. Zhou, *Analysis of a domain splitting method for non-stationary convection-diffusion problems*, East-West J. Numer. Math., 2 (1994), pp. 151–172.

[20] S.-D. Shih and R.B. Kellogg, *Asymptotic analysis of a singular perturbation problem*, SIAM J. Math. Anal., 18 (1987), pp. 1467–1511.

[21] S. Turek, *Efficient Solvers for Incompressible Flow Problems: An Algorithmic Approach in View of Computational Aspects*, Springer-Verlag, Berlin, Heidelberg, 1999.

[22] J. van Kan, *A second-order accurate pressure-correction scheme for viscous incompressible flow*, SIAM J. Sci. Comput., 7 (1986), pp. 870–891.

SIAM J. SCI. COMPUT.
Vol. 22, No. 3, pp. 917–950
© 2000 Society for Industrial and Applied Mathematics

NUMERICAL METHODS FOR THE EINSTEIN EQUATIONS IN NULL QUASI-SPHERICAL COORDINATES*

ROBERT BARTNIK† AND ANDREW H. NORTON†

Abstract. We describe algorithms used in our construction of a fourth-order in time evolution for the full Einstein equations and assess the accuracy of some representative solutions. The scheme employs several novel geometric and numerical techniques, including a geometrically invariant coordinate gauge, which leads to a characteristic-transport formulation of the underlying hyperbolic system, combined with a "method of lines" evolution; convolution splines for radial interpolation, regridding, differentiation, and noise suppression; representations using spin-weighted spherical harmonics; and a spectral preconditioner for solving a class of first-order elliptic systems on S^2. Initial data for the evolution is unconstrained, subject only to a mild size condition. For sample initial data of "intermediate" strength (19% of the total mass in gravitational energy), the code is accurate to 1 part in 10^5, until null time $z = 55$M when the coordinate condition breaks down.

Key words. black hole, convolution spline, Einstein equations, preconditioned elliptic system, spherical harmonics

AMS subject classifications. 83-08, 83C05, 65D07, 65M70, 65M20

PII. S1064827599356171

1. Introduction.

The Einstein equations present difficulties of size and complexity somewhat greater than those normally encountered in scientific computation, starting with the theoretical problem of finding a well-posed and geometrically natural formulation of the full system of Einstein equations. Numerical algorithms are then needed which can control the various facets of this system, efficiently, since very large data structures are inevitable when modeling fully 3+1-dimensional spacetimes. Finally there are the twin theoretical and practical problems of understanding the nature of the solution represented by the data, and of certifying its reliability.

In [4] we presented a new coordinate formulation for the vacuum Einstein equations, based on a characteristic (null) coordinate and a quasi-spherical foliation [3]. This null quasi-spherical (NQS) formulation is well adapted to modeling spacetimes containing a single black hole, extending from the black hole to null infinity.

The purpose of this paper is to describe the numerical algorithms we have used to solve the NQS Einstein equations, and to present results of some accuracy tests of the code. Interactive access to the data sets described here, and many other simulations, is available online at [8].

From the numerical point of view, the most significant features of the code are:

1. a characteristic coordinate $z \sim t - r$ (cf. [56]) plays the role of "time," with the numerical evolution in the direction of increasing z;
2. spherical polar coordinate grids are used on the z-level sets $\mathcal{N}_z$;
3. S^2 dependencies are handled by a combination of: spectral coefficients with respect to a basis of spin-weighted spherical harmonics (for spins $0, 1, 2$, corresponding to scalar, vector, and tensor harmonics); field values on a uniform grid in the spherical polar coordinates (ϑ, φ); and Fourier coefficients of field values on the polar grid;

*Received by the editors July 7, 1999; accepted for publication (in revised form) January 28, 2000; published electronically August 31, 2000. This project was supported by Australian Research Council grants A69330046 and A69802586.

http://www.siam.org/journals/sisc/22-3/35617.html

†School of Mathematics and Statistics, University of Canberra, ACT 2601, Australia (Bartnik@ise.canberra.edu.au, AndrewN@ise.canberra.edu.au).

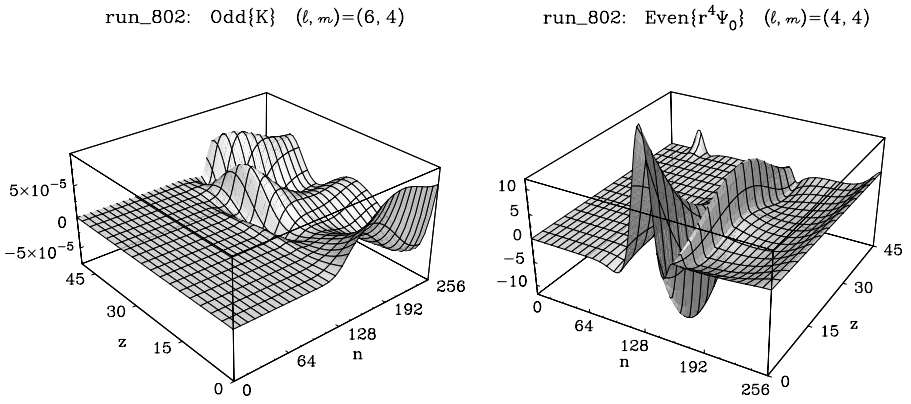


FIG. 1.1. *Sample field plots: heuristically, K and Ψ_0 represent outgoing and incoming radiation, respectively* [4]. *Observe that the nonzero limit of $r^4\Psi_0$ at $r = \infty$ ($n = 256$) is inconsistent with the Bondi–Penrose peeling hypothesis.*

4. a nonuniform radial grid in the outgoing null hypersurfaces $\mathcal{N}_z$, which compactifies future null infinity $\mathcal{I}^+$ and is adjusted dynamically so radial grid points approximately follow the inward null geodesics;

5. reformulations of the hypersurface (radial) Einstein equations which enable the numerical modeling of the asymptotic expansions of fields near null infinity;

6. fourth-order Runge–Kutta (RK4) time evolution;

7. the characteristic transport (hypersurface, radial) equations are treated using spectral collocation and integrated using an eighth-order Runge–Kutta (RK8) method [46];

8. an eighth-order convolution spline is used for interpolation, for computing radial derivatives, to realign the fields with the dynamically varying radial grid, and for suppressing high-frequency modes;

9. a first-order elliptic system on S^2 is solved at each radius and time step by an iterative method with spectral preconditioning.

Numerical consistency checks suggest that most quantities of interest which are calculated by the code (e.g., the NQS metric functions, the connection coefficients, and the Weyl curvature spinors) have relative errors of about 0.001% for simulations where the gravitational waves carry no more than 20% of the total spacetime mass. Of course, greater accuracy is found for more nearly linear, weak field, simulations. The major factor limiting the accuracy appears to be the spherical harmonic resolution, currently at $L \leq 15$. Although the code can run with $L \leq 31$, present hardware restrictions limit use at this higher resolution.

The code was developed on a 300MHz DEC Alpha with 512Mb memory and presently runs on a 300MHz Sun Ultra 2 with 784Mb, with a typical ($L = 15$) run taking between two and four days. Preliminary descriptions of the code were given in [5, 9, 7, 39, 40]. Figure 1.1 shows sample displays from the interactive website [8]. The Cauchy and characteristic initial value problems differ significantly in the nature of their appropriate initial conditions. Cauchy initial data consists of the initial 3-metric and extrinsic curvature [54], whereas initial data for a characteristic initial surface is just the null metric. In the NQS case the null metric is

$$ds^2_{\mathcal{N}_z} = (r\,d\vartheta + \beta^1\,dr)^2 + (r\sin\vartheta\,d\varphi + \beta^2\,dr)^2, \tag{1.1}$$

parameterized by the angular shear vector $\beta = \frac{1}{\sqrt{2}}(\beta^1 - \mathrm{i}\,\beta^2) = \beta(z, r, \vartheta, \varphi)$. Unlike the Cauchy problem, the NQS Einstein equations [4] do not impose any additional algebraic or differential constraints on β, so the initial data $\beta(z = 0)$ is an arbitrary vector function of the spherical polar coordinates (r, ϑ, φ), except for a mild size constraint (2.17). Heuristically β represents the in-going gravitational radiation of the spacetime; an interpretation which is consistent with the initial $(z = 0)$ values of β being freely specifiable. Note that the geometric invariant σ_{NP}/ρ_{NP} [43] becomes $-2\eth\beta/(2 - \mathrm{div}\beta)$ in NQS coordinates [4].

The NQS gauge provides a formulation of the Einstein equations as an explicit *characteristic transport* system (2.20)–(2.23), coupled to a time evolution equation. This structure is also found in characteristic formulations of other hyperbolic equations such as the wave equation, and is well known for the Einstein equations in other characteristic coordinate gauges [29, 50, 56, 37]. Although there are existence results for characteristic initial value formulations of hyperbolic equations, these rely on reduction to the Cauchy problem [48, 66]. The only exception of which we are aware is the analysis of the linear wave equation in [1, 2]. It would be valuable to have theoretical existence results for systems of characteristic transport equations, which could justify the numerical formulation described here.

The characteristic-based approach has been advocated by Winicour and coworkers [13, 27, 26], who have developed codes for solving the scalar wave equation [27], axially symmetric spacetimes [25], and for the full Einstein equations [12], based on the Bondi coordinate system [29]. These works have been fundamental in establishing the feasibility of numerical formulations based on a characteristic coordinate and in motivating the present implementation. However, the stability analyses and experience of [27, 25] are not directly applicable to our evolution procedure, since the formulations and numerical methods used have significant differences. Like the Bondi parameterization, the NQS gauge conditions are directly implemented in the metric form; however, the NQS Einstein equations are considerably simpler than the Bondi coordinate form of the Einstein equations [4, 12].

The treatment of angular derivatives is greatly simplified by the quasi-spherical condition, which encourages the use of spherical harmonic expansions. This assists comparison with black hole perturbation theory results [47, 65, 35, 20] and underpins the use of spectral methods for angular derivatives.

The combination of the characteristic-transport and method of lines techniques considerably simplifies the use of high-order algorithms, such as RK4 for time evolution. The method of lines approach to evolution equations is common in fluid dynamics [19] but has not been attempted previously in numerical relativity. Note that other high-order algorithms have been developed by Berger and Moncrief [11] to study cosmological singularities.

Higher-order techniques can produce considerably more accurate results than would be possible with second-order finite difference methods. However, there are restrictions, both geometric and algorithmic, on the class of spacetimes which can be modeled by our code.

The algorithms assume the spacetime metric is quite smooth, and hence they should not be expected to reliably treat spacetimes with strongly localized features, such as planets and gravitational shocks [4]. However, gravitational radiation arising from disruptions of a black hole is expected to be very smooth and slowly varying in angular directions, and this is strongly supported by our numerical results.

The use of an outgoing characteristic coordinate restricts the code to model-

ing spacetime regions admitting a foliation by expanding null cones. This includes the exterior regions of spacetimes which are asymptotically Minkowski/Schwarzschild (considered here) or anti-DeSitter. It turns out (see section 4.1) that it is geometrically natural and algorithmically convenient to impose inner boundary conditions corresponding to a central interior black hole. Thus the code is very well suited to modeling gravitational waves in the exterior region of a single roughly spherical black hole.

The existence of quasi-spherical coordinates [4] is not an issue here, since the code directly constructs spacetimes with NQS coordinates. Indeed, the success of the numerical code itself provides indirect support for the conjecture that NQS coordinates exist in the exterior regions of generic spacetimes with outgoing null foliations.

The paper is organized as follows.

Section 2 gives the NQS form of the equations and describes the structure of the equations and the formal steps in the solution algorithm. The geometric significance of the resulting equations is described in [4].

Section 3 describes the numerical techniques, including the representations used for fields on S^2, and the high-order convolution splines used for interpolation and differentiation in the radial direction.

Two aspects of the treatment of S^2 fields appear to be nonstandard [42, 18]: the use of fast Fourier transforms (FFTs) in both the φ and ϑ directions, based on the "torus" model of S^2 [39]; and the use of *spin-weighted* spherical harmonics to handle vector and higher rank tensor fields as well as covariant angular derivative operators.

Section 4 describes the various stages in the evolution algorithm: solving the hypersurface equations out to null infinity $\mathcal{I}^+$; reconstructing the metric from the connection variables (which includes the solution of a first-order elliptic system on the 2-sphere at each radial grid position); and the evolution of the primary field β.

Section 5 describes techniques for estimating the accuracy of the code, testing both numerical and geometric properties of the numerical solution. Numerical convergence tests estimate the effects of separately refining the resolution in the radial, time, and angular directions. The consistency of the numerical spacetime metric is tested by verifying the constraint equations for the Einstein tensor components G_{nn}, G_{nm} ((2.24), (2.25)), and by checking the accuracy of the solution at infinity using the Trautman–Bondi mass decay formula [63, 64, 15, 29]. These provide highly nontrivial tests of the consistency of the numerical solution.

2. Einstein equations and NQS metric functions.

2.1. Spacetime metric. We consider spacetimes admitting global null-polar coordinates $(z, r, \vartheta, \varphi)$ in which the metric takes the NQS form [4]

$$ds^2_{NQS} = -2u\,dz(dr + v\,dz) + (r\,d\vartheta + \beta^1\,dr + \gamma^1\,dz)^2 + (r\sin\vartheta\,d\varphi + \beta^2\,dr + \gamma^2\,dz)^2,$$

where $u > 0, v$, and $\beta = \beta^1\partial_\vartheta + \beta^2\csc\vartheta\partial_\varphi$, $\gamma = \gamma^1\partial_\vartheta + \gamma^2\csc\vartheta\partial_\varphi$ are the six unknown NQS metric functions. Note that $\partial_\vartheta, \partial_\varphi$ denote equivalently the coordinate tangent vectors and the coordinate partial differential operators $\frac{\partial}{\partial\vartheta}$, $\frac{\partial}{\partial\varphi}$. We may consider β, γ either as vector fields on S^2 or as spin 1 quantities, defined by the complex combinations [24]

$$\beta = \tfrac{1}{\sqrt{2}}(\beta^1 - \mathrm{i}\,\beta^2), \qquad \gamma = \tfrac{1}{\sqrt{2}}(\gamma^1 - \mathrm{i}\,\gamma^2). \tag{2.1}$$

The canonical example of a spacetime with metric in NQS form is Schwarzschild

spacetime in Eddington–Finkelstein retarded coordinates [21, 31]

$$ds^2_{Schw} = -2dz\,(dr + \tfrac{1}{2}(1 - 2\mathrm{M}/r)\,dz) + r^2(d\vartheta^2 + \sin^2\vartheta\,d\varphi^2) \tag{2.2}$$

with $u = 1$, $v = \frac{1}{2}(1 - 2\mathrm{M}/r)$, $\beta^A = \gamma^A = 0$, and $\mathrm{M} = const.$ This includes Minkowski space $\mathbb{R}^{3,1}$ as the case $\mathrm{M} = 0$ and $z = t - r$.

2.2. Edth. Using the complex notation (2.1), we have the canonical angular covariant derivative operator "edth" [24, 44, 22],

$$\eth\eta = \frac{1}{\sqrt{2}}\left(\frac{\partial}{\partial\vartheta} - s\cot\vartheta - \frac{\mathrm{i}}{\sin\vartheta}\frac{\partial}{\partial\varphi}\right)\eta, \tag{2.3}$$

acting on a spin s field η, and its "conjugate" operator $\bar{\eth}$,

$$\bar{\eth}\eta = \frac{1}{\sqrt{2}}\left(\frac{\partial}{\partial\vartheta} + s\cot\vartheta + \frac{\mathrm{i}}{\sin\vartheta}\frac{\partial}{\partial\varphi}\right)\eta. \tag{2.4}$$

All geometric angular derivative operators may be defined in terms of $\eth$, $\bar{\eth}$. For example, the covariant directional derivative of a spin s field η in the direction β is $\nabla_\beta\eta = \beta\bar{\eth}\eta + \bar{\beta}\eth\eta$; the divergence and curl (of a vector) are

$$\mathrm{div}\beta = \bar{\eth}\beta + \eth\bar{\beta} = \nabla_1\beta^1 + \nabla_2\beta^2, \tag{2.5}$$

$$\mathrm{curl}\beta = \mathrm{i}\,(\bar{\eth}\beta - \eth\bar{\beta}) = \nabla_1\beta^2 - \nabla_2\beta^1\ ; \tag{2.6}$$

and the spherical Laplacian is

$$\Delta\eta = (\eth\bar{\eth} + \bar{\eth}\eth)\eta\ . \tag{2.7}$$

Further properties of edth are described in section 3 and in [22, 44, 4, 10].

2.3. Connection variables. In addition to the metric functions (u, v, β, γ) we introduce the connection fields $H, J, K, Q, Q^\pm$

$$H = u^{-1}(2 - \mathrm{div}\beta), \tag{2.8}$$

$$J = v(2 - \mathrm{div}\beta) + \mathrm{div}\gamma, \tag{2.9}$$

$$K = v\eth\beta - \eth\gamma, \tag{2.10}$$

$$Q^\pm = u^{-1}(Q \pm \eth u), \tag{2.11}$$

$$Q = r\frac{\partial\beta}{\partial z} - r\frac{\partial\gamma}{\partial r} + \gamma + \nabla_\beta\gamma - \nabla_\gamma\beta. \tag{2.12}$$

Observe that u, v, H, J are real and have spin 0, whereas $\beta, \gamma, Q, Q^+, Q^-$ have spin 1 and K has spin 2.

Given the metric functions u, v, β, γ on a z-level set $\mathcal{N}_z$, we may construct H, J, K on $\mathcal{N}_z$ directly, and Q (and $Q^\pm$) may be reconstructed if in addition, $\partial\beta/\partial z$ is also known on $\mathcal{N}_z$. It is clear from (2.8)–(2.12) that this construction does not require any compatibility conditions on the data $u, v, \beta, \gamma, \frac{\partial}{\partial z}\beta$.

There is a converse construction for the metric functions u, v, γ, and $\partial\beta/\partial z$, which also involves totally free and unconstrained data, namely the connection variables H, J, K, Q. This contrasts sharply with the description of the connection via the Newman–Penrose spin coefficients [36, 44], which requires numerous differential constraint equations, expressing the property that the connection is torsion-free.

The converse construction works as follows. Given β and the connection variables (H, J, K) on $\mathcal{N}_z$, we reconstruct u via the relation

$$u = H^{-1}(2 - \mathrm{div}\beta), \tag{2.13}$$

and we find v, γ by solving an elliptic system for γ,

$$\mathcal{L}_\beta \gamma := \eth\gamma + \frac{\eth\beta}{2 - \mathrm{div}\beta}\,\mathrm{div}\gamma = J\frac{\eth\beta}{2 - \mathrm{div}\beta} - K, \tag{2.14}$$

and setting

$$v = \frac{J - \mathrm{div}\gamma}{2 - \mathrm{div}\beta}. \tag{2.15}$$

The system (2.14) is $\mathbb{R}$-linear and elliptic with 6-dimensional kernel, provided $\eth\beta$ is not too large ($|\eth\beta| < (2 - \mathrm{div}\beta)/\sqrt{3}$ is sufficient). Prescribing the $l = 1$ spherical harmonic coefficients of γ (for example, by requiring $\gamma_{l=1} = 0$) suffices to determine the solution γ uniquely. The remaining connection parameter Q, together with the now known values of β, γ on $\mathcal{N}_z$, determines the *evolution equation*

$$\frac{\partial\beta}{\partial z} = \frac{\partial\gamma}{\partial r} + \frac{1}{r}(Q + \nabla_\gamma\beta - \nabla_\beta\gamma - \gamma). \tag{2.16}$$

To summarize, given β on a single level set $\mathcal{N}_z$, satisfying the size constraint

$$|\eth\beta| < (2 - \mathrm{div}\beta)/\sqrt{3}, \tag{2.17}$$

the map $(u, v, \gamma, \beta_z) \mapsto (H, J, K, Q, \gamma_{l=1})$ is invertible, assuming all fields are sufficiently smooth. In section 4.4 we describe the numerical implementation of the inverse map.

2.4. NQS Einstein equations. To compute the NQS form of the Einstein tensor, we introduce the complex null vector frame $(\ell, n, m, \bar{m})$,

$$\begin{aligned} \ell &= \partial_r - r^{-1}\beta, \\ n &= u^{-1}(\partial_z - r^{-1}\gamma - v(\partial_r - r^{-1}\beta)), \\ m &= \frac{1}{r\sqrt{2}}(\partial_\vartheta - \mathrm{i}\,\csc\vartheta\,\partial_\varphi), \end{aligned} \tag{2.18}$$

and the directional derivative operators

$$\mathcal{D}_r = \partial_r - r^{-1}\nabla_\beta\,, \qquad \mathcal{D}_z = \partial_z - r^{-1}\nabla_\gamma. \tag{2.19}$$

Expressions for the Newman–Penrose spin coefficients [36] with respect to the frame $(\ell, n, m, \bar{m})$ are given in terms of H, J, K, Q in [4].

The frame components of the Einstein tensor G_{ab}, $a, b = \ell, n, m, \bar{m}$, may be written in terms of the NQS metric functions and NQS connection variables. These expressions may be grouped into *hypersurface equations* (or *main equations* [29, 51]):

$$r\mathcal{D}_r H = \left(\tfrac{1}{2}\mathrm{div}\beta - \frac{2|\eth\beta|^2 + r^2 G_{\ell\ell}}{2 - \mathrm{div}\beta}\right) H, \tag{2.20}$$

$$r\mathcal{D}_r Q^- = (\eth\bar{\beta} - uH)Q^- + \bar{Q}^-\eth\beta + 2\bar{\eth}\eth\beta + u\eth H - H\eth u + 2r^2 G_{\ell m}, \tag{2.21}$$

$$r\mathcal{D}_r J = -(1 - \mathrm{div}\beta)J + u - \tfrac{1}{2}u|Q^+|^2 - \tfrac{1}{2}u\,\mathrm{div}(Q^+) - ur^2 G_{\ell n}, \tag{2.22}$$

$$r\mathcal{D}_r K = \left(\tfrac{1}{2}\mathrm{div}\beta + \mathrm{i}\,\mathrm{curl}\beta\right) K - \tfrac{1}{2}J\eth\beta + \tfrac{1}{2}u\eth Q^+ + \tfrac{1}{4}u(Q^+)^2 + \tfrac{1}{2}ur^2 G_{mm}, \tag{2.23}$$

the *boundary equations* (or *subsidiary equations*)

$$(2.24)\quad r\,\mathcal{D}_z\,(J/u) = v^2\,r\mathcal{D}_r\,(J/(uv)) + \tfrac{1}{2}(\mathrm{div}\gamma - v\,\mathrm{div}\beta)J/u + 2u^{-1}|K|^2 - \nabla_{Q^+}v - \Delta v + ur^2G_{nn},$$

$$(2.25)\quad r\,\mathcal{D}_zQ^+ = (v\,r\mathcal{D}_r + J - v\eth\bar\beta + \eth\bar\gamma)Q^+ - K\bar Q^+ + 2\bar\eth K + 2u^{-1}r\mathcal{D}_r(u\eth v) - (2 + \mathrm{i}\,\mathrm{curl}\beta)\eth v + \eth J - 2u^{-1}J\eth u - 2ur^2G_{nm},$$

and the *trivial equation*

$$(2.26)\quad ur^2G_{m\bar m} = r\mathcal{D}_rJ - \tfrac{1}{2}\mathrm{div}\beta\,J - u|Q^+|^2 + \tfrac{1}{2}u\,\mathrm{div}Q^+ + \bar Q^+\eth u + Q^+\bar\eth u + \bar K\eth\beta + K\bar\eth\bar\beta + r^2(v\mathcal{D}_r - \mathcal{D}_z)(u^{-1}\mathcal{D}_ru) + u^{-1}r^2\mathcal{D}_r(u\mathcal{D}_rv).$$

Of course, the Einstein tensor components in these formulae are set to zero for the vacuum equations.

Observe that the hypersurface equations (2.20)–(2.23) have no explicit z-derivatives, and they each contain only one radial (r) derivative. The form of the connection variables (H, J, K, Q) was determined by exactly these properties. Consequently, the hypersurface equations may be written schematically in terms of $U = (H, Q^-, J, K)$ in the "characteristic-transport" form

$$(2.27)\qquad r\frac{\partial U}{\partial r} = F(\beta(z,r), U(z,r))$$

by treating angular derivatives such as $\eth U$ as determined by the set of values $U(z,r)$ on the full S^2. This system has the effect of transporting the fields U along the characteristic curves with tangent vector ℓ which foliate the null hypersurfaces $\mathcal{N}_z$.

Note that alternative reformulations of the hypersurface equations are possible, preserving the general characteristic transport structure. For reasons associated with reliably capturing the asymptotic behavior of the fields, the present version of the code integrates the following radial equations, for the variables $\log(H/2)$ (instead of H), $j = \frac{1}{4}r(2 - HJ) - \mathrm{M}$ (instead of J), and rQ^+ (instead of Q^-):

$$(2.28)\quad r\partial_r\log(\tfrac{1}{2}H) = \nabla_\beta\log(H/2) + \tfrac{1}{2}\mathrm{div}\beta - \frac{2|\eth\beta|^2 + r^2G_{\ell\ell}}{2 - \mathrm{div}\beta},$$

$$(2.29)\quad r\partial_r(rQ^+) = \nabla_\beta(rQ^+) - (1 - 2\eth\bar\beta)(rQ^+) + \nabla_{rQ^+}\beta + 2r\eth(r\mathcal{D}_r\log u) + 2r\beta - \mathrm{i}\,r\eth\mathrm{curl}\beta + 2r^3G_{\ell m},$$

$$(2.30)\quad r\partial_rj = \nabla_\beta j + \left(\tfrac{3}{2}\mathrm{div}\beta - \frac{2|\eth\beta|^2}{2 - \mathrm{div}\beta}\right)(j + \mathrm{M} - \tfrac{1}{2}r) + \tfrac{1}{4}r\mathrm{div}\beta + \tfrac{1}{8}r(2 - \mathrm{div}\beta)(|Q^+|^2 + \mathrm{div}Q^+) + \tfrac{1}{4}r^3(u^{-1}JG_{\ell\ell} + uHG_{\ell n}),$$

$$(2.31)\quad 2r\partial_rK = 2\nabla_\beta K + (\mathrm{div}\beta + 2\mathrm{i}\,\mathrm{curl}\beta)\,K - J\eth\beta + u\eth Q^+ + \tfrac{1}{2}u(Q^+)^2 + ur^2G_{mm}.$$

Here $\mathrm{M} = 1$ fixes the bare mass of the background Schwarzschild black hole.

It is remarkable that the boundary equations (2.24), (2.25) and the trivial equation (2.26) may be regarded as *compatibility* relations, by virtue of the conservation (contracted Bianchi) identity $G_{ab}{}^{;b} = 0$ [51, 4]. This identity is valid for any Einstein tensor G_{ab}, regardless of the metric. Substituting the hypersurface equations $G_{\ell\ell} = G_{\ell m} = G_{\ell n} = G_{mm} = 0$ into the conservation identity, yields equations $HG_{m\bar m} = 0$ and a propagation system for G_{nn}, G_{nm} which has the unique solution

$G_{nn} = G_{nm} = 0$ if the boundary equations are satisfied on one hypersurface transverse to the outgoing null surfaces $\mathcal{N}_z$. Thus, in order to construct a solution of the full vacuum Einstein equations, it suffices to satisfy the hypersurface equations everywhere, and the boundary equations just on the boundary surface $r = r_0$.

3. Numerical techniques. In this section we describe the data representation and manipulation techniques. These consist mainly of techniques for handling angular fields and derivatives, and an unusual convolution spline used for interpolation, differentiation, and high frequency filtering in the radial and time directions.

3.1. Fields on S^2. The evolution algorithm treats the angular derivatives $\frac{\partial}{\partial\vartheta}, \frac{\partial}{\partial\varphi}$ as "lower order," compared to the radial and time derivatives $\frac{\partial}{\partial r}, \frac{\partial}{\partial z}$. This attitude in a numerical computation can be justified only if it is possible easily and accurately to compute and manipulate angular derivatives. This is achieved by using spectral representations (both Fourier and spherical harmonic) for fields on S^2. This approach is widely used in geophysical and meteorology applications [34, 42, 16, 58, 61] and is known to have significant advantages compared to finite difference approaches [18], based on either angular coordinate grids or overlapping stereographic projection charts [55, 53, 28]. Nevertheless, spectral methods have been used only rarely in numerical general relativity (cf. [45, 14]) and they have not been used previously for solving the full Einstein equations.

The basic manipulations required of S^2 fields are:

(i) computing nonlinear algebraic terms, such as $1/(2 - \mathrm{div}\beta)$, $u|Q^+|^2$, etc.;

(ii) computing angular derivative terms, such as $\mathrm{div}\beta$, $\eth Q^+$, etc.;

(iii) inverting the linear elliptic operator $\mathcal{L}_\beta$ (see (2.14)); and

(iv) projecting S^2 grid values of aliased or noisy fields onto certain subspaces of spin-weighted spherical harmonics.

Three separate representations for fields on S^2 have been implemented to carry out these manipulations:

(i) *field values* $\eta_{jk} = \eta(\vartheta_j, \varphi_k)$ at the polar coordinate grid points

$$(\vartheta_j, \varphi_k) = ((j - \tfrac{1}{2})\Delta\vartheta, (k-1)\Delta\varphi), \tag{3.1}$$

where $\Delta\vartheta = 2\pi/N$, $\Delta\varphi = 2\pi/N$ with $1 \le j \le N/2, 1 \le k \le N$, and (in our implementations) $N = 16$, 32, or 64;

(ii) *Fourier coefficients* arising from FFT transforms in either the ϑ or φ directions, of the field values η_{jk};

(iii) spin-weighted *spherical harmonic coefficients* η^{lm}, $|m| \le l$, $l = s, \ldots, L$, $L = N/2 - 1$ (for spin $s = 0, 1$ or 2).

The field values are used when computing nonlinear algebraic terms such as $u|Q^+|^2$. The Fourier representation is used for computing ϑ and φ angular derivatives, which are needed in the formulas for $\eth$, div, for example. The spherical harmonic representation is used in solving the elliptic system (2.14), to spectrally limit field values by projection to spherical harmonic data, and to save the results.

For fields which do not alias on the (ϑ, φ)-grid, the three representations are equivalent in the sense that conversion between them is essentially exact, depending only on machine precision and algebraic details of the FFT. The requirement that a field does not alias is satisfied when it can be represented by a finite expansion in spin-weighted spherical harmonics with angular momentum $l \le L = N/2 - 1$. We use spectral cutoffs $L = 15$ (for $N = 32$) and $L = 31$ (for $N = 64$).

Transformation to the spherical harmonic representation involves a projection, because both the field value and Fourier representations have approximately twice as many degrees of freedom. For example, a (real) spin 0 field with $l \le L = N/2 - 1$ has $(L+1)^2 = N^2/4$ spherical harmonic coefficients, whereas it has $N^2/2$ values on the (ϑ, φ)-grid. The space of nonaliasing spherical harmonics is a *linear* subspace of the space of functions represented by either Fourier coefficients or field values.

For example, when the (ϑ, φ)-grid field values of the product of two fields is calculated, the result, which will contain components up to $l \le 2L$, is aliased onto the grid in such a way that its field values no longer lie in the appropriate spin-weighted spherical harmonic subspace. To clean up after such nonlinear effects, we project the result back onto the correct subspace, as described in section 3.5.

To minimize the possibility of unstable feedback of quadratic aliasing errors, we may invoke the Orszag 2/3 rule [19, 41] at various points within the code. The effective spectral resolution of the code is then $l_{\max} \approx 2L/3$ ($l_{\max} = 10$ for $N = 32$ and $l_{\max} = 20$ for $N = 64$).

3.2. Spherical harmonics. We first summarize the more important properties of $\eth$ ("edth") and spin-weighted spherical harmonics. The edth formalism provides a unified geometric approach to the treatment of angular derivatives on S^2 and of vector and higher-rank tensor harmonics. Detailed descriptions of the properties of spin-weighted fields and spherical harmonics may be found in Penrose and Rindler [44] or [24, 22]. Here we describe only the basic formulae.

We use a real-valued basis Y_{lm}, $l = 0, 1, 2, \ldots$, $m = -l, \ldots, l$ for the space of spin 0 spherical harmonic functions, defined by

$$Y_{lm} = \overline{P}_{lm}(\vartheta) F_m(\varphi)\,, \tag{3.2}$$

where

$$F_m(\varphi) = \begin{cases} 1 & m = 0, \\ \sqrt{2}\cos m\varphi & m > 0, \\ \sqrt{2}\sin |m|\varphi & m < 0, \end{cases} \tag{3.3}$$

and the $\overline{P}_{lm}(\vartheta) = \overline{P}_{l|m|}(\vartheta)$ are related to the associated Legendre functions P_{lm} by

$$\overline{P}_{lm}(\vartheta) = (-1)^m \sqrt{2l+1} \sqrt{\frac{(l-m)!}{(l+m)!}} P_{lm}(\cos\vartheta)\,, \tag{3.4}$$

$$P_{lm}(\cos\vartheta) = \frac{(-1)^m}{2^l l!} \sin^m\vartheta \left[\frac{d^{l+m}}{dx^{l+m}} (x^2-1)^l \right]_{x=\cos\vartheta}. \tag{3.5}$$

The spin s spherical harmonics Y^s_{lm} are then defined explicitly by [44]

$$Y^s_{lm} = (-1)^s \left[\frac{2^s (l-s)!}{(l+s)!} \right]^{1/2} \eth^s Y_{lm}\,, \qquad s > 0, \tag{3.6}$$

$$Y^{-s}_{lm} = \left[\frac{2^s (l-s)!}{(l+s)!} \right]^{1/2} \bar{\eth}^s Y_{lm}\,, \qquad -s < 0, \tag{3.7}$$

where necessarily $l \ge |s|$. Note that the differential operator $\eth$ is spin-raising, sending spin s into spin $(s+1)$ fields, and $\bar{\eth}$ is spin-lowering,

$$\eth Y^s_{lm} = -\left[\tfrac{1}{2}(l+s+1)(l-s)\right]^{1/2} Y^{s+1}_{lm}, \tag{3.8}$$

$$\bar{\eth} Y^s_{lm} = \left[\tfrac{1}{2}(l-s+1)(l+s)\right]^{1/2} Y^{s-1}_{lm}, \tag{3.9}$$

TABLE 3.1
Nomenclatures and representations for the Hodge–Helmholtz decomposition of a vector field on S^2.

Even:	polar	irrotational	$\eth f$	$\operatorname{grad} f$	$(\nabla_1 f)\, v_1 + (\nabla_2 f) v_2$
Odd:	axial	divergence-free	$\mathrm{i}\,\eth g$	$\operatorname{curl} g$	$(\nabla_2 g)\, v_1 - (\nabla_1 g) v_2$

for all $s \in \mathbb{Z}$, and that Y^s_{lm} and Y^{-s}_{lm} are related by complex conjugation,

$$Y^{-s}_{lm} = (-1)^s \bar{Y}^s_{lm}. \tag{3.10}$$

Since $\Delta Y_{lm} = -l(l+1)Y_{lm}$, the fundamental commutation relation

$$[\bar{\eth}, \eth]\eta = (\bar{\eth}\eth - \eth\bar{\eth})\eta = s\eta, \tag{3.11}$$

for any spin s field η, may be used to show that

$$\Delta Y^s_{lm} = (s^2 - l(l+1))Y^s_{lm}, \tag{3.12}$$

where $\Delta = \eth\bar{\eth} + \bar{\eth}\eth$. With these conventions we have the orthogonality relations

$$\frac{1}{4\pi}\int_{S^2} Y^s_{lm}\, \bar{Y}^s_{l'm'} \sin\vartheta \, d\vartheta d\varphi = \delta_{ll'}\delta_{mm'},$$

which show that the Y^s_{lm} form a basis (over $\mathbb{C}$) of the Hilbert space of square-integrable spin s fields on S^2, which is orthonormal in the natural Hermitian inner product

$$\langle \phi, \psi \rangle = \frac{1}{4\pi}\oint_{S^2} \mathrm{Re}(\bar{\phi}\psi). \tag{3.13}$$

From (3.2)–(3.5) it is evident that the spin 0 harmonics Y_{lm} are trigonometric polynomials in ϑ and φ. Using expression (2.3) for $\eth$, we also see that the Y^s_{lm} are trigonometric polynomials. The highest wave number Fourier modes which occur in the set of basis functions $\{Y^s_{lm} : s \le l \le L,\ |m| \le l\}$ are $\cos(L\vartheta)$, $\sin(L\vartheta)$, $\cos(L\varphi)$, and $\sin(L\varphi)$. Therefore, a uniform (ϑ, φ)-grid of size $N/2 \times N$ can represent spin-weighted spherical harmonic functions up to $L = N/2 - 1$.

3.3. Even/odd decomposition. Because $\eth$ is surjective onto the space of smooth spin s fields for $s \ge 1$ [44], the decomposition of spin 0 fields into real and imaginary parts may be propagated to higher spin. The resulting decomposition into *even* and *odd* components plays an important role in the analysis of the linearized Einstein equations about the Schwarzschild spacetime [47]. Because the NQS geometry distinguishes the Schwarzschild metric and is also based on spherical harmonics, it is ideally suited to comparing nonlinear evolution to the comparatively well-understood black hole linearized Einstein equations [20, 23].

We say that the spin $s \ge 0$ field η is *even* if $\eta = \eth^s f$ for some real-valued function f, and η is *odd* if $\eta = \mathrm{i}\,\eth^s g$ for some real-valued function g. This matches the usage in [47]—note that for axially symmetric fields the terms *polar* (for even) and *axial* (for odd) are sometimes used [20]. The surjectivity of $\eth$ onto spin $s \ge 1$ ensures that every spin s field may be uniquely decomposed into a sum of even and odd parts. For $s = 1$ this decomposition corresponds to the Hodge–Helmholtz decomposition of a vector field into the sum of a gradient and a dual gradient (curl); see Table 3.1. This has a

natural interpretation in terms of the spectral decomposition

$$\eta = \sum_{l=s}^{\infty} \sum_{m=-l}^{l} \eta^{lm} Y^s_{lm} \tag{3.14}$$

of a spin s field η, because we are using a basis of real-valued Y_{lm}. Namely, η is *even* if the spectral coefficients η^{lm} are real, and *odd* if the coefficients are pure imaginary. We may thus decompose $\eta = \mathrm{Even}(\eta) + \mathrm{Odd}(\eta)$ with

$$\mathrm{Even}(\eta) = \sum \mathrm{Re}(\eta^{lm}) Y^s_{lm}, \qquad \mathrm{Odd}(\eta) = \mathrm{i} \sum \mathrm{Im}(\eta^{lm}) Y^s_{lm}. \tag{3.15}$$

Observe that $\eth Y^s_{sm} = 0$ for $s \geq 0$, and thus $\eth$ acting on spin s fields with $s \geq 0$ has kernel having (complex) dimension $2s+1$. Likewise the formal adjoint $-\bar{\eth}$ acting on spin $s \leq 0$ fields has $(2|s|+1)$-dimensional kernel. In particular, $\eth$ acting on spin 1 fields has kernel consisting of the $\mathbb{C}$-linear space spanned by the three $l = 1$ spin 1 spherical harmonics Y^1_{1m}—the corresponding real vector fields are the dual gradients of functions linear in $\mathbb{R}^3$, which are the infinitesimal rotations, and the gradients, which are the infinitesimal conformal dilations.

The correspondence between vector fields on S^2 and spin 1 fields generalizes to spin 2 fields, which correspond to symmetric traceless 2-tensors on S^2. If λ is a symmetric traceless 2-tensor, then with respect to the standard orthonormal frame

$$e_1 = \partial_\vartheta, \quad e_2 = \csc\vartheta\, \partial_\varphi, \tag{3.16}$$

we have the correspondence $\lambda = \lambda^{ij} e_i \otimes e_j \sim \frac{1}{2}(\lambda^{11} - \lambda^{22} - 2\mathrm{i}\,\lambda^{12})$. This correspondence extends to higher integer spins with higher rank symmetric traceless tensors on S^2.

3.4. Fourier representation. The FFT is used to transform between Fourier coefficients and field values on the uniform (ϑ, φ)-grid. Fourier convergence problems arising from discontinuities in coordinate derivatives and vector and tensor components at the poles are sidestepped by an observation relating fields on S^2 to fields on the torus $\mathbb{T}^2 = S^1 \times S^1$ [39]. The torus method enables derivatives for all types of field on S^2 to be computed using Fourier methods, so is particularly well suited to handling the derivative operator $\eth$ (2.3). This approach to handling component discontinuities at the poles is simpler than the techniques reviewed in [59] for manipulating vector fields and readily extends to any rank $s \geq 0$.

For integer s the real and imaginary parts of a spin s field on S^2 may be identified with the two independent frame components of a completely symmetric trace-free tensor of rank $|s|$ on S^2 [44]. Since the frame e_1, e_2 (3.16) is not continuous at the poles, the tensor components will not be continuous at the poles, so are not obviously suited to Fourier expansion in the ϑ direction.

However, along any smooth curve crossing through a pole, both basis vectors e_1 and e_2 reverse direction at the pole. Thus, for any smooth tensor field $T = T^{j_1 \cdots j_s} e_{j_1} \otimes \cdots \otimes e_{j_s}$, by continuity of T the component functions $T^{j_1 \cdots j_s}$ will change by a factor $(-1)^s$ across the poles. Consequently, if we extend the domain of definition of $T^{j_1 \cdots j_s}$ to $\vartheta \in [-\pi, \pi]$ by

$$T^{j_1 \cdots j_s}(-\vartheta, \varphi) \;=\; (-1)^s T^{j_1 \cdots j_s}(\vartheta, \varphi + \pi)\,, \quad \text{for } \vartheta \in [0, \pi] \tag{3.17}$$

(using the 2π periodicity in φ), then the resulting extension is 2π-periodic and continuous in ϑ. This argument extends to higher (covariant) derivatives of T, showing that

the extension is in fact *smooth* and periodic in ϑ. Derivatives of $T^{j_1\cdots j_s}$ with respect to ϑ can then be calculated just as for φ derivatives, provided that the direction of increasing ϑ is properly taken into account.

In effect, the extension just described defines a (smooth) field $T^{j_1\cdots j_s}$ on the torus $\mathbb{T}^2 = S^1 \times S^1$. This may be understood geometrically by noting that the map

$$\Upsilon : \mathbb{T}^2 \to S^2, \quad (\vartheta, \varphi) \mapsto \begin{cases} (\vartheta, \varphi), & \vartheta \in (0, \pi],\ \varphi \in [0, 2\pi) \\ (-\vartheta, \varphi + \pi), & \vartheta \in (-\pi, 0],\ \varphi \in [0, 2\pi) \end{cases} \tag{3.18}$$

is in fact *smooth*. This follows by noting that because ϑ is a radial coordinate near the north pole $\vartheta = 0$, the differential structure near the pole is represented by the rectangular coordinates $(\xi, \eta) = (\vartheta \cos\varphi, \vartheta \sin\varphi)$ and the map $(\vartheta, \varphi) \mapsto (\xi, \eta)$ is manifestly C^∞ for ϑ near 0. Consequently any smooth tensor T on S^2, when expressed in a cotangent basis, pulls back to a smooth tensor on $\mathbb{T}^2$ (i.e., $\Upsilon^*(T) \in C^\infty(\mathbb{T}^2)$), and thus admits a well-behaved Fourier representation on $\mathbb{T}^2$. The converse is of course false: a smooth tensor on $\mathbb{T}^2$ does not necessarily arise from a smooth tensor on S^2, even if it satisfies the parity condition (3.17) satisfied by pull-back tensors.

The ϑ and φ derivatives in the coordinate form (2.3) of $\eth$ (and similarly any other covariant angular derivative operator) can be evaluated easily by transforming to the appropriate Fourier representation of the field, multiplying the Fourier coefficients by the appropriate wavenumber factors, then transforming back to the field value representation. Thus, computing a derivative operator such as $\eth$ is an $O(N^2 \log N)$ operation. Since $\csc(\frac{1}{2}\Delta\vartheta) \simeq 10$ for $N = 32$, there is no significant loss of precision in calculating (2.3) near the poles.

It should be emphasized that because the Y^s_{lm} are trigonometric polynomials in (ϑ, φ), the FFT computation of their numerical derivatives is *algebraically exact*. For example, the Laplacian relation (3.12) is numerically verified to 1 part in 10^{12} [39].

3.5. The spherical harmonic representation. In this section we describe the methods used to transform fields from the field value and Fourier representations to the spherical harmonic representation. Transformations for fields of spin 0, 1, and 2 are required in the code; the spin 0 case which we describe here to illustrate the technique is slightly less complicated, since we may assume the field f is real-valued.

There are $(L+1)^2$ basis functions in the set $\{Y_{lm} : 0 \le |m| \le l \le L\}$. However, to represent these functions as trigonometric polynomials on a regular S^2 grid we require a grid of size $(L+1) \times 2(L+1)$, and thus $2(L+1)^2$ real coefficients. The spin 0 functions of angular momentum at most L therefore form a subspace of real dimension $(L+1)^2$ in the space of Fourier series representable on the grid, which has real dimension $2(L+1)^2$. We use a projection onto the spherical harmonic subspace which is orthogonal with respect to the natural inner product in the Fourier space,

$$\begin{aligned} \langle f_1,\, f_2 \rangle_{_F} &= \frac{1}{4\pi^2} \int_0^{2\pi} \int_{-\pi}^{\pi} f_1(\vartheta, \varphi) f_2(\vartheta, \varphi)\, d\vartheta d\varphi \\ &= \frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} f_1(\vartheta_i, \varphi_j) f_2(\vartheta_i, \varphi_j)\,, \end{aligned} \tag{3.19}$$

where $\{(\vartheta_i, \varphi_j) : i, j = 1, \ldots, N\}$ are grid points (cf. (3.1)) and $N = 2(L+1)$.

To make use of (3.19) we use (3.17) to extend functions defined on S^2 to functions defined on the torus $\mathbb{T}^2 = S^1 \times S^1$. In particular, given any set of values $\{f_{ij} \in \mathbb{R} : i = 1, \ldots, N/2,\ j = 1, \ldots, N\}$ on the S^2 grid, we use (3.17) to construct grid values

on $\mathbb{T}^2$. There is then a unique interpolating trigonometric polynomial f such that $f(\vartheta_i, \varphi_j) = f_{ij}$. We project f to the $l \le L$ spherical harmonic subspace as follows.

The Y_{lm} are not orthonormal with respect to (3.19), but instead have Fourier inner product

$$G_{lm\, l'm'} = \langle Y_{lm},\, Y_{l'm'} \rangle_F = \langle \overline{P}_{lm}(\vartheta),\, \overline{P}_{l'm}(\vartheta) \rangle_F \, \delta_{mm'} \,. \tag{3.20}$$

Here, the index pair lm (and $l'm'$) is a combined index which takes $(L+1)^2$ values, and (3.20) is the matrix for the induced Fourier metric on the spin 0 subspace. The summation convention will be employed for raised and lowered repeated indices.

For fixed m, the inner product (3.20) of the $\overline{P}$ functions forms a matrix,

$$A_{(m)ll'} = \langle \overline{P}_{lm}(\vartheta),\, \overline{P}_{l'm}(\vartheta) \rangle_F \,.$$

These matrices are defined only for $|m| \le l, l' \le L$, so are square and of size $(L+1-|m|) \times (L+1-|m|)$. Denoting the inverse matrix by $A^{ll'}_{(m)}$, we have

$$G_{lm\, l'm'} = A_{(m)ll'} \delta_{mm'} \,, \tag{3.21}$$

and the components of the inverse metric are given by

$$G^{lm\, l'm'} = A^{ll'}_{(m)} \delta^{mm'} \quad \text{(no sum on } m\text{)}. \tag{3.22}$$

The dual basis vectors for the spin 0 subspace are

$$Y^{lm} = G^{lm\, l'm'} Y_{l'm'} \,, \tag{3.23}$$

and satisfy $\langle Y_{lm},\, Y^{l'm'} \rangle_F = \delta^{l'}_{l}\, \delta^{m'}_{m}$. The orthogonal projection of f onto the subspace is given by $\mathrm{proj}(f) = \langle f,\, Y^{lm} \rangle_F Y_{lm} = f^{lm} Y_{lm}$, where

$$f^{lm} = \langle f,\, Y^{lm} \rangle_F \tag{3.24}$$

are the spherical harmonic coefficients of the function f. To calculate the inner product (3.24), first note that using (3.2), (3.22), and (3.23), the dual basis vectors can be written as

$$Y^{lm} = \overline{P}^{lm}(\vartheta) F^m(\varphi) \tag{3.25}$$

(in analogy with (3.2)), where we have set

$$\overline{P}^{lm}(\vartheta) = A^{ll'}_{(m)} \overline{P}_{l'm}(\vartheta) \,, \quad F^m(\varphi) = F_m(\varphi). \tag{3.26}$$

By Fourier analysis of f in the φ direction we can write $f = \hat{f}^k(\vartheta) F_k(\varphi)$. In particular, by φ-FFT of $\{f_{ij}\}$ we get the numbers $\hat{f}^k(\vartheta_i)$. The spectral coefficients f^{lm} can then be evaluated using (3.19) and (3.25) as

$$\begin{aligned} f^{lm} &= \langle \hat{f}^k(\vartheta) F_k(\varphi),\, \overline{P}^{lm}(\vartheta) F^m(\varphi) \rangle_F \\ &= \langle \hat{f}^m(\vartheta),\, \overline{P}^{lm}(\vartheta) \rangle_F \\ &= \frac{1}{N} \sum_{i=1}^{N} \hat{f}^m(\vartheta_i) \overline{P}^{lm}(\vartheta_i) \,. \end{aligned} \tag{3.27}$$

The converse process of reconstructing the function values $f_{ij} = f(\vartheta_i, \varphi_j)$ from the spherical harmonic coefficients f^{lm} follows from

$$f = f^{lm}\, Y_{lm}(\vartheta, \varphi) = f^{lm} \overline{P}_{lm}(\vartheta) F_m(\varphi)\,.$$

First we construct the quantities

$$\hat{f}^m(\vartheta_i) = \sum_{l=|m|}^{L} f^{lm} \overline{P}_{lm}(\vartheta_i) \qquad \text{(no sum on } m\text{)}, \tag{3.28}$$

and then we use inverse FFTs in the φ direction to reconstruct f_{ij} via

$$f_{ij} = \sum_{m=-L}^{L} \hat{f}^m(\vartheta_i) F_m(\varphi_j)\,.$$

Both constructions, of f^{lm} from f_{ij} and conversely, are $O(L^3)$ operations, due to the matrix multiplications in (3.27), (3.28). Routines for transforming between grid values and spherical harmonic coefficients have been implemented for maximum angular momentum $L = 7, 15,$ and 31. The grid values $\overline{P}^{lm}(\vartheta_i)$ which appear in the sum (3.27) were precomputed in multiple precision using REDUCE [32]. The functions $\overline{P}^{lm}(\vartheta)$ defined by (3.26) were constructed symbolically using exact inversion of the matrices $A_{(m)\,ll'}$. This symbolic approach is feasible because the metric $G_{lm\,l'm'}$ factorizes as the tensor product (3.20), thus allowing exact inversion of G using matrices of size at most $(L+1) \times (L+1)$ rather than $(L+1)^2 \times (L+1)^2$.

The analysis of spin 1 and spin 2 grid functions into spherical harmonic coefficients is similar, but is complicated by the fact that the induced metric on the subspace factorizes as a tensor product only in a complex (mixed parity) basis. Separating the even and odd parity coefficients therefore requires some extra book-keeping.

Techniques for handling spherical harmonic representations have been described by many authors [34, 42, 58, 18, 33]. Our method differs from the Muchenhauer and Daly projection (see [58]) in the choice of inner product (3.19) used to define the orthogonal subspace. These methods are also $O(L^3)$. Jakob [33] gives an $O(L^2 \log L)$ spectral projection, which however bypasses the construction of spherical harmonic coefficients. Because we work with a relatively small value of L, and need the spectral coefficients, the Jakob projection would not provide any improvement.

The torus method described here and in [39] has the advantage that it applies also to higher-rank tensors, in particular vectors and 2-tensors. Representations in terms of spin-weighted fields are more efficient for vectors (spin $s = 1$) than 3-vector representations [59, 60], and the operator $\eth$ gives a transparent derivation of all invariant derivative combinations [59].

3.6. Convolution splines. At various stages it is necessary to interpolate and differentiate grid-based fields. For example, each step of the radial integration of the hypersurface equations by the RK8 method requires interpolation of the source field β at 10 intermediate points; the dynamic regridding of the radial grid uses interpolation to determine the field values of β at the new grid points; and derivatives such as $\partial\gamma/\partial r$ and $\partial Q^+/\partial z$ must be computed from field values on grids. A convolution spline algorithm described in [38] provides a convenient technique.

TABLE 3.2
Convolution coefficients $a_i^{(9)}$.

i:	$1,8$	$2,7$	$3,6$	$4,5$
$a_i^{(9)}$:	$-\frac{67}{2520}$	$\frac{1111}{5040}$	$-\frac{421}{560}$	$\frac{1333}{1260}$

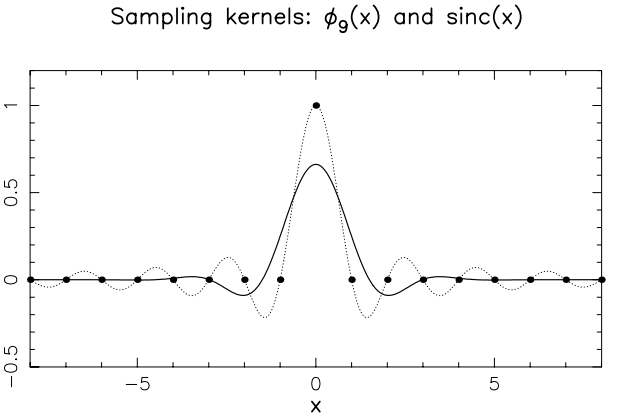


FIG. 3.1. *Comparison of the* C^7 *spline kernel* ϕ_9 *(solid curve) and the sinc method* [17, 57] *kernel* $\mathrm{sinc}(x) = \sin(\pi x)/\pi x$ *(dotted curve). The kernel for a sampling method is the response to the delta-like discrete data (solid dots). The advantages of the convolution spline method are that the kernels have finite support (*$[-8, 8]$ *in this case) and the data is automatically filtered. The convolution* (3.29) *using kernel* ϕ_9 *exactly reproduces polynomials of degree* 7 *or less.*

The method has the effect of fitting a spline curve to sample data, and is implemented by a convolution of the form [38]

$$\bar{f}(x) = \sum_{k\in\mathbb{Z}} f(k)\phi_n(x-k), \tag{3.29}$$

where the $f(k)$ are the raw data (samples) and $\phi_n(x)$ is a C^{n-2} sampling kernel. ϕ_n is constructed as a certain sum of central B-splines M_n of order n,

$$\phi_n(x) = \sum_{i=1}^{n-1} a_i^{(n)} M_n(x - \tfrac{n}{2} + i), \tag{3.30}$$

where the coefficients $a_i^{(n)}$, $i = 1, \ldots, n-1$ are chosen so that the convolution (3.29) acts as the identity on polynomials $f(x)$ of degree $n-1$ (n even) or $n-2$ (n odd). Recall that the central B-spline $M_n(x)$ is a C^{n-2} piecewise polynomial of degree $n-1$ normalized by $\sum_{k\in\mathbb{Z}} M_n(x-k) = 1$, with support on $|x| \le n/2$ [52]. The support of the kernel $\phi_n(x)$ is therefore $|x| \le n-1$.

Algorithms for computing the $a_i^{(n)}$ and tabulations for $n \le 11$ are given in [38]. Coefficients for the kernel ϕ_9 used in the code are given in Table 3.2, and ϕ_9 is plotted in Figure 3.1. The expressions (3.29), (3.30) may be rearranged into a form which is more efficient for numerical calculations,

$$\bar{f}(x) = \sum_{k\in\mathbb{Z}} \tilde{f}_k M_n(x - \tfrac{n}{2} - k), \tag{3.31}$$

where the modified sample values $\tilde{f}_k$ are given by

$$\tilde{f}_k = \sum_{i=1}^{n-1} f(k+i) a_i^{(n)}.$$

The advantage of (3.31) is that the $\tilde{f}_k$ can be computed once and then reused to evaluate $\bar{f}(x)$ at many different points x, using the explicitly known values of the B-spline $M_n(x)$. The same $\tilde{f}_k$ values may also be used to compute derivatives of the spline function $\bar{f}$,

$$\bar{f}^{(j)}(x) = \sum_{k\in\mathbb{Z}} \tilde{f}_k M_n^{(j)}(x - k - \tfrac{n}{2}), \tag{3.32}$$

where again the derivatives $M_n^{(j)}(x)$ are known functions. These techniques are routinely used to supply intermediate values and derivatives of fields in the radial and time directions.

Nonuniform distributions of sample points are handled by a mapping between the independent variable and the sample number variable (x in the above). Numerical derivatives are then calculated using the chain rule. For example, the radial grid described in section 4.2 is nonuniform, specified by some known relation of the general form $r = r(n)$ (where n is now being used to denote the sample number variable, with radial grid points being at $n = 0, \ldots, n_\infty$). The operator $\frac{\partial}{\partial r}$ is then implemented as $\left(\frac{dr}{dn}\right)^{-1} \frac{\partial}{\partial n}$, with a formula for the first factor being known explicitly.

Similarly, one can transform to an independent variable $s = s_0 + hx$ which has grid spacing h, to examine the behavior of the approximation (3.29) as $h \to 0$. Let $g(s) := f((s - s_0)/h) = f(x)$, so $g^{(j)}(s) = h^{-j} f^{(j)}(x)$. It can be shown [38] that using the ϕ_9 kernel, the Taylor series truncation errors for (3.29) at a grid point s are

$$|\bar{g}^{(j)}(s) - g^{(j)}(s)| = c_j h^8 |g^{(8+j)}(s)| + O(h^9), \quad j = 0, 1, 2, \tag{3.33}$$

where $c_0 = \frac{2021}{134400}$, $c_1 = \frac{4547}{302400}$, $c_2 = \frac{4549}{302400}$. This reflects the fact that convolution with ϕ_9 is exact on polynomials of degree 7.

The predicted h^8 convergence of the ϕ_9 spline convolution is clearly evident in Figure 3.2. Here the function $v(x) = e^x \sin 10x$ has been approximated at varying grid resolutions corresponding to $N = 2^p$ grid points over the interval $[-1, 1]$. Convolution splines do not generally preserve sample values, except for samples from polynomials of degree less than or equal to the degree of reproduction. This results in some damping of high-frequency components of the data, which we expect helps to suppress numerical noise and algorithmic instabilities.

Within the context of spectral methods for PDEs, the direct filtering of Fourier coefficients of a numerical solution is common practice and has been extensively studied (cf. [19, section 8.3] and the references therein). On the other hand, explicit use of a digital filter [30] in conjunction with finite difference methods is comparatively rare. Nevertheless, from an algorithmic point of view, this is the effect of using a convolution spline.

The filtering inherent in the method can be examined via the response function

$$\Phi_n(\theta) = \sum_{k\in\mathbb{Z}} \phi_n(k) \cos k\theta, \quad 0 \le \theta \le \pi, \tag{3.34}$$

which is equal to the factor by which the Fourier mode $e^{\mathrm{i}\,\theta x}$ is amplified by the approximation (3.29) at a grid point $x \in \mathbb{Z}$. The value $\theta = \pi$ corresponds to the Nyquist

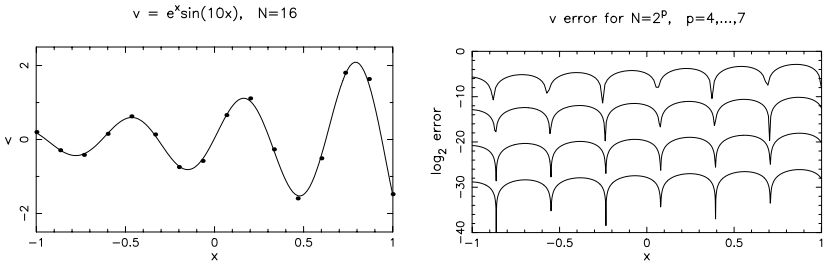


FIG. 3.2. *Convergence of the ϕ_9 spline convolution:* (a) *samples of $v(x) = e^x \sin 10x$ at $N = 16$ points over $[-1, 1]$, and the corresponding convolution spline;* (b) *logarithmic plots of the absolute error $|v(x) - \bar{v}(x)|$ for grid resolutions of $N = 2^p$ with $p = 4, \ldots, 7$, showing a reduction in the error by a factor of 2^8 on each doubling of the resolution.*

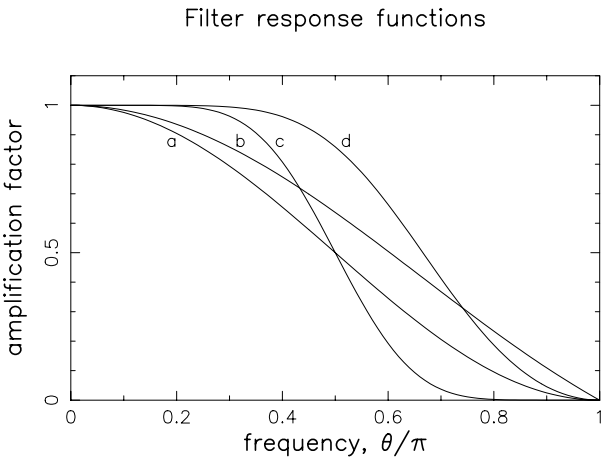


FIG. 3.3. *Filter response functions:* (a) *raised cosine filter $\sigma(\theta) = \frac{1}{2}(1 + \cos\theta)$;* (b) *Lanczos filter $\sigma(\theta) = \theta^{-1} \sin\theta$;* (c) *sharpened raised cosine* [19]*;* (d) $\Phi_9(\theta)$.

frequency for the grid. Figure 3.3 shows the response function $\Phi_9(\theta)$, compared to the well-known Lanczos and raised cosine (artificial viscosity) filters [30]. The filtering characteristics of convolution splines and their derivatives are described in [38]. The limitations of convolution splines are illustrated in Figure 3.4, which shows Gibbs-like effects associated with approximation of step-like data. The figures also give an indication of the number of grid points needed to resolve a sharp transition. In order that the convolution spline smoothing should introduce only negligible errors, the grid resolution should be chosen sufficiently fine that typical field variations take place over enough grid points that the expected frequency θ of the field data lies well within the part of the Nyquist frequency interval where $\Phi_n(\theta) \approx 1$. Of course this requires some prior knowledge of the length scale of the field, and cannot be applied where shocks (or arbitrarily rapid variations) occur in the data. In such cases the spherical harmonic representation would become equally unsuitable.

The choice of high-order convolution splines (h^8 rather than say h^4) was motivated by the need to reduce storage requirements. The smoothing associated with low-order spline convolutions results in a markedly reduced usable proportion of the Nyquist interval [38], so to achieve comparable accuracy with a low-order method would require

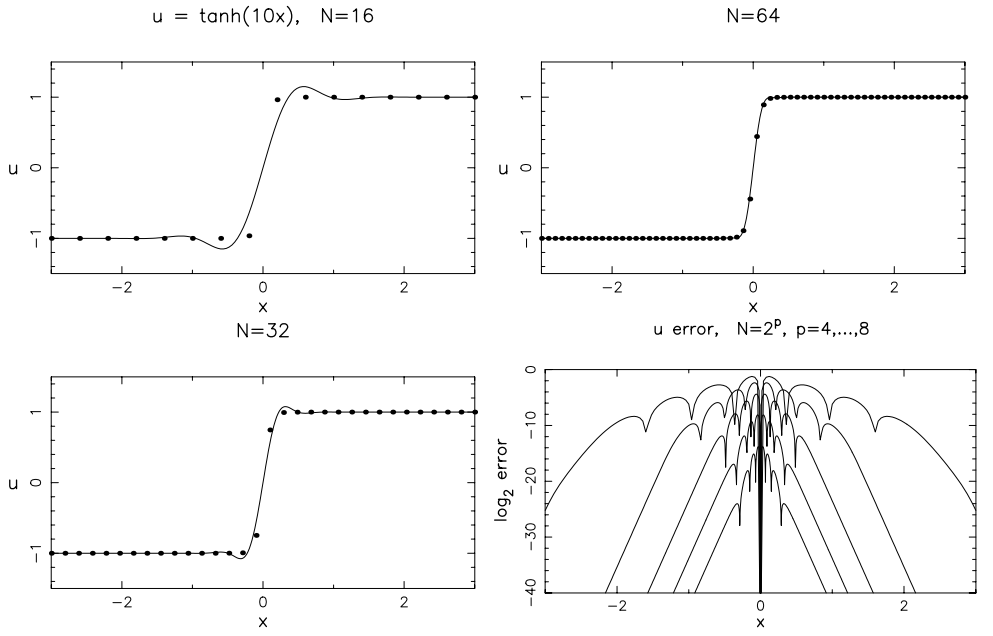


FIG. 3.4. *Convolution spline approximation of step-like data. Here the function* $u = \tanh(10x)$ *is represented by* N *samples over the interval* $[-3, 3]$. *For* $N = 16$, *the transition from* $u \approx -1$ *to* 1 *takes just one grid interval and a Gibbs-like phenomenon is evident. The convolution splines are constructed using the* $\phi_9(x)$ *kernel. The logarithmic plot shows the absolute error,* $|u(x) - \bar{u}(x)|$.

significantly higher grid point densities. The ϕ_9 kernel was chosen also because its accuracy matches that of the RK8 method used for the radial integration.

To use convolution splines near endpoints of a data set we extend the data set, using a suitable mapping between the independent variable and the sample number variable [40]. The mapping is chosen so that when expressed in terms of the sample number variable n, $0 \le n \le n_\infty$, the fields admit expansions in powers of n^2 near $n = 0$ and $(n_\infty - n)^2$ near $n = n_\infty$. The sampling kernel convolution is then applied to the even extension of the fields through $n = 0$ or $n = n_\infty$. This technique is particularly important in extracting radiation data near null infinity (scri, $\mathcal{I}^+$), where the radial grid is chosen so that $n_\infty - n = O(r^{-1/2})$, as described in section 4.2.

4. Solution algorithm. The hypersurface equations (2.20)–(2.23) suggest the following process for evolving the metric in the exterior region with interior boundary the cylinder $r = r_0$:

1. Choose boundary data (H, Q^-, J, K) on the cylinder $r = r_0$, consistent with the boundary equations (2.24), (2.25);

2. Assume β is given on a null hypersurface $\mathcal{N}_z$;

3. Solve the $\mathcal{N}_z$ hypersurface equations $r\partial_r U = F(\beta, U)$ by integrating along $(z, \vartheta, \varphi) = const.$ with initial conditions at $r = r_0$ determined in step 1;

4. Reconstruct the metric functions u, v, γ from H, J, K, and β using the converse construction (2.13)–(2.15);

5. Reconstruct $\partial\beta/\partial z$ from Q, using (2.16) and the known values of β, γ on $\mathcal{N}_z$;

6. Use $\partial\beta/\partial z$ from step 5 to evolve β to the "next" null hypersurface $\mathcal{N}_{z+\Delta z}$ and repeat from step 3.

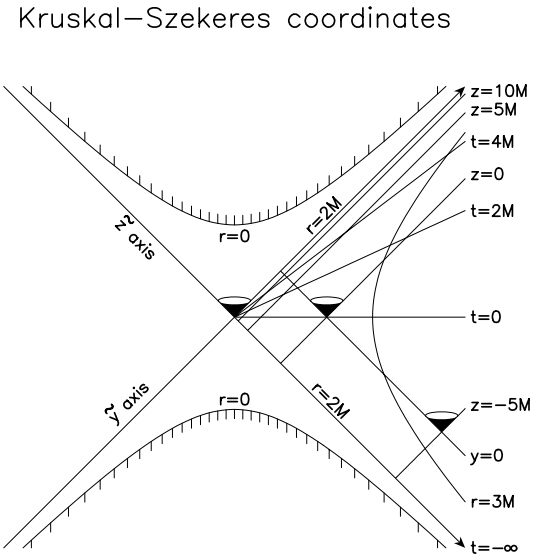


FIG. 4.1. *Schwarzschild spacetime in Kruskal–Szekeres coordinates. Radial light rays are at* 45°.

In the following we will show how this heuristic algorithm is implemented numerically, using the techniques and data representations of the previous section.

4.1. Geometry and inner boundary conditions. The code models gravitational waves propagating on a black hole spacetime, with metric approximating that of the Schwarzschild solution in the Kruskal–Szekeres coordinates [31]. Introducing the double-null coordinates

$$z = t - r^*, \qquad y = t + r^*, \qquad r^* = r + 2\mathrm{M}\log\left(\frac{r}{2\mathrm{M}} - 1\right),$$

the Schwarzschild metric (2.2) becomes $ds^2_{Schw} = -(1 - 2\mathrm{M}/r)\,dy\,dz + r^2 d\Omega^2$, where $d\Omega^2 = d\vartheta^2 + \sin^2\vartheta\, d\varphi^2$. The coordinate singularities at the past and future horizons $t = \pm\infty$ are removed by defining $\tilde{y} = e^{y/4\mathrm{M}}$, $\tilde{z} = e^{-z/4\mathrm{M}}$, giving the metric

$$ds^2_{Schw} = \frac{32\mathrm{M}^3}{r} e^{-r/2\mathrm{M}}\, d\tilde{y} d\tilde{z} + r^2\, d\Omega^2\,,$$

where $r = r(\tilde{y}, \tilde{z})$ is defined implicitly by (see Figure 4.1)

$$e^{r/2\mathrm{M}}\left(\frac{r}{2\mathrm{M}} - 1\right) = \tilde{y}\tilde{z}. \tag{4.1}$$

The surfaces $\tilde{z} = 0$ and $\tilde{y} = 0$ (i.e., $r = 2\mathrm{M}$) form the past and future event horizons, and these are smooth hypersurfaces with bounded curvature. The approximate Minkowski structure of Schwarzschild spacetime is better illustrated by the radial null geodesics in (r, t) coordinates; see Figure 4.2. Note however that the (r, t) coordinates are singular along the event horizons $r = 2\mathrm{M}$. Initial conditions for β are imposed on $\{z = 0, r \geq 2\mathrm{M}\}$ (with $\mathrm{M} = 1$ usually), by specifying the spherical harmonic coefficient functions $\beta_{lm}(r)$. The initial coefficient functions $\beta_{lm}(r)$ may be freely chosen, subject only to the size condition (2.17).

For simplicity the inner boundary conditions are set at $r_0 = 2\mathrm{M} = 2$ to agree with the Schwarzschild past horizon: $H_0 = 2$, $Q_0 = J_0 = K_0 = 0$. Since we choose

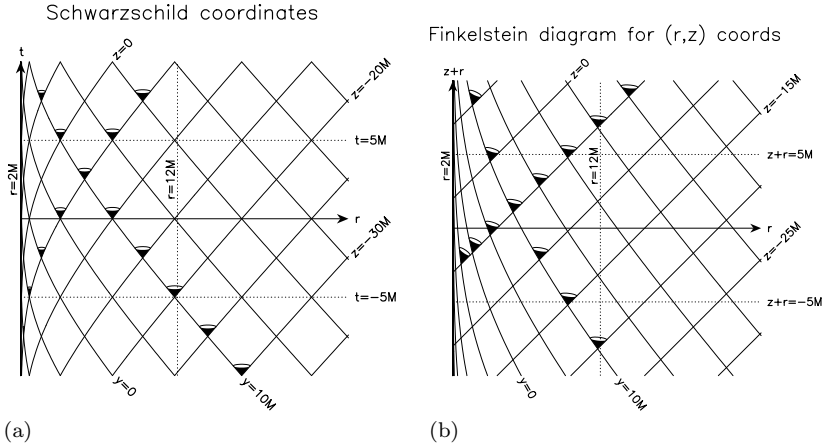


FIG. 4.2. *Exterior region $r > 2\text{M}$ of Schwarzschild spacetime, with radial null geodesics.* (a) *In Schwarzschild coordinates (r, t);* (b) *in retarded Eddington–Finkelstein coordinates (r, z).*

$\beta(0, r) = 0$ for $2 \leq r \leq 5$, by causality the solution should agree with the Schwarzschild metric in a neighborhood of $r = 2$ for all time $z \geq 0$, producing a "white hole" past horizon in the spacetime. This choice of inner boundary condition has the considerable advantage that the boundary equations (2.24), (2.25) are automatically satisfied, and it is not necessary for this class of simulations to separately ensure that the boundary data are numerically compatible with the boundary equations.

The resulting spacetimes have the geometry of an isolated black hole with future event horizon at $z = \infty$, $r > 2\text{M}$, and Schwarzschild-like white hole boundary along $r = 2\text{M}, 0 \leq z < \infty$. Adding shear β at the initial hypersurface $z = 0$ results in spacetimes with gravitational radiation interacting with a single black hole.

Although the fixed past horizon boundary conditions used in the present code allow many interesting issues to be addressed, it would be desirable to implement more general inner boundary conditions. Such conditions specify H, J, K, Q^+ at an inner surface ($r = 1$, for example), subject to the dynamical ($\partial/\partial z$) constraints on the evolution of J/u and Q^+ determined by the boundary equations (2.24),(2.25) [4]. The free data on the inner boundary consist of u_0, K_0, where u_0 represents a certain coordinate gauge freedom, while K_0 describes the gravitational radiation injected into the system through the inner boundary. Various exact solutions with such boundary conditions are described in [6] (Robinson–Trautman [49], boosted Schwarzschild, twisted Minkowski space), and would provide useful accuracy checks on the numerical methods. However, implementing general inner boundary conditions raises numerical difficulties, and constraining the radiation data K_0 such that the spacetime is still Schwarzschild near the past horizon is a difficult geometric problem. An arbitrary choice of K_0 will inject additional energy into the spacetime.

4.2. Dynamic radial grid. There are two geometric features which the code should model accurately: future null infinity ("scri" or $\mathcal{I}^+$, where $r \to \infty$, z finite), and the future horizon $r \sim 2\text{M}$, $z \to \infty$.

The field near null infinity $\mathcal{I}^+ \cap \mathcal{N}_z = (r = \infty, z)$ determines the outgoing gravitational waves as seen by a distant observer and is consequently very important for applications to gravitational wave astronomy. Experience with 3+1 codes shows that it is not possible (as yet) to provide boundary conditions on an outer time-like

boundary at a finite radius which do not either inject radiation or reflect radiation back into the grid. This deficiency has the effect of severely limiting the overall time duration of most $3+1$ simulations. We avoid all such reflection problems by using a radial grid coordinate n, which compactifies $r=\infty$ and leads to accurate modeling of gravitational radiation.

The (z,r) coordinates become singular near the future horizon $z\to\infty$ in the Schwarzschild spacetimes (see Figures 4.1, 4.2). In our case this picture is not exact, since the spacetime geometry is only approximately Schwarzschild, and thus the future horizon will not be located exactly at $z=\infty$. However, the NQS parameterization must still become singular eventually, as the outgoing null hypersurfaces $\mathcal{N}_z$ approach the future event horizon.

One effect of nearly singular coordinates (r,z) at late times is that the in-falling gravitational features near the event horizon will be compressed into a region of small r-variation, and this compression will accelerate in time z, while retaining field structures from early times. Consequently no r-grid which is constant in time is able to accurately represent the in-falling radiation at late times. We have observed that numerical problems with a fixed radial grid arise as early as $z=10$.

To overcome these problems, a dynamic and variable radial grid is used, based on double null coordinates $(z,\tilde{y})$. The time steps in the evolution direction are regular, with $\Delta z = 0.1, 0.05, 0.025$ being typical. The grid in the radial direction is chosen to satisfy the criteria that it compactify null infinity and concentrate grid points in the region of greatest variation in the seed field β. Because the field features propagate along the inward and outward null characteristics, which correspond, respectively, to the curves $\tilde{y}=const.$ (approximately!) and $z=const.$ (exactly), the numerical grid is taken to be rectangular in the $(z,\tilde{y})$ coordinates, with radial grid point positions being determined by an initial distribution of grid points on the surface $z=0$.

Introducing the radial grid coordinate n with range $0\le n\le n_\infty$ (with typical values of n_∞ being 128, 256, 512, and grid points at integer n), we specify an initial grid point distribution $r(z=0,n)=f(n)$, where f is some monotone increasing function such that $f(n_\infty)=\infty$. The radial grid points on the initial ($z=0$, $\tilde{z}=1$) surface have $\tilde{y}$ ordinates given by (4.1),

$$\tilde{y} = (f(n)/2\mathrm{M}-1)\exp(f(n)/2\mathrm{M}) = \phi(f(n)/2\mathrm{M}), \tag{4.2}$$

where $\phi(x):=(x-1)e^x$ is monotone and invertible for $x\ge 0$. Since $\tilde{y},z,r$ are related by $\tilde{y}=e^{z/4\mathrm{M}}\phi(r/2\mathrm{M})$ and the grid points are required to inflow along the curves of constant $\tilde{y}$, we can determine the dynamic radial grid point distribution $r=r(z,n)$ in terms of the initial grid distribution function $f(n)$ by

$$r(z,n) = 2\mathrm{M}\,\phi^{-1}(\exp(-z/4\mathrm{M})\phi(f(n)/2\mathrm{M}))\,, \tag{4.3}$$

where the inverse function $\phi^{-1}:[-1,\infty)\to[0,\infty)$ is evaluated numerically. With this definition, the surfaces $n=const.$ correspond to in-falling null hypersurfaces in the reference Schwarzschild metric. To express the hypersurface equations in terms of n rather than r, we compute $\partial r/\partial n$ from (4.3),

$$\frac{\partial r}{\partial n} = e^{-z/4\mathrm{M}}\frac{f(n)}{r(z,n)}\exp\left(\frac{f(n)-r(z,n)}{2\mathrm{M}}\right)\frac{df}{dn}. \tag{4.4}$$

It remains to choose the initial grid distribution $f(n)$. The condition $n_\infty - n = O(r^{-1/2})$ is achieved by setting $f(n)=f_1(\nu)/(1-\nu)^2$, where $\nu=n/n_\infty$ and f_1 :

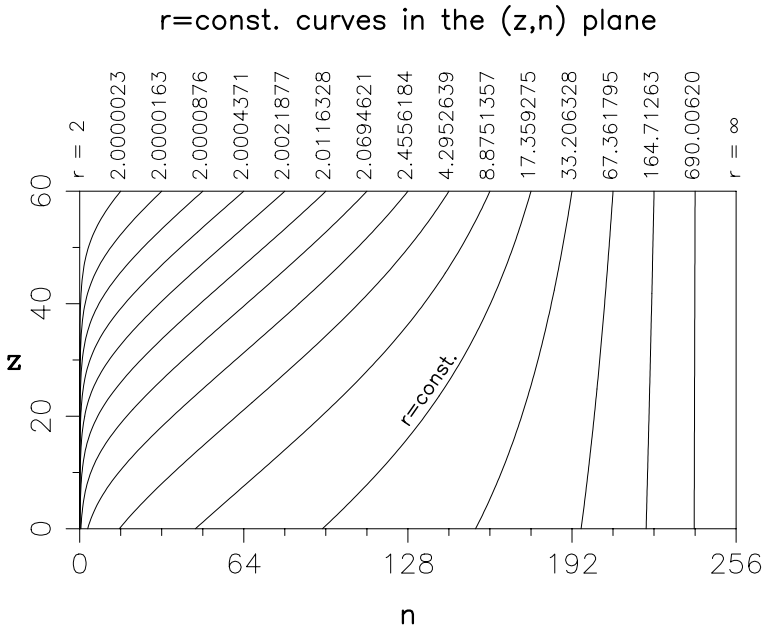


FIG. 4.3. *The map between radial grid point number n and radius r is dynamic, chosen so that grid points approximately follow inward null geodesics. At late null-time z, grid points cluster near the black hole horizon at $r = 2$.*

$[0,1] \to \mathbb{R}$ is any suitable smooth monotone bounded function. In the code, f_1 is a quadratic polynomial, with coefficients chosen to concentrate grid points across the support of the chosen initial data $\beta(z = 0)$. Figure 4.3 shows sample curves $r(z, n) = const.$, illustrating the in-falling nature of the (z, n) grid coordinates. This heuristic prescription for distributing the grid points works well in practice—Figure 4.4 shows the shear over the (z, n) plane for `run_160` and clearly demonstrates the in-falling structure of this solution. The simulation eventually terminates at $z = 55$ because of some geometric effect associated with breakdown of the NQS gauge condition near the future event horizon.

4.3. Hypersurface equations. The hypersurface equations are solved by treating them as a large system of ODEs, with the radial grid coordinate n playing the role of independent variable, and the dependent variables being the values taken by the fields (H, J, Q, K) at the $N^2/2$ points of the (ϑ, φ)-grid.

The form ((2.28)–(2.31)) of the hypersurface equations, for the variables $\log H$, rQ^+, j and K, proves to be better behaved near $r = \infty$, since each of these variables has a finite (usually nonzero) limit. Integration of these radial ODEs is possible up to and including the final point $n = n_\infty$, with results whose numerical effectiveness may be seen by inspecting the field values in a neighborhood of null infinity [8]. Tests described in the following section, in particular the consistency of the constraint equations and the accuracy of the Trautman–Bondi mass decay formula (Figure 5.9) also confirm that asymptotic behavior has been reliably calculated.

Note that unlike methods based on Bondi–Sachs or Newman–Unti coordinates [29, 37], integration along the r-coordinate lines does not correspond to integrating along the radial null geodesics (the characteristics of the Einstein equations), since in general the NQS shear β is nonzero and the null direction is $\ell = \partial/\partial r - r^{-1}\beta$.

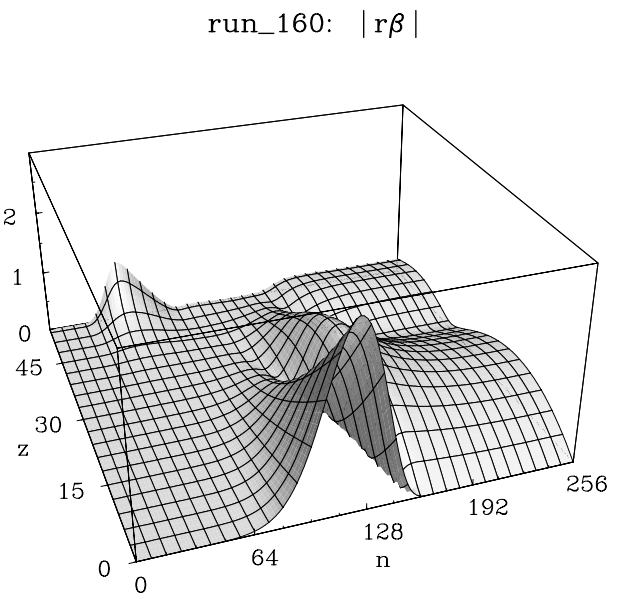


FIG. 4.4. *Evolution of $r\beta$ for $0 \le z \le 55$, in radial in-falling coordinates. Observe that the in-falling grid tracks the dynamical evolution. This simulation has timestep $\Delta z = 0.05$, $n = 0$ is the past horizon $r = 2\text{M}$, and $n = 256$ represents future null infinity $\mathcal{I}^+$.*

The radial integration with respect to n is performed using an RK8 scheme [46], with RK step size $\Delta n = 1$. This method requires 13 derivative evaluations per RK step, of which 10 are at intermediate points not on the radial grid. Values of β and its angular derivatives at these intermediate points are provided by convolution splines generated using the kernel ϕ_9 with samples at integer n.

4.4. Reconstructing the metric. Step 4 of the solution algorithm requires us to reconstruct the metric functions (u, v, γ) from the solution (H, J, K, Q) of the hypersurface system ((2.20)–(2.23)) with seed β and boundary data $(H, J, K, Q)|_{r=r_0}$. The reconstruction is carried out as described in section 2.3. This process requires solving the system (2.14) on each S^2 of the radial grid. If β is not too large, then (2.14) is an elliptic system of PDEs on the sphere S^2, mapping surjectively to the space of spin 2 fields. We solve (2.14) by introducing the preconditioning variable $\Gamma = \eth\gamma$, so (2.14) becomes

$$\Gamma + \frac{\eth\beta}{2 - \text{div}\beta}\text{div}(\eth^{-1}\Gamma) = -K + J\frac{\eth\beta}{2 - \text{div}\beta}\,, \tag{4.5}$$

where $\eth^{-1}$ is defined spectrally by

$$\eth^{-1}Y^2_{lm} = -\left[\tfrac{1}{2}(l+2)(l-1)\right]^{-1/2} Y^1_{lm}, \quad l \ge 2.$$

Note that this definition of $\eth^{-1}$ gauges the $l = 1$ spherical harmonic components of γ to zero—a similar but more expensive construction may be used if nonzero $\gamma_{l=1}$ components are desired. The advantage of (4.5) over (2.14) is that the operator in (4.5) is close to the identity for small $B := \eth\beta/(2 - \text{div}\beta)$.

Writing (4.5) as $(1+\mathcal{A})\Gamma = F$, where $\Gamma \mapsto \mathcal{A}\Gamma = B\text{div}\eth^{-1}\Gamma$ is an $\mathbb{R}$-linear operator

and $F = -K + JB$, suggests the iterative algorithm

$$\Gamma_{n+1} = F - \mathcal{A}\Gamma_n, \quad n = 0, 1, \ldots. \tag{4.6}$$

For $\Gamma = \sum_{l\geq 2,m} \Gamma^{lm} Y^2_{lm}$ we have the explicit spectral representation

$$\mathrm{div}\eth^{-1}\Gamma = -\sum_{l=2}^{\infty}\sum_{m=-l}^{l}\left[\frac{l(l+1)}{(l-1)(l+2)}\right]^{1/2}(\Gamma^{lm} + \bar{\Gamma}^{lm})Y_{lm}, \tag{4.7}$$

which shows that $\mathrm{div}\eth^{-1}$ is a zero-order operator with kernel consisting of purely odd spin 2 fields. It follows that (4.6) converges for B small.

The action of $\mathcal{A}$ is evaluated using (4.7) to compute $\mathrm{div}\eth^{-1}\Gamma$, then using the field value representation to compute the product with $B = \eth\beta/(2 - \mathrm{div}\beta)$, and finally converting back to the spectral representation. The iterations (4.6) continue until the error, measured by the sum of squares of spherical harmonic coefficients of the difference of the two sides of (4.5), is 10^{-2} times the size of the aliasing error in the source term. This aliasing error is the difference between the raw field values of the source term (which is necessarily calculated in the field value representation because it involves products and quotients) and its field values after projection into the subspace spanned by spin 2 spherical harmonics. It provides an estimate of the error in the source term, and hence (because the operator $1 + \mathcal{A}$ is close to the identity) it is reasonable to accept a solution of comparable accuracy.

The iteration scheme (4.6) turns out to be quite efficient, typically requiring fewer than 10 iterations for an S^2 grid of size $N/2 \times N = 16 \times 32$. On this size grid we resolve all components of Γ up to angular momentum $L = N/2 - 1 = 15$, so in this case we solve for $2((L+1)^2 - 4) = 504$ spectral coefficients.

Solving for Γ in the spectral representation uses fewer unknowns $(2(L+1)^2 - 8$ compared to $4(L+1)^2$ grid values), and gauge conditions which specify the $l = 1$ components of γ (e.g., $\gamma_{l=1} = 0$) can be directly implemented. It is possible to adapt the algorithm to allow for other NQS gauges (e.g., $\beta_{l=1} = 0$), which could have some geometric advantages [6], but this is numerically more expensive since (2.14) must be solved four times at each sphere rather than once.

4.5. Evolution. Given β on a null hypersurface $\mathcal{N}_z$, we construct the time derivative $\partial\beta/\partial z$ by solving the hypersurface equations with seed β, determining γ, v as outlined in the previous section, and then using formula (2.16) to evaluate $\partial\beta/\partial z$. Let us write the result of this process as

$$\frac{\partial\beta}{\partial z} = \mathcal{B}(\beta, U_0), \tag{4.8}$$

where the operator $\mathcal{B}$ is determined by the value of β on the hypersurface $\mathcal{N}$ and the initial conditions $U_0 = (H_0, Q_0, J_0, K_0)$ at $r = r_0$ for the hypersurface equations.

The evolution formula (4.8) provides the basis of the spacetime evolution algorithm, which incorporates (4.8) into a standard RK4 algorithm. This is just the method of lines, treating the evolution equations as a large system of ODEs for the (r, ϑ, φ)-grid values of $\beta(z) = \beta_{|\mathcal{N}_z}$.

The method of lines, applied blindly in this manner, is generally prone to instabilities. Tests suggest the relative stability of the NQS code derives from the smoothing effects (a) of the convolution spline and (b) of the spectral projection. The filtering implicit in the convolution spline is applied to β during the radial integration of the

hypersurface equations, at each of the four stages of the RK4 algorithm. It is not possible to turn off this radial filtering because the convolution splines for β are an essential part of the algorithm for evaluating the right-hand side (RHS) of (4.8).

Smoothing of β in the angular directions is done explicitly, by projecting β onto the spin 1 subspace with maximum angular momentum L or $2L/3$ (the Orszag 2/3 rule, to eliminate quadratic aliasing). This angular filtering is done after each of the four stages of the RK4 algorithm. Removal of the angular filtering results in very rapid disintegration of the evolution, which then typically lasts only a few integration steps.

For simplicity, the four RK4 stages evolve β in the z direction in the (z, r) coordinates, along $r = const$. At the end of each full RK4 time step the key field β is interpolated onto the new radial grid (4.3) using a convolution spline for β.

The RK4 time integration of β evolves field values on the (r, ϑ, φ)-grid. Equivalently, we could have evolved its spherical harmonic coefficients, of which there are half as many. However, the computation saved by doing so is insignificant in comparison to that required to evaluate $\partial\beta/\partial z$, so this choice is made for convenience.

Likewise, the RK8 radial integration of the system of hypersurface equations uses the field value representation. In this case, however, it is found that projecting the fields onto their appropriate spherical harmonic subspaces during the integration is not required for either stability or accuracy. There is a definite computational advantage in staying within the field value representation, since many relatively expensive $O(L^3)$ projections are avoided.

Evaluating $\partial\beta/\partial z$ requires $\partial\gamma/\partial r$ (2.16). Numerical r-derivatives of γ are calculated as the derivatives of convolution splines for γ in the radial direction, making use of formula (4.4) and the chain rule for derivatives. The radial derivative term $\mathcal{D}_r \log u$ which appears in the hypersurface equation (2.29) can be expressed using (2.28) and (2.13) in terms of the 1st radial and angular derivatives of β.

The program is normally run until the solution ceases to be well behaved. Blowup is detected by monitoring $2 - \mathrm{div}\beta$, which must remain everywhere positive. For the initial data that we have used, the blowup always occurs in $l = 2$ modes of β, at low n values corresponding to $r \approx 2\mathrm{M}$ (see Figures 4.3, 4.4). Although the precise cause of blowup is not yet understood, it is not a numerical instability, since it is unaffected by changes in radial or timestep resolutions, nor does it appear to be primarily geometric, since most curvature scalars remain bounded. This suggests the blowup is a coordinate effect arising from proximity to the future event horizon.

For smooth initial data of intermediate strength, the evolution extends to $z \sim 55$. The final time varies with the strength of the initial data—see Table 5.1. The evolution of an intermediate strength solution is shown in Figure 4.4, which plots the mean square or $L^2(S^2)$ size of β at each radial sphere, for time $0 \le z \le 55$. The blowup feature is at low radius, and is apparent from time $z = 45$ onwards.

5. Accuracy tests. The complexity of the NQS Einstein equations and the variety of algorithms employed in the code make it problematic to prove rigorously that the simulations accurately model the physics and geometry of the spacetime. Instead we rely on a range of tests to justify the reliability of the code, probing the numerical accuracy of the solutions through their convergence and geometric consistency.

We consider here tests based on the *numerical convergence* of the solutions as algorithmic parameters are varied; and on the *algebraic consistency* of the numerical solutions. The consistency tests measure the constraint identities and the Trautman–Bondi mass decay formula [63, 29].

TABLE 5.1
Sizes of the example initial data sets

Field strength:	weak	intermediate	strong
$\beta(0)$ scale factor:	1	4.48	10
$m_B(0)/\mathrm{M} - 1$:	0.9472×10^{-2}	0.1915	0.9940
Last z:	61	55	51

The resolution of the simulations is determined by three parameters: the spherical harmonic spectral limit L (or effective limit $l_{\max}$); the number of radial zones n_∞; and the time step Δz. We shall examine in turn how the accuracy of a solution depends on each of these parameters.

It is clear that numerical convergence can be estimated from the convergence properties of the key field β. However, convergence of β guarantees only that the (limit) solution satisfies *some* system of equations, which may not coincide with the desired vacuum Einstein equations. (For example, the Einstein equations may have been incorrectly implemented.) Thus, to assert that the correct equations have been solved, it is essential to provide independent tests of the correctness of the code.

The most natural independent test is to compare the numerical solution with an explicitly known solution. Unfortunately the Schwarzschild metric (2.2) is trivial in the NQS gauge and does not provide a useful comparison test, while the twisted shear-free metrics [6] require interior boundary conditions which are more general than those available in the present version of the code.

Instead we consider here another class of independent tests based on constraint relations. Such relations are typical of geometric equations arising in geometry and physics, which admit gauge and coordinate freedoms. Thus, we check the geometric consistency of the solution by evaluating $r^2 G_{nn}$ and $r^2 G_{nm}$ using (2.24) and (2.25). Neither of these relations is used in generating the numerical solutions, and in theory these components should evaluate to zero. In practice, since each is a sum of terms having magnitude approximately $|\beta| \sim 1$, the extent to which $r^2 G_{nn}$, $r^2 G_{nm}$ evaluate to zero serves both to confirm the consistency of the numerical solution with the vacuum Einstein equations, and also to assess the relative accuracy of the solution.

The Trautman–Bondi mass decay formula provides another such test of geometric consistency, and of the accuracy of the solution near $r = \infty$. This theoretical result is a relation between the asymptotic ($r = \infty$) values and z-derivatives of the fields H, J, and K, and may be readily tested for our numerical solutions.

In the following we discuss numerical solutions generated by three reference initial $\beta(z = 0)$ fields, which differ only by the scale factors in Table 5.1. In each case the initial β consists of pure $l = 2, m = 2$ spherical harmonics with equal strength odd and even parts, and radial profile being a bump supported on $5 \le r \le 40$. We use the terms *weak*, *intermediate*, and *strong* to describe these solutions.

A convenient measure of the strength of the gravitational field is the initial relative mass difference $m_B(0)/\mathrm{M} - 1$, between the initial Bondi mass of the numerical spacetime (cf. (5.3)) and the background Schwarzschild mass ($\mathrm{M} = 1$). Table 5.1 gives the initial relative mass differences for the three reference initial β fields.

The qualitative conclusions of the error analysis of this section are:

1. The major factor determining the overall accuracy is the spectral limit L. Truncating spherical harmonic coefficients at L has the effect of modifying the equations being solved to a system for which the constraint identities are no longer valid.

With $L = 15$ the weak and intermediate field solutions can be well resolved, but this is not sufficient to obtain adequate (beyond 10^{-3}) accuracy for strong field simulations.

2. Unstable quadratic aliasing can be suppressed using Orszag's 2/3 rule.

3. Within the bounds governed by the spectral limit L, accuracy can be improved by increasing the radial resolution n_∞. For the weak field solution, $n_\infty = 1024$ reduces the radial error to the level of the spectral truncation error (see Figure 5.6(b)).

4. For given resolutions L and n_∞, there is a range of values Δz for which the simulation remains stable. Outside this range, the simulation follows the standard solution for some time, then rapidly blows up. The simulation errors are largely insensitive to the value of Δz within the stable range, so Δz may be chosen as large as possible, consistent with stable evolution.

5.1. Dependence on the spectral limit *L*. Using our current hardware it is not generally feasible to run the code at $L = 31$, and $L = 7$ is too low to be of interest. The code is normally run at the $L = 15$ resolution (16×32 (ϑ, φ)-grid) with an antialiasing cutoff at $l_{\max} = 10$.

Orszag [19, 41] observed that quadratic aliasing can be eliminated by periodically removing the upper 1/3 of the spectral bandwidth of a numerical solution. If fields contain only modes for which $l \leq \frac{2}{3}L$, then a quadratic product is band limited to $l \leq \frac{4}{3}L$. With a working bandwidth L, the modes for which $L \leq l \leq \frac{4}{3}L$ become aliased onto the modes $\frac{2}{3}L \leq l \leq L$. Therefore, truncation at $l_{\max} = \frac{2}{3}L$ will remove quadratic aliasing contamination.

If no $l_{\max}$ cutoff is used (i.e., the full $L = 15$ resolution is retained), then the high l-modes of the intermediate strength simulations blow up at $z \approx 8$. The time until blow up is largely independent of the time step and radial resolution. This suggests that the nonlinear aliasing contamination is best regarded as changing the system into one which has unstable solutions.

Because the nonlinear interactions are predominantly quadratic, it is not surprising that the $l_{\max} = 10$ cutoff is sufficient for long term stability. The intermediate strength solution lasts until $z = 55$, when the code terminates for other reasons.

Figure 5.1(a) shows blow up of `run_453`, an $L = 15$ simulation of the intermediate field strength solution with no antialiasing cutoff. The $l = 15$ modes show rapid growth beyond $z = 6$, indicating the instability of the aliasing feedback. Figure 5.1(b) shows the difference between `run_453` and the stable simulation `run_456`, which has an $l_{\max} = 10$ cutoff. Until the onset of the high l-mode instability (i.e., for $z \leq 6$) there is good agreement between the two simulations, with approximately 10^{-10} relative difference for $l = 2$ modes and 10^{-2} relative difference for $l = 10$ modes. The spectral limit L is critical in determining the relation between gravitational field strength and simulation accuracy. This can be appreciated by observing the decay rate of the l-spectrum of β, as in Figure 5.2. By extrapolation, the absolute error introduced by the antialiasing cutoff at $l_{\max} = 10$ should be no larger than the $l = 10$ coefficient, and a relative error estimate follows by comparing the $l = 10$ and the $l = 2$ coefficients.

Figures 5.2(a) and 5.2(b) show the dramatic difference in decay rates of the l-modes of β for weak and strong fields. Assuming an $l_{\max} = 10$ cutoff, it is evident that the relative errors should be at most 10^{-8} for weak field simulations and 10^{-3} for strong field simulations, assuming the error is dominated by the spectral limit. From the observed decay rate of the l-modes of β for a given field strength, it is possible to estimate the resolution L required to achieve a prescribed accuracy. Thus although we cannot directly investigate the behavior of errors with varying spectral limit L

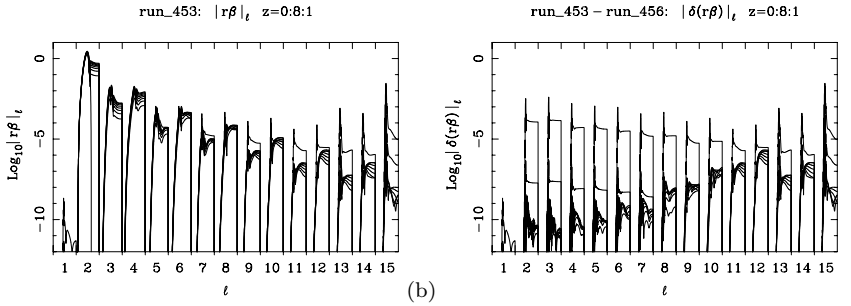


FIG. 5.1. *Orszag's* 2/3 *rule is used to remove aliasing instability:* (a) *unstable evolution of the high* l*-modes of an* $L = 15$ *simulation (no antialiasing cutoff);* (b) *difference between an unstable* $L = 15$ *simulation (no cutoff) and a stable simulation with an* $l_{\max} = 10$ *cutoff. Each* l*-bin contains a radial plot (linear in* n*, with* $n = 0, \ldots, n_\infty$*) of the square root of the sum of the squares of the* (l, m)*-components for fixed* l *with* $m = -l, \ldots, l$*. These simulations have* $n_\infty = 512$ *and* $\Delta z = 0.05$*.*

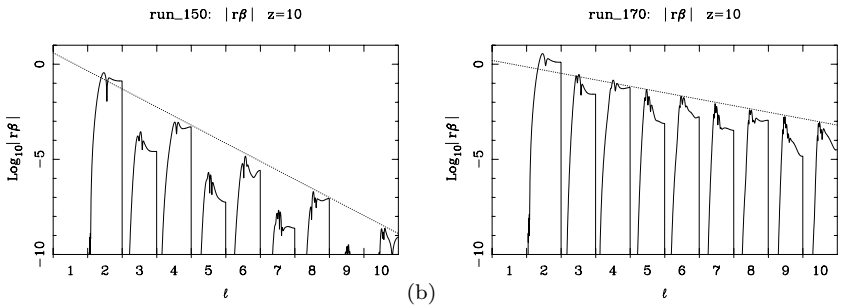


FIG. 5.2. *Spectral resolution and field strength:* (a) *well-resolved weak field with fast* l*-mode decay;* (b) *poorly resolved strong field with slow decay of* l*-modes (see Table* 5.1 *for field strength details).*

(due to hardware constraints), we can still investigate spectral resolution effects by altering the β field strength.

Figures 5.3(a) and 5.3(b) show the effect of field strength (weak, intermediate, strong) on the constraint quantities $r^2 G_{nn}$ and $r^2 G_{nm}$. The parameters for these simulations are $l_{\max} = 10$, $n_\infty = 256$, and $\Delta z = 0.05$. The four curves in each band are for times $z = 10, 20, 30, 40$. There is no significant z dependence of either G_{nn} or G_{nm} until within about 5M of the final blow up time. The second Bianchi identity implies the conservation law $G^{;b}_{ab} = 0$, which leads to a radial system of equations for G_{nn}, G_{nm} with sources linear in the hypersurface Einstein tensor components $G_{\ell\ell}$, $G_{\ell m}$, $G_{\ell n}$, G_{mm}. Thus G_{nn}, G_{nm} give a measure of the accumulated error in the hypersurface equations in the radial direction. This provides some explanation of the structure of the G_{nn}, G_{nm} graphs, particularly for the strong field solution: the numerical solution of the hypersurface equations will have greatest error in the region where the fields are strongest, in this case the range $64 < n < 128$, and this is precisely the region of greatest increase in G_{nn}, G_{nm}.

5.2. Dependence on radial grid resolution n_∞. The radial regridding and interpolation of β, the radial differentiation of γ, and the radial integration of the hypersurface equations are all formally eighth-order accurate. Figure 5.4 shows this is consistent with the observed convergence of β on increasing the radial resolution. The constraint quantities G_{nn} and G_{nm} also exhibit convergence. Figures 5.5(a) and

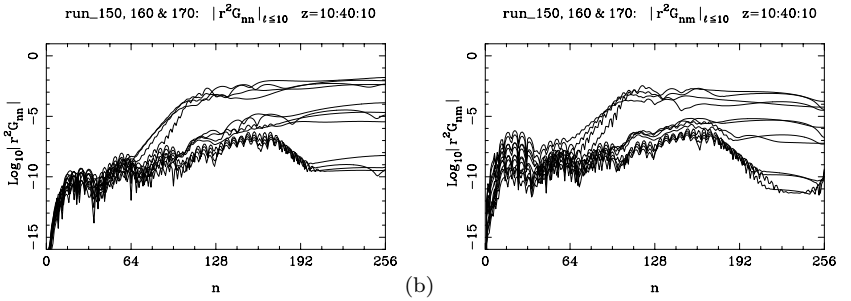


FIG. 5.3. *Effect of spectral resolution on constraint quantities* (a) $|r^2G_{nn}|_{S^2}$*;* (b) $|r^2G_{nm}|_{S^2}$*, at times* $z = 10, 20, 30, 40$ *for strong (top* 4 *curves), intermediate (middle* 4 *curves), and weak fields (bottom* 4 *curves).*

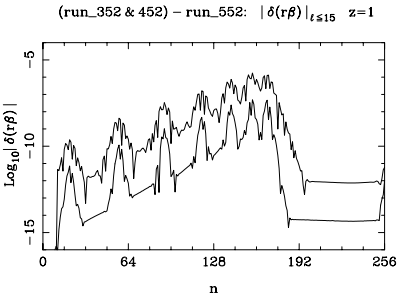


FIG. 5.4. *Convergence of* β *with increasing radial resolution: weak field solutions with* $n_\infty = 256, 512$ *compared to* $n_\infty = 1024$*. The error decreases by approximately a factor of* 2^8 *on doubling the radial resolution.*

5.5(b) show significant improvement between $n_\infty = 256$ and 512, but little between 512 and 1024. The form of the $n_\infty = 1024$ curve indicates that constraint errors at the highest radial resolution are dominated by spectral truncation.

5.3. Dependence on time step Δz. At the typical resolutions at which the code is run, the RK4 errors are completely dominated by errors arising from the spectral truncation L and/or the radial resolution n_∞. This is illustrated by Figure 5.6(a), which shows no significant difference in the constraint quantity G_{nn} between $\Delta z = 0.1$ and $\Delta z = 0.05$ when $n_\infty = 256$. However, when the solution is highly resolved in the radial direction, an effect can be observed, cf. Figure 5.6(b), for $n_\infty = 1024$. Figure 5.7 shows $r\beta$ errors for runs with $\Delta z = 0.1, 0.05$ and $n_\infty = 512$. Again convergence with decreasing Δz is evident only where the RK4 error is not dominated. Consequently, Δz is optimally chosen as large as possible, subject to resulting in stable evolution. For $n_\infty = 256$ and $L = 15$ with an antialiasing $l_{\max} = 10$ cutoff, the evolution is stable for $\Delta z = 0.1$ and unstable for $\Delta z = 0.2$, which blows up at time $z = 25$, after 125 RK4 steps.

5.4. Energy and asymptotic decay tests. The Hawking mass

$$m_H(\Sigma) = \sqrt{\frac{\mathrm{area}(\Sigma)}{16\pi}}\left(1 - \frac{1}{2\pi}\oint_\Sigma \rho_{NP}\mu_{NP}\, dv_\Sigma\right) \tag{5.1}$$

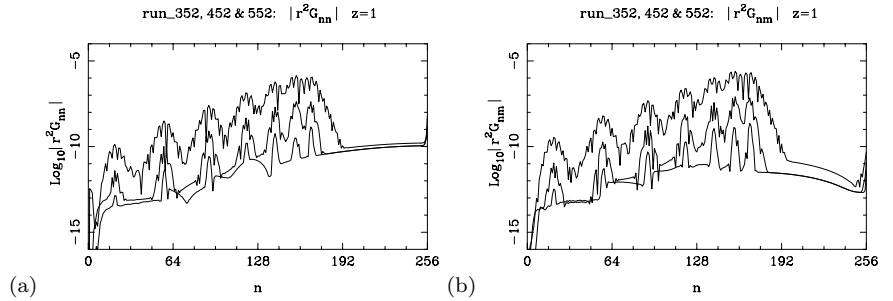


FIG. 5.5. *Effect of radial resolution on weak field constraint quantities* (a) $|r^2 G_{nn}|_{S^2}$, (b) $|r^2 G_{nm}|_{S^2}$. *In each case the three curves are for* $n_\infty = 256, 512, 1024$ *(top, middle, and bottom curves, respectively).*

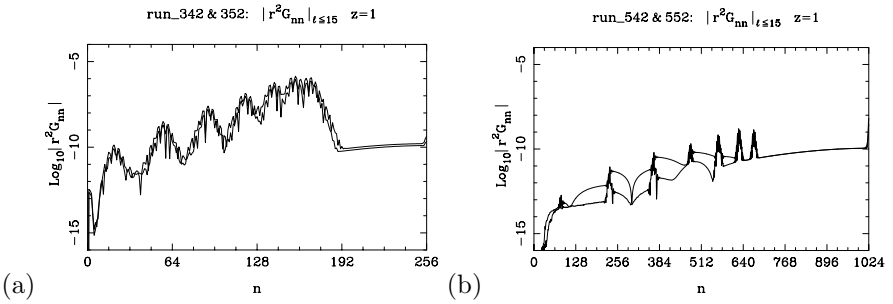


FIG. 5.6. *Effect of time step resolution on the constraint quantity* $|r^2 G_{nn}|_{S^2}$ *for the weak field solution:* (a) $n_\infty = 256, \Delta z = 0.1, 0.05$*: the error is dominated by the radial discretization error for* $n < 192$ *and by the spectral truncation error for* $n > 192$*. Refining* Δz *produces no appreciable improvement in the solution.* (b) $n_\infty = 1024, \Delta z = 0.1, 0.05$*: the radial discretization error is small enough that the RK4 integration error can be observed. For* $n > 700$ *the error is dominated by the spectral truncation, resulting in the same tail as in* (a)*, while for* $n < 700$ *the constraint improves in places, consistent with a factor of* 16 *decrease in the error.*

of a 2-surface Σ reduces in the NQS gauge to

$$m_H(z,r) = \tfrac{1}{2} r \left(1 - \frac{1}{8\pi} \oint_{S^2} HJ\right). \tag{5.2}$$

$m_H(z,r)$ provides a quantity representing the "quasi-local" mass contained within the sphere (z,r), and has asymptotic limit equal to the Bondi mass

$$m_B(z) = \lim_{r\to\infty} m_H(r,z). \tag{5.3}$$

The Bondi mass is easily computed numerically, by $m_B(z) = m_H(n = n_\infty, z)$. Figure 5.8(a) shows the Hawking mass plotted against the radial coordinate, for times $z = 0, 1, \ldots, 55$. There are several features of interest in this plot: the limit Bondi mass (Figure 5.8(b)) decays in time, reflecting the Trautman–Bondi mass loss formula (5.4); the energy is radiated in bursts, reflecting near-linear behavior dominated by pure $l = 2$ modes; the Hawking and Bondi masses decay to the background black hole mass $\mathrm{M} = 1$ at late times, suggesting that in this example, almost all the gravitational radiation has been scattered to $\mathcal{I}^+$ and essentially none is absorbed by the black hole; and finally, the rapidly growing feature at about $n = 20$ at late times in Figure 4.4 does not affect the Hawking mass.

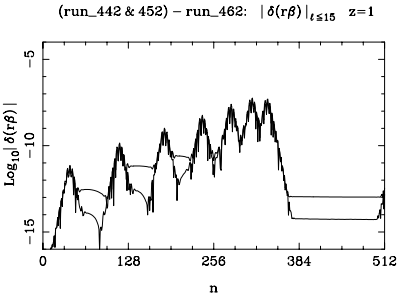


FIG. 5.7. *Convergence of β with decreasing time step: weak field solutions for $\Delta z = 0.1, 0.05$, compared against $\Delta z = 0.025$. Where the error is not dominated by the radial discretization error, the curves show a decrease in error which is consistent with fourth-order convergence.*

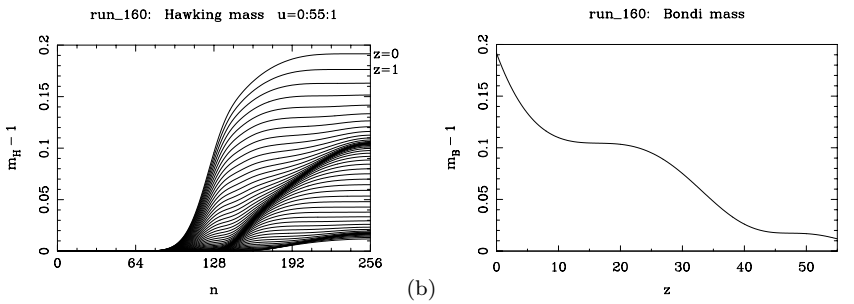


FIG. 5.8. *Mass functions:* (a) *Hawking mass for times $z = 0, 1, \ldots, 55$;* (b) *Bondi mass.*

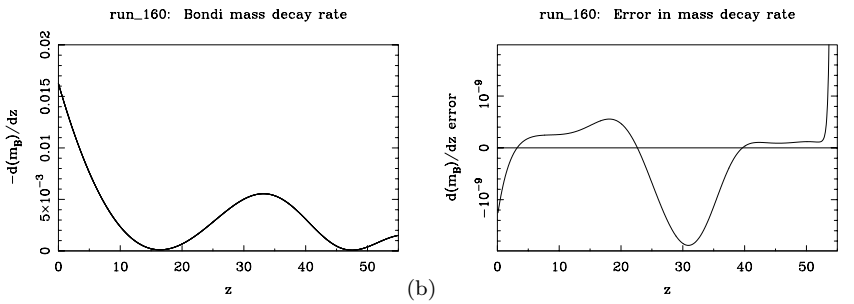


FIG. 5.9. *Trautman–Bondi mass loss formula as a test of numerical accuracy at $r = \infty$:* (a) *Bondi mass decay rate;* (b) *error in the mass decay formula, given by* LHS(5.4) − RHS(5.4).

The Trautman–Bondi mass loss formula [62, 63, 15, 29]

$$\frac{d}{dz} m_B(z) = -\frac{1}{16\pi} \lim_{r\to\infty} \oint_{S^2(z,r)} H|K|^2 \tag{5.4}$$

provides a test of the geometric consistency of the solution, particularly near null infinity. By comparing the numerical derivative dm_B/dz with the computed value of the RHS (evaluated at $n = n_\infty$), we may construct the error $\frac{d}{dz} m_B - \mathrm{RHS}(5.4)$. Figure 5.9(b) plots this error against time z, suggesting that the asymptotic ($r = \infty$) fields of `run_160` are accurate to about 0.00001%.

REFERENCES

[1] R. Balean, *The Null-Timelike Boundary Problem for the Linear Wave Equation*, Ph.D. thesis, University of New England, New South Wales, Australia, 1996.

[2] R. Balean, *The null-timelike boundary problem for the linear wave equation*, Comm. Partial Differential Equations, 22 (1997), pp. 1325–1360.

[3] R. Bartnik, *Quasi-spherical metrics and prescribed scalar curvature*, J. Differential Geom., 37 (1993), pp. 31–71.

[4] R. Bartnik, *Einstein equations in the null quasi-spherical gauge*, Classical Quantum Gravity, 14 (1997), pp. 2185–2194.

[5] R. Bartnik, *Einstein equations in the null quasi-spherical gauge: Progress report*, in the Proceedings of the First Australasian Conference on General Relativity and Gravitation, University of Adelaide, 1996, D. Wiltshire, ed., Institute for Theoretical Physics, University of Adelaide, South Australia, 1996, pp. 25–38.

[6] R. Bartnik, *Shear-free null quasi-spherical spacetimes*, J. Math. Phys., 38 (1997), pp. 5774–5791.

[7] R. Bartnik, *Interaction of gravitational waves with a black hole*, in the Proceedings of the 12th International Congress of Mathematical Physics (ICMP '97), University of Queensland, Brisbane, Australia, 1997, T. Bracken and D. D. Wit, eds., International Press, Boston, MA, 1999, pp. 3–14.

[8] R. Bartnik and A. H. Norton, *NQS data explorer website*, University of Canberra, 1997, http://relativity.ise.canberra.edu.au/.

[9] R. Bartnik and A. H. Norton, *Numerical solution of the Einstein equations*, in Computational Techniques and Applications: CTAC97, Proceedings of the Eighth Biennial Conference, University of Adelaide, 1997, J. Noye, M. Teubner, and A. Gill, eds., World Scientific, Singapore, 1998, pp. 91–98.

[10] R. Bartnik and A. H. Norton, *Geometric Implementation of Spherical Harmonics*, Tech. rep., School of Mathematics and Statistics, University of Canberra, Australia, 1999; Also available online at http://gular.canberra.edu.au/relativity.html.

[11] B. K. Berger and V. Moncrief, *Numerical evidence that the singularity in polarized $U(1)$ symmetric cosmologies on $T^3 \times R$ is velocity dominated*, Phys. Rev. D, 57 (1998), pp. 7235–7240.

[12] N. Bishop, R. Gómez, L. Lehner, M. Maharaj, and J. Winicour, *High-powered gravitational news*, Phys. Rev. D, 56 (1997), p. 6298.

[13] N. T. Bishop, *Some aspects of the characteristic initial value problem in numerical relativity*, in Approaches to Numerical Relativity, R. d'Inverno, ed., Cambridge University Press, Cambridge, UK, 1992, pp. 20–33.

[14] S. Bonazzola, E. Gourgoulhon, and J.-A. Marck, *Spectral methods in general relativistic astrophysics*, J. Comput. Appl. Math., 109 (1999), pp. 433–473.

[15] H. Bondi, *Gravitational waves in general relativity*, Nature, 186 (1960), p. 535.

[16] J. P. Boyd, *The choice of spectral functions on a sphere for boundary and eigenvalue problems: A comparison of Chebyshev, Fourier, and associated Legendre expansions*, Monthly Weather Rev., 106 (1978), pp. 1184–1191.

[17] J. P. Boyd, *Chebyshev and Fourier Spectral Methods*, Lecture Notes in Engrg. 49, Springer, Berlin, 1989.

[18] G. L. Browning, J. J. Hack, and P. N. Swarztrauber, *A comparison of three numerical methods for solving differential equations on the sphere*, Monthly Weather Rev., 117 (1989), pp. 1058–1075.

[19] C. Canuto, M. Y. Hussaini, A. Quarteroni, and T. A. Zang, *Spectral Methods in Fluid Dynamics*, Series in Computational Physics, Springer, Berlin, New York, 1988.

[20] S. Chandrasekhar, *The Mathematical Theory of Black Holes*, Oxford University Press, London, UK, 1984.

[21] R. D'Inverno, *Introducing Einstein's Relativity*, Oxford University Press, London, UK, 1992.

[22] M. Eastwood and P. Tod, *Edth – a differential operator on the sphere*, Math. Proc. Cambridge Philos. Soc., 92 (1982), pp. 317–330.

[23] J. Futterman, F. Handler, and R. Matzner, *Scattering from Black Holes*, Cambridge University Press, Cambridge, UK, 1988.

[24] J. N. Goldberg, A. J. Macfarlane, E. T. Newman, F. Rohrlich, and E. C. G. Sudarshan, *Spin-s spherical harmonics and* $\eth$, J. Math. Phys., 8 (1967), pp. 2155–2161.

[25] R. Gómez, P. Papadopoulos, and J. Winicour, *Null cone evolution of axi-symmetric vacuum space-times*, J. Math. Phys., 35 (1994), pp. 4184–4203.

[26] R. Gómez and J. Winicour, *Asymptotics of gravitational collapse of scalar waves*, J. Math. Phys., 33 (1992), p. 1445.
[27] R. Gómez, J. Winicour, and R. Isaacson, *Evolution of scalar fields from characteristic data*, J. Comp. Phys., 98 (1992), p. 11.
[28] R. Gómez, J. Winicour, and P. Papadopoulos, *The eth formalism in numerical relativity*, Classical Quantum Gravity, 14 (1997), pp. 977–990.
[29] H. Bondi, M. G. J. van der Burg, and A. W. K. Metzner, *Gravitational waves in general relativity* VII, Proc. Roy. Soc. Lond. A, 269 (1962), pp. 21–51.
[30] R. W. Hamming, *Digital Filters*, Prentice-Hall, Upper Saddle River, NJ, 1977.
[31] S. W. Hawking and G. R. Ellis, *The large-scale structure of spacetime*, Cambridge University Press, Cambridge, UK, 1973.
[32] A. C. Hearn, *REDUCE Users Manual, Version* 3.6, RAND Publication CP78, Santa Monica, CA, 1996.
[33] R. Jakob-Chien and B. K. Alpert, *A fast spherical filter with uniform resolution*, J. Comput. Phys., 136 (1997), pp. 580–584.
[34] P. E. Merilees, *The pseudo-spectral approximation applied to the shallow water equations on a sphere*, Atmosphere, 11 (1973), pp. 13–20.
[35] V. Moncrief, *Gravitational perturbations of spherically symmetric systems*, Ann. Physics, 88 (1974), pp. 323–342.
[36] E. Newman and R. Penrose, *An approach to gravitational radiation by a method of spin coefficients*, J. Math. Phys., 3 (1962), pp. 566–578.
[37] E. T. Newman and T. Unti, *A class of null flat-space coordinate systems*, J. Math. Phys., 4 (1963), pp. 1467–1469.
[38] A. H. Norton, *Finite Difference Operators for PDEs, Based on Sampling Kernels for Spline Quasi-Interpolation*, Tech. rep., School of Mathematics, University of New South Wales, Australia, 1992.
[39] A. H. Norton, *Spectral collocation methods for solution of Einstein's equations in null quasi-spherical coordinates*, in the Proceedings of the First Australasian Conference on General Relativity, University of Adelaide, 1996, D. L. Wiltshire, ed., Institute for Theoretical Physics, University of Adelaide, South Australia, 1996, pp. 39–47.
[40] A. H. Norton, *The use of convolution splines in the NQS Einstein evolution code*, in Computational Techniques and Applications: CTAC97, Proceedings of the Eighth Biennial Conference, University of Adelaide, 1997, J. Noye, M. Teubner, and A. Gill, eds., World Scientific, Singapore, 1998, pp. 473–480.
[41] S. A. Orszag, *On the elimination of aliasing in finite difference schemes by filtering high wave number components*, J. Atmosph. Sci., 28 (1971), p. 1074.
[42] S. A. Orszag, *Fourier series on spheres*, Monthly Weather Rev., 102 (1974), pp. 56–75.
[43] R. Penrose, *The geometry of impulsive gravitational waves*, in General Relativity: Papers in Honour of J. L. Synge, L. O'Raifaertaigh, ed., Clarendon Press, Oxford, 1972.
[44] R. Penrose and W. Rindler, *Spinors and Space-Time*, Vol. I, II, Cambridge University Press, UK, 1984, 1986.
[45] D. A. Prager and A. W. C. Lun, *Computational methods in the physical interpretation of Robinson-Trautman spacetimes*, in the Proceedings of the First Australasian Conference on General Relativity and Gravitation, University of Adelaide, 1996, D. L. Wiltshire, ed., Institute for Theoretical Physics, University of Adelaide, South Australia, 1996, pp. 48–59.
[46] P. J. Prince and J. R. Dormand, *High order embedded Runge-Kutta formulae*, J. Comput. Appl. Math., 7 (1981), pp. 67–75.
[47] T. Regge and J. A. Wheeler, *Stability of a Schwarzschild singularity*, Phys. Rev., 108 (1957), pp. 1063–1069.
[48] A. D. Rendall, *Reduction of the characteristic initial value problem to the Cauchy problem and its application to the Einstein equations*, Proc. Roy. Soc. Lond. A, 427 (1990), pp. 221–239.
[49] I. Robinson and A. Trautman, *Spherical gravitational waves in general relativity*, Proc. Roy. Soc. Lond. A, 270 (1962), pp. 103–126.
[50] R. K. Sachs, *Gravitational waves in general relativity,* VIII. *Waves in asymptotically flat space-time*, Proc. Roy. Soc. Lond. A, 270 (1962), pp. 103–126.
[51] R. K. Sachs, *Gravitational waves in general relativity* VIII. *Waves in asymptotically flat space-time*, Proc. Roy. Soc. Lond. A, 270 (1962), pp. 103–126.
[52] L. J. Schoenberg, *Cardinal Spline Interpolation*, CBMS–NSF Regional Conference Series in Applied Mathematics 12, SIAM, Philadelphia, PA, 1973.
[53] D. Singleton, *Robinson-Trautman Solution of Einstein's Equations*, Ph.D. thesis, Monash University, Melbourne, Australia, 1990.

[54] L. Smarr and J. J. W. York, *Kinematical conditions in the construction of spacetime*, Phys. Rev. D, 17 (1978), pp. 2529–2551.
[55] G. Starius, *Composite mesh difference methods for elliptic and boundary value problems*, Numer. Math., 28 (1977), pp. 243–248.
[56] K. Stellmacher, *Ausbretungsgesetze für charakteristische Singularitäten der Gravitationsgleichungen*, Math. Annalen, 145 (1938), pp. 740–783.
[57] F. Stenger, *Numerical Methods Based on Sinc and Analytic Functions*, Springer Ser. Comput. Math. 20, Springer, New York, 1993.
[58] P. N. Swarztrauber, *On the spectral approximation of discrete scalar and vector functions on the sphere*, SIAM J. Numer. Anal., 16 (1979), pp. 934–949.
[59] P. N. Swarztrauber, *The approximation of vector functions and their derivatives on the sphere*, SIAM J. Numer. Anal., 18 (1981), pp. 191–210.
[60] P. N. Swarztrauber, *Software for the spectral analysis of scalar and vector functions on the sphere*, in Large Scale Scientific Computation, Academic Press, Orlando, FL, 1984.
[61] P. N. Swarztrauber, *Spectral transform methods for solving the shallow-water equations on the sphere*, Monthly Weather Rev., 124 (1996), pp. 730–744.
[62] A. Trautman, *King's College lecture notes on general relativity*, manuscript, 1958.
[63] A. Trautman, *Radiation and boundary conditions in the theory of gravitation*, Bull. Acad. Polon. Sci. Sér. Sci. Math., 6 (1958), pp. 407–412.
[64] A. Trautman, *Conservation laws in general relativity*, in Gravitation: An Introduction to Current Research, L. Witten, ed., Wiley, New York, 1962, pp. 169–198.
[65] F. J. Zerilli, *Effective potential for even parity regge-wheeler gravitational perturbation equations*, Phys. Rev. Lett., 24 (1970), pp. 737–738.
[66] H. M. zum Hagen, *Characteristic initial value problem for hyperbolic systems of second order differential equations*, Ann. Inst. H. Poincaré, 53 (1990), pp. 159–216.

SIAM J. SCI. COMPUT.
Vol. 22, No. 3, pp. 951–985
© 2000 Society for Industrial and Applied Mathematics

COMPUTING PERIODIC ORBITS AND THEIR BIFURCATIONS WITH AUTOMATIC DIFFERENTIATION*

JOHN GUCKENHEIMER† AND BRIAN MELOON†

Abstract. This paper formulates several algorithms for the direct computation of periodic orbits as solutions of boundary value problems. The algorithms emphasize the use of coarse meshes and high orders of accuracy. Convergence theorems are given in the limit of increasing order with a fixed mesh. The algorithms are implemented with the use of MATLAB and ADOL-C, a software package for automatic differentiation. Automatic differentiation enables accurate computation of high-order derivatives of functions without the truncation errors inherent in finite difference calculations. We embed the algorithms in a continuation framework and extend them to compute saddle-node bifurcations of periodic orbits directly. We present data from numerical studies of four test problems, making some comparisons with other methods for computing periodic orbits. These results demonstrate that high-order methods based upon automatic differentiation are capable of high precision with small meshes.

Key words. periodic orbit, automatic differentiation, saddle-node bifurcation

AMS subject classifications. 65L10, 37G15, 37M20, 65P99

PII. S1064827599359278

1. Introduction.

Periodic orbits are fundamental objects in dynamical systems. They are frequently important in applications of the dynamical systems theory. Computational tools that locate periodic orbits and their bifurcations are an essential ingredient for the numerical investigation of vector fields. Stable periodic orbits are often found in these investigations as the limit sets of trajectories computed with numerical integration. However, there are many circumstances in which we want to compute periodic orbits that are not accessible in this manner. Direct methods based upon the solution of boundary value problems are called for in these circumstances. Formulation and implementation of effective boundary value solvers for periodic orbits is a difficult task. There are a few tools that have been developed for this purpose, notably the package AUTO [10] based upon a collocation method. This paper examines alternate approaches for computing periodic orbits of dynamical systems.

Our goal is to explore the limits of accuracy that are attainable for these calculations within the pragmatic context of IEEE floating point arithmetic. A previous paper [17] described a global boundary value solver of very high-order accuracy that relies upon automatic differentiation. We extend that work here, introducing multiple shooting algorithms that use automatic differentiation and augmenting these to perform direct computation of saddle-node bifurcations. We compare our algorithms with other numerical methods for finding periodic orbits, examining robustness and accuracy.

Robust algorithms facilitate the automated analysis of dynamical systems. Accuracy is necessary to achieve robustness for two reasons. First, many dynamical systems have fine scale structure that makes them difficult to compute. This is especially true of dynamical systems with multiple time scales and systems that are

*Received by the editors July 23, 1999; accepted for publication (in revised form) January 26, 2000; published electronically August 31, 2000. This research was partially supported by the Air Force Office of Scientific Research, the Department of Energy, and the National Science Foundation.
http://www.siam.org/journals/sisc/22-3/35927.html

†Department of Mathematics, Cornell University, Ithaca, NY 14853 (gucken@cam.cornell.edu, bam11@math.cornell.edu).

chaotic. Second, defining equations for bifurcations of periodic orbits involve derivatives of the vector field along the orbit. Newton's method applied to the bifurcation equations requires first derivatives of the defining equations and second derivatives of the periodic orbit equations. Calculation of second derivatives with finite differences layered on imprecise calculation of periodic orbits is hardly a recipe for robustness. Improved algorithms of greater accuracy for computing periodic orbits and the use of automatic differentiation to compute defining equations enhance existing technology for analysis of dynamical systems.

Our algorithms approach the limits of precision attainable with double precision IEEE-754 arithmetic for computing periodic orbits. The algorithms utilize three strategies that contribute to their accuracy:

- Automatic differentiation is employed to calculate high-order derivatives of a vector field and its trajectories.
- The methods emphasize compact parametrizations of function spaces that include high-order approximations to the periodic orbits.
- A posteriori error estimates of the accuracy of numerically computed periodic orbits are used for mesh adaptation.

The methods have additional attractive geometric features from both theoretical and geometric perspectives. They utilize directly the geometric objects that are prominent in the theory. These objects can be readily examined and manipulated, and they can be used adaptively to enable the algorithms to respond to changes in the geometry of a periodic orbit during continuation. The algorithms give dense output of uniform order. More specifically, no further interpolation is required to approximate periodic orbits at all points to the same order of accuracy as at mesh points. Constraints are readily imposed upon mesh points, enabling the accurate computation of periodic orbits of piecewise analytic vector fields.

The plan of the paper is as follows. Section 2 describes four algorithms for computing periodic orbits, each of which uses Taylor series expansions at the mesh points to achieve high-order. Section 3 describes a continuation framework for the algorithms and formulates defining equations for local bifurcations of periodic orbits. Section 4 describes how we implemented the algorithms and discusses our strategies for mesh adaptation. Section 5 presents data from numerical studies of four systems. These examples were chosen to test the accuracy and robustness of the methods on challenging problems. The paper ends with a brief set of conclusions.

2. Periodic orbit algorithms. The equations describing periodic orbits are boundary value problems for a system of ordinary differential equations [2]. Only a small number of algorithms have been widely used for the computation of periodic orbits. These fall into three *classes* [2]:

1. Numerical integration of initial values in the domain of attraction of a stable periodic orbit γ converge to γ [20, 21]. (Despite the robustness of numerical integration algorithms, we display examples where they are unable to compute stable periodic orbits.)
2. Shooting methods apply root finding algorithms to approximate flow maps computed with numerical integration [8, 25, 7].
3. Global methods project the differential equations onto spaces of discretized closed curves and solve these projected equations with root finding algorithms [2, 10].

We discuss both shooting methods and global methods of high order. Our methods fall within general classes that have been studied previously, but we explore the limit

of increasing order in the methods as contrasted with the usual limit of increasingly fine meshes. This has some resemblance to p refinement in "h-p" methods studied extensively by Babuška and Suri [3], though we appear to go much farther in our investigation of convergence on fixed meshes. Although Taylor series integration has been previously studied [4] and Taylor series methods have been described for two-point boundary value problems [27], its use in periodic orbit algorithms appears to be new.

This section presents four different algorithms that we have implemented using automatic differentiation to compute Taylor series of trajectories. We give convergence proofs, showing that the methods converge to hyperbolic periodic orbits on fixed meshes with increasing order. We use a common terminology for all the algorithms that is now described.

A *periodic orbit* of a vector field

$$\dot{x} = f(x),\ f : R^n \to R^n \tag{2.1}$$

with period T is a trajectory with $x(T) = x(0)$. T is called the *period* of the orbit. As a point set in the phase space, a periodic orbit may be isolated, but time translation gives a one parameter family of different parametrized curves corresponding to each periodic orbit. Local analysis of a periodic orbit is based upon the formulation of linear variational equations along the periodic orbit:

$$\dot{\xi} = Df_{x(t)}\xi.$$

Taking ξ as a matrix variable, the time T solution of the variational equation with identity initial condition at $t = 0$ is called the *monodromy matrix* of the periodic orbit. The eigenvalues of the monodromy matrix are independent of the coordinate system and the parametrization of the periodic orbit. There is always an eigenvalue 1 with eigenvector tangent to the periodic orbit. A periodic orbit is *elementary* if 1 is a simple eigenvalue of the monodromy matrix. Elementary periodic orbits are isolated in the sense described above, and they vary smoothly with parameters. We shall denote the flow of the system of equations by $\phi_t(x)$: ϕ_t is the map that advances points t time units along their trajectories.

Periodic orbits are also studied by introduction of *return maps*. A cross-section Σ to f is a codimension 1 submanifold of R^n with the property that f is never tangent to Σ. The return map $\theta : \Sigma \to \Sigma$ is defined by $\theta(y) = y(s)$ with $y(s)$ the first point of the trajectory with initial condition y that lies in Σ. Periodic orbits have cross-sections whose return maps have fixed points at the intersection of the cross-section with the periodic orbit. The Jacobian derivative of the return map at such a fixed point is closely related to the monodromy matrix of the periodic orbit. In particular, the Jacobian of the return map can be obtained as a quotient of the monodromy matrix by projecting out the flow direction. The fixed point of a return map for an elementary periodic orbit is an isolated, regular fixed point of the return map. This observation can be used as the basis for simple shooting algorithms for computing the periodic orbits.

We assume throughout this paper that $f(x)$ is analytic. Periodic orbits of analytic vector fields are analytic, but spaces of functions that are discontinuous and/or piecewise smooth underlie algorithms that compute approximate periodic orbits. Since computational representations of periodic orbits only use finite sets of data, we work with finite-dimensional function spaces and sequences of such spaces, indexed by a degree. The spaces are parametrized by discrete closed curves, defined below. Each

discrete closed curve is used to parametrize a sequence of function spaces of fixed dimension that provide increasingly good approximations to periodic orbits. Different algorithms use different function spaces.

The data structure that we use to represent an approximate periodic orbit is a discrete closed curve. A *discrete curve* $\delta = [(t_0, x_0), (t_1, x_1), \ldots, (t_N, x_N)]$ is defined as a set of times $t_0 < t_1 < \cdots < t_N$ and points $x_0, x_1, \ldots, x_N \in R^n$ associated to these times. If $x_0 = x_N$, then we say that the discrete curve is *closed* and has period $t_N - t_0$. The $(n+1)(N+1)$-dimensional space of discrete curves with $N+1$ points will be denoted $D_{(n,N)}$. The discrete closed curves comprise a linear subspace $D^c_{(n,N)}$ of $D_{(n,N)}$ of dimension $N(n+1)+1$ and codimension n. Each periodic orbit is represented by an $N+1$ dimensional family of discrete closed curves, coming from the selection of different points on the periodic orbit (N degrees of freedom) and time translation. In formulating boundary value methods, we seek systems of defining equations that are square and regular. This can be done by either restricting the domain of the defining equations to a subspace of discrete closed curves which contains a (locally) unique representative of each periodic orbit, or by adjoining additional equations that select a unique discrete closed curve representing the periodic orbit we seek. We have experimented with both strategies. We always remove the degree of freedom due to time translation by restriction to the subspace $t_0 = 0$. Restriction to a subspace that eliminates the degrees of freedom due to moving points on the periodic orbit results in smaller systems of defining equations but can make the computation of the defining equations more difficult. For example, the strategy used in [17] for selecting a subspace of discrete closed curves is to select N cross-sections Σ_i to the periodic orbit and restrict x_i to lie in Σ_i. This requires that we define coordinate systems for the Σ_i and incorporate transformations between these coordinates and Euclidean coordinates in our algorithms. On the other hand, adjoining additional equations only increases the number of equations from nN to $(n+1)N$, and as the system size increases, the extra work in solving the augmented system becomes relatively small. Abstractly, the difference between the two strategies can be expressed as the difference between solving a triangular system of equations

$$\begin{aligned} g(x, y) &= 0, \\ h(y) &= 0, \end{aligned}$$

by elimination, first solving $h(y) = 0$ for y and then substituting this value into $g(x, y) = 0$, as contrasted with solving the two-dimensional system simultaneously for x and y. We have found in the canard example described below that augmenting the system of defining equations produced much better results than the elimination method.

There are two different choices of subspaces of $D^c_{(n,N)}$ that we use to fix the location of points along a periodic orbit. The first is to use cross-sections to the vector field as described above. Cross-sections are hyperplanes of R^n with the property that f is never tangent to Σ_i in the region of interest, defined in the algorithms as hyperplanes perpendicular to the vector field at a point. The subspace $D^s_{(n,N)}$ of $D^c_{(n,N)}$ with $t_0 = 0$ and $x_i \in \Sigma_i$ has an isolated point of intersection with an elementary periodic orbit. The methods most often employed in previous work on computing periodic orbits attempt as much as possible to fix the times $t_0 < t_1 < \cdots < t_N$ of the points $x_0, x_1, \ldots, x_{N-1}, x_N = x_0$ in the discrete closed curve. Fixing all of the times does not work because the period $t_N - t_0$ of the orbit is not known and must be obtained as part of the solution of the defining equations. There is one degree of

freedom that must be retained in allowing the times to vary. This is done typically by either fixing $t_0 < t_1 < \cdots < t_{N-1}$ and allowing t_N to vary, or by fixing the ratios $(t_{i+1} - t_i)/(t_i - t_{i-1})$ for $0 < i < N$. One additional constraint, called a phase condition, must be imposed on the points x_i that eliminates the degree of freedom coming from moving all of the points along the flow simultaneously by time τ and replacing t_i by $t_i - \tau$. Two choices of phase condition that have been employed are to restrict x_0 to lie on a cross-section, and to fix the value of a scalar integral on the periodic orbit.

The next step in defining equations for approximate periodic orbits is to map the spaces of discrete curves to function spaces. Let $\mathcal{C}$ be the space of piecewise smooth curves $g : R \to R^n$ defined by the following properties: $g \in \mathcal{C}$ if there are $u_0 < u_1 < \cdots < u_l$ such that g is analytic on the intervals (u_i, u_{i+1}) and has a C^∞ extension to the closed interval $[u_i, u_{i+1}]$. We consider nonlinear maps $E : D^c_{n,N} \to \mathcal{C}$ with the property that the image functions are analytic on the intervals (t_i, t_{i+1}), $0 \le i < N$. Note that E is allowed to depend upon the vector field f.

A map $E : D^c_{n,N} \to \mathcal{C}$ pulls back the periodic orbit equations from $\mathcal{C}$ to $D^c_{(n,N)}$. These equations give a horrendously overdetermined system on $D^c_{n,N}$, an infinite set of nonlinear equations on a finite-dimensional space. Therefore, a finite set of nN equations is selected from the periodic orbit equations. The choice of the map E together with the definition of a regular set of defining equations, perhaps on a subspace $D^s_{(n,N)}$ of $D^c_{(n,N)}$, determines the boundary value solver. We give here a list of four different solvers, specifying the approximate periodic orbit equations for each.

- *Forward multiple shooting:* Let $\delta = [(t_0, x_0), \ldots, (t_N, x_N)]$ be a discrete closed curve. The map E assigns to each time interval (t_i, t_{i+1}) of δ the degree d Taylor polynomial p_i of the trajectory with $x(t_i) = x_i$. Since polynomials are analytic on the entire line, this associates a piecewise analytic function in $\mathcal{C}$ to δ. The approximate periodic orbit equations are now defined by

  $$P(\delta) = (p_0(t_1) - x_1, \ldots, p_{N-1}(t_N) - x_N).$$

  P vanishes on discrete closed curves δ for which $E(\delta)$ is continuous. It is evident that P is smooth as a map from $R^{(n+1)N+1}$ to R^{nN} since the degree d Taylor series of a trajectory of an analytic vector field depends smoothly on the initial point. The approximate periodic orbit equations P restricted to $D^s_{(n,N)}$ are a set of nN equations in nN variables. Here $D^s_{(n,N)}$ can be the subspace of $D^c_{(n,N)}$ defined via cross-sections and $t_0 = 0$, or by fixing N times and a phase condition. One can also fix $t_0 = 0$ and augment the periodic orbit equations with a set of N additional equations. In the theorem proved below we describe several ways of choosing N augmenting linear equations so that the resulting set of $(n + 1)N$ equations in the variables $(x_0, \ldots, x_{N-1}; t_1, \ldots, t_N)$ are regular.
- *Symmetric multiple shooting:* Symmetric multiple shooting is a small modification of the multiple shooting method described above that makes the method time reversible. Let $\delta = [(t_0, x_0), \ldots, (t_N, x_N)]$ be a discrete closed curve. We set $u_0 = t_0 - (t_N - t_{N-1})/2$ and $u_i = (t_i + t_{i-1})/2$, $0 < i \le N$. The map E then assigns to each time interval (u_i, u_{i+1}) the degree d Taylor polynomial p_i of the trajectory with $x(t_i) = x_i$. We define the map P by

  $$P(\delta) = (p_0(u_0) - p_{N-1}(u_N), p_1(u_1) - p_0(u_1), \ldots, p_{N-1}(u_{N-1}) - p_{N-2}(u_{N-1})).$$

  Finally, P is restricted to $D^s_{n,N}$ to obtain a set of nN equations in nN variables, or the set of defining equations is augmented with N additional

equations. In implementations of these algorithms, we replace the variables t_i by the equivalent set of variables $t_i - t_{i-1}$ representing the time increment of each mesh interval.

- *Hermite interpolation:* Let $\delta = [(t_0, x_0), \ldots, (t_N, x_N)]$ be a discrete closed curve. The map E assigns to each time interval (t_i, t_{i+1}) of δ the polynomial p_i of degree $2d+1$ that has the same Taylor series of degree d as the trajectories with $x(t_i) = x_i$ and $x(t_{i+1}) = x_{i+1}$. The image of E lies in the set of piecewise analytic functions that are continuous and d times differentiable. We define the function P by

$$P(\delta) = (e_0, \ldots, e_{N-1})$$

with $e_i = f(p_i((t_i + t_{i+1})/2)) - \dot{p}_i((t_i + t_{i+1})/2)$. P is restricted to $D^s_{n,N}$ to obtain nN equations in nN variables.

- *Smooth weighting:* Let $\delta = [(t_0, x_0), \ldots, (t_N, x_N)]$ be a discrete closed curve, and let $\beta : [0,1] \to [0,1]$ be a monotonic C^∞ function that is flat at its endpoints ($\beta^{(k)}(u) = 0$ for all $k > 0$) and has $\beta(0) = 0$, $\beta(1) = 1$. We assume that β is analytic in $(0,1)$ with positive derivative. Let p_i be the degree d Taylor polynomial of the trajectory with $x(t_i) = x_i$. To each interval (t_i, t_{i+1}), we assign the C^∞ function

$$g_i(t) = (1 - \beta((t - t_i)/(t_{i+1} - t_i)))p_i(t) + \beta((t - t_i)/(t_{i+1} - t_i))p_{i+1}(t).$$

This defines a C^∞ closed curve that interpolates the degree d Taylor series expansions at the mesh points. We define the function P by

$$P(\delta) = (e_0, \ldots, e_{N-1})$$

with $e_i = f(g_i((t_i + t_{i+1})/2)) - \dot{g}_i((t_i + t_{i+1})/2)$. P is restricted to $D^s_{n,N}$ to obtain nN equations in nN variables.

Each of these four methods yields a system of equations defined by a map P that depends on the same number of equations as variables. Note that there is an inherent difference between the first two algorithms and the last two. The first two are multiple shooting algorithms that use spaces of discontinuous functions and their maps P do not depend directly upon evaluating the vector field. The approximation to the periodic orbit is implicit in the choice of the function space and depends upon the accuracy of the trajectory segments. The last two solvers are global methods defined on smooth curves (the degree of smoothness of the Hermite interpolations depends on d) and their equations P evaluate both the vector field at selected points and the tangent vectors to curves in the function space.

We want convergence proofs and error estimates for these algorithms in terms of the vector field, characteristics of the periodic orbit, and algorithmic parameters. There are two types of limits that are of interest: increasing degree d of Taylor expansions and increasing mesh fineness. For fixed degree of the Taylor polynomials and increasingly fine meshes, the algorithms are variants of standard shooting and collocation algorithms. Therefore, we focus our mathematical discussion on the limit of increasing degree with a fixed mesh. This provides mathematical foundations for algorithms that seek to compute periodic orbits using polynomials of high degree with coarse meshes.

The theorems stated below depend upon the geometry of the periodic orbits that are being approximated. If periodic orbits are not elementary, there is little hope

that the projected equations P will be regular. Thus, our convergence results focus on the case of elementary periodic orbits. We assume that all discrete closed curves $\delta = [(t_0, x_0), (t_1, x_1), \ldots, (t_N, x_N)]$ have the property that the radius of convergence of the trajectories through the points x_i are larger than $t_i - t_{i-1}$ and $t_{i+1} - t_i$. Within a compact subset of R^n, there is a number $D > 0$ such that this property will be satisfied if $|t_i - t_{i-1}| < D$ for each i. In particular, this assumption implies that the Taylor polynomials of the trajectories converge uniformly to the trajectories with increasing degree throughout each mesh interval. The assumption that meshes are sufficiently fine that each mesh interval is contained in the radius of convergence of the trajectories at its endpoints will be maintained throughout this paper.

THEOREM 2.1. *Let γ be a periodic orbit of the vector field $\dot{x} = f(x)$ such that 1 is a simple eigenvalue of its monodromy matrix. Let $D \subset R^{(n+1)(N+1)}$ be the $(n+1)(N)$-dimensional subspace with coordinates $(x_0, \ldots, x_N; t_0, \ldots, t_N)$ defined by $x_0 = x_N$ and $t_0 = 0$. For any $\epsilon > 0$, there are*

- *a neighborhood U of γ,*
- *$\tau > 0$,*
- *integers d, N,*

so that the system of equations

$$|x_{i+1} - p_i(t_{i+1})| = 0,\ i = 0, \ldots, N-1$$

has a smooth N parameter family of solutions in D with

- *p_i the Taylor polynomial of degree d for the trajectory taking the value x_i at t_i,*
- *$0 < s_i = t_{i+1} - t_i < \tau$,*
- *$|x_{i+1} - \phi_{s_i}(x_i)| < \epsilon$.*

The Jacobian of the system of equations has maximal rank nN.

Proof. Select a neighborhood U of γ and τ so that the Taylor series of trajectories with initial point in U have radius of convergence larger than τ. Next, select a set of N points $y_0, \ldots, y_{N-1}, y_N = y_0$ on γ so that $y_{i+1} = \phi_{u_i}(y_i)$ with $u_i < \tau$. Set

$$r_i = \sum_{j=0}^{i} u_j.$$

Consider the map $F_f^d : D \to R^{nN}$ defined by

$$\begin{gathered} p_0(r_0) - x_1 \\ p_1(r_1) - x_2 \\ \vdots \\ p_{N-2}(r_{N-2}) - x_{N-1} \\ p_{N-1}(r_{N-1}) - x_0 \end{gathered}$$

as an approximation of the map $F_f^\infty : D \to R^{nN}$

$$\begin{gathered} \phi_{s_0}(x_0) - x_1 \\ \phi_{s_1}(x_1) - x_2 \\ \vdots \\ \phi_{s_{N-2}}(x_{N-2}) - x_{N-1} \\ \phi_{s_{N-1}}(x_{N-1}) - x_0. \end{gathered}$$

The map F_f^d depends upon the degree d of the Taylor polynomials and converges to the map F_f^∞ as $d \to \infty$. (Since the maps F_f^∞ and F_f^d are defined on finite-dimensional spaces of discrete closed curves, convergence is independent of norms for these spaces.) The map F_f^∞ has an N-dimensional manifold S of zeros passing through the discrete closed curve $[(r_0, y_0), \ldots, (r_N, y_N)] \in D$. S is obtained by varying the points y_i along γ and making suitable changes in the u_i. If this zero of F_f^∞ is regular, then, for sufficiently large d, F_f^d will have a regular zero that converges to the zero of F_f^∞. Thus, the crux of the proof lies in demonstrating that the rank of DF_f^∞ is nN.

We relate the Jacobian of F_f^∞ to the monodromy matrix of γ. In D, we use coordinates $(x_0, \ldots, x_{N-1}; t_1, \ldots, t_N)$. The Jacobian DF_f^∞ has a block structure

$$DF_f^\infty = \begin{pmatrix} D_1 & -I & & & & v_1 & & & & & \\ & D_2 & -I & & & -v_2 & v_2 & & & & \\ & \cdots & \cdots & & & & \cdots & \cdots & & & \\ & & \cdots & \cdots & & & & \cdots & \cdots & & \\ & & & D_{N-1} & -I & & & & -v_{N-1} & v_{N-1} & \\ -I & & & & D_N & & & & & -v_N & v_N \end{pmatrix}$$

with $D_i = D_x(\phi)(x_{i-1}, t_i - t_{i-1})$ and $v_i = f(x_i)$. Note that $D_i(v_{i-1}) = v_i$. The blocks D_i are nonsingular since the flow maps ϕ_t are diffeomorphisms. To determine the rank of the matrix DF_f^∞, we compute its LU factorization in the form

$$L = \begin{pmatrix} I & & & & \\ & \ddots & & & \\ & & & I & \\ L_1 & L_2 & \ldots & L_{n-1} & I \end{pmatrix},$$

$$U = \begin{pmatrix} D_1 & -I & & & & v_1 & & & & & \\ & D_2 & -I & & & -v_2 & v_2 & & & & \\ & \cdots & \cdots & & & & \cdots & \cdots & & & \\ & & \cdots & \cdots & & & & \cdots & \cdots & & \\ & & & D_{N-1} & -I & & & & -v_{N-1} & v_{N-1} & \\ & & & & \tilde{D}_N & w_1 & w_2 & \ldots & \ldots & w_{N-1} & w_N \end{pmatrix}.$$

This yields a system of equations for the L_i, $\tilde{D}_N$, and w_i:

$$\begin{aligned} L_1 D_1 &= -I, \\ L_i D_i - L_{i-1} I &= 0, \\ \tilde{D}_N - L_{N-1} I &= D_N, \\ L_{i-1} v_{i-1} - L_i v_i + w_{i-1} &= 0, \\ L_{N-1} v_{N-1} + w_{N-1} &= -v_N, \\ w_N &= v_N, \end{aligned}$$

where $1 < i < N$. The solutions of this system are

$$\begin{aligned} L_i &= -D_1^{-1} \cdots D_i^{-1}, \\ \tilde{D}_N &= D_N - D_1^{-1} \cdots D_{N-1}^{-1}, \\ w_i &= 0, \\ w_N &= v_N, \end{aligned}$$

with $0 < i < N$. The factor L has rank nN since it is lower triangular with ones along the diagonal. The factor U has rank nN if and only if the rank of $\tilde{D}_N$ is $n-1$ and v_N does not lie in the image of $\tilde{D}_N$. By definition, γ is an elementary periodic orbit if and only if 1 is a simple eigenvalue of the monodromy matrix $D_N \cdots D_1$. Thus, $\tilde{D}_N$ has rank $n-1$ with kernel v_{N-1} if γ is elementary. If $v_N = v_0$ is in the image of $\tilde{D}_N$, then $v_{N-1} = D_{N-1} \cdots D_1(v_0)$ is in the image of $D_{N-1} \cdots D_1 \tilde{D}_N = D_{N-1} \cdots D_1 D_N - I$. But $D_{N-1} \cdots D_1 D_N$ is the monodromy matrix with initial point y_{N-1} and v_{N-1} is the eigenvector for the simple eigenvector 1, so v_{N-1} is not in the image of $D_{N-1} \cdots D_1 D_N - I$. We conclude that if γ is elementary, then v_N does not lie in the image of $\tilde{D}_N$ and U has rank nN. Thus the system of equations F_f^∞ has maximal rank at a zero formed from N points on an elementary periodic orbit γ. Perturbations of this system also have maximal rank and the manifold of solutions varies smoothly with perturbation. Since the Taylor polynomials of the flow map converge to the flow map on time intervals of length smaller than τ, the conclusions of the theorem will be satisfied for the map F_f^d when the degree d of the Taylor polynomials is sufficiently large. This proves the theorem.

Similar arguments to the one used above prove the convergence of other multiple shooting algorithms, notably the symmetric multiple shooting algorithm with increasing degree. The zeros of the map F_s^∞

$$\begin{array}{c} \phi_{s_1}(x_0) - \phi_{-s_1}(x_1) \\ \phi_{s_2}(x_1) - \phi_{-s_2}(x_2) \\ \vdots \\ \phi_{s_{N-1}}(x_{N-2}) - \phi_{-s_{N-1}}(x_{N-1}) \\ \phi_{s_N}(x_{N-1}) - \phi_{-s_N}(x_0) \end{array}$$

with $s_i = (t_i - t_{i-1})/2$ give discrete closed curves on the periodic orbit γ. The Jacobian DF_s^∞ of F_s^∞ has a similar block structure to F_f^∞:

$$\begin{pmatrix} D_1 & -E_1 & & & & v_1 & & & & & \\ & D_2 & -E_2 & & & -v_2 & v_2 & & & & \\ & \cdots & \cdots & & & & \cdots & \cdots & & & \\ & & \cdots & \cdots & & & & \cdots & \cdots & & \\ & & & D_{N-1} & -E_{N-1} & & & & -v_{N-1} & v_{N-1} & \\ -E_N & & & & D_N & & & & & -v_N & v_N \end{pmatrix}$$

with D_i the Jacobian of ϕ_{s_i} at x_{i-1}, E_i the Jacobian of ϕ_{-s_i} at x_i and $v_i = f(\phi_{-s_i}(x_i))$. Multiplying DF_s^∞ on the left by the block diagonal matrix with entries

$$E_1^{-1}, E_2^{-1}, E_3^{-1}, \ldots, E_N^{-1}$$

gives the same matrix as DF_f^∞ above.

Restriction of F_f^∞ or F_f^d to a subspace $L \subset R^{(n+1)N}$ is equivalent to augmenting these maps with an additional linear map whose kernel gives the subspace L. There are three choices of subspace we have used in our numerical work. The method used in AUTO [10] is to fix a single linear constraint of the phase space variables x_i (called a *phase condition*) and to fix the times t_i up to a scale factor that yields the period of the periodic orbit. The second method is to fix cross-sections Σ_i orthogonal to the vector field at x_i and constrain x_i to vary in Σ_i. The Jacobian DF_f^∞ for the

augmented map is

$$J_c = \begin{pmatrix} D_1 & -I & & & & v_1 & & & & & \\ & D_2 & -I & & & -v_2 & v_2 & & & & \\ & \cdots & \cdots & & & & \cdots & \cdots & & & \\ & & \cdots & \cdots & & & & \cdots & \cdots & & \\ & & & D_{N-1} & -I & & & & -v_{N-1} & v_{N-1} & \\ -I & & & & D_N & & & & & -v_N & v_N \\ v_0^t & & & & & & & & & & \\ & v_1^t & & & & & & & & & \\ & \cdots & \cdots & & & & & & & & \\ & & \cdots & \cdots & & & & & & & \\ & & & v_{N-2}^t & & & & & & & \\ & & & & v_{N-1}^t & & & & & & \end{pmatrix}.$$

If γ is an elementary periodic orbit, then J_c is non-singular. This is proved as follows. If $J_c(y_0^t, \ldots, y_{N-1}^t, u_1, \ldots, u_N) = 0$, then

$$v_i \cdot y_i = 0,$$
$$D_{i+1} y_i - y_{i+1} = (u_i - u_{i+1}) v_{i+1},$$

for $i = 0, \ldots, N-1$ with $u_0 = 0$. Let π_i be the projection of R^n onto Σ_i. Then $\pi_i D_i y_{i-1} - \pi_i y_i = 0$. $y_0 \neq 0$ is an eigenvector of $\pi_N D_N \cdots \pi_1 D_1$ with eigenvalue 1 if it does not vanish. But $\pi_N D_N \cdots \pi_1 D_1$ is the Jacobian of the return map of Σ_0, and 1 is not an eigenvalue. We conclude that $y_0 = 0$ and consequently $\pi_1 y_1 = 0$. Together with the equation $v_1 \cdot y_1 = 0$, this implies $y_1 = 0$. Proceeding inductively in this fashion, we conclude that $y_i = 0$ for $0 \leq i < N$. Finally, we conclude that $(u_{i+1} - u_i) v_{i+1} = 0$, implying $u_i = 0$ for $1 \leq i \leq N$. Thus J_c is regular.

The third way we restrict F_f^∞ to a subspace appears to be novel, but natural. We observe that if $[(t_0, x_0), \ldots, (t_N, x_N)]$ is a discrete closed curve lying in γ, then so is the curve that replaces (t_i, x_i) by $(t_i + h, \phi_h(x_i))$. We restrict to the subspace orthogonal to these curves for $i = 1, \ldots, N-1$. For $i = 0$, we preserve the constraint $t_0 = 0$ by restricting to the orthogonal complement to the vector $(v_0, \ldots, 0; -1, \ldots -1, 0)$ corresponding to $x_0 \to \phi_h(x_0)$ and $t_i \to t_i - h$ for $i = 1, \ldots, N-1$. This gives a restricted subspace that is orthogonal to the manifold of solutions of $F_f^\infty = 0$. The block structure of the Jacobian for this method is shown in section 4.1. The proof that it is regular is very similar to the argument for J_c and is not given here.

The theory underlying the Hermite interpolation algorithm is somewhat more complex than the other algorithms described here because the Hermite polynomials on a fixed mesh interval approach a discontinuous function with increasing degree. A particular example illustrates this. Consider polynomials p_d of degree $2d+1$ with the properties that $p_d(-1) = 0$, $p_d(1) = 1$ and $p_d^{(j)}(\pm 1) = 0$ for $1 < j \leq d$. There is an explicit formula for p_d, namely $p_d = (\frac{1+x}{2})^{(d+1)} q_d$, where q_d is the degree d Taylor polynomial of $(\frac{1+x}{2})^{-(d+1)}$ at $x = 1$. Figure 2.1 shows a plot of p_{11}. The polynomials p_d are Hermite interpolants between the points $(x_0, t_0) = (0, -1)$ and $(x_1, t_1) = (1, 1)$ for the trivial vector field $\dot{x} = 0$ on R. As d increases, the p_d converge to constant functions 0 and 1 on $[-1, 0)$ and $(0, 1]$ with a "transition" layer of length comparable to $1/\sqrt{d}$. This behavior is typical.

THEOREM 2.2 (see Shen and Strang [28, 29]). *Let g and h be two functions analytic on the interval $[-1, 1]$, let g_d and h_d be the degree d Taylor polynomials of g and h at -1 and 1, and let p_d be the polynomial of degree $2d+1$ whose Taylor*

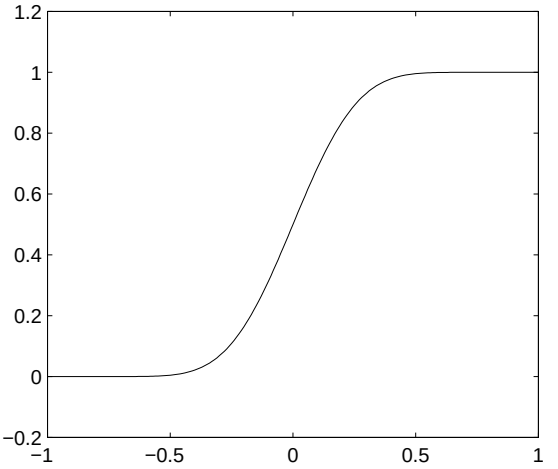


FIG. 2.1. *The Hermite interpolating polynomial of degree* 23 *connecting constant functions* 0 *and* 1 *on the interval* $[-1,1]$. *The interpolating polynomial is almost flat at the ends of the interval with a monotonic transition layer near its center.*

polynomials of degree d *at* -1 *and* 1 *agree with those of* g *and* h, *respectively. Assume that the Taylor series of* g *at* -1 *and* h *at* 1 *have radius of convergence larger than* 1. *As* d *increases*

$$p_d(x) = \begin{cases} g(x) + O(\frac{|1-x^2|^d}{\sqrt{d}|x|}), & x < 0, \\ h(x) + O(\frac{|1-x^2|^d}{\sqrt{d}|x|}), & x > 0. \end{cases}$$

Remark. The values $p_d(0)$ and $p_d'(0)$ are both linear combinations of the Taylor series coefficients of g and h. The magnitude of $p_d'(0)$ is comparable to $|g(0)-h(0)|\sqrt{d}$.

The Hermite interpolation algorithm constructs interpolating polynomials on each interval of a discrete closed curve from the Taylor series coefficients of trajectories at mesh points. The system of defining equations for the algorithm comes from collocation in each mesh interval: $p'(\tau)-f(p(\tau))=0$ with p the Hermite interpolating polynomial and τ the midpoint of the interval. To obtain a limiting set of equations for a discrete closed curve lying in a periodic orbit, these equations need to be scaled by $\sqrt{d}$ with increasing d.

THEOREM 2.2. *Let* γ *be a periodic orbit of the vector field* $\dot{x} = f(x)$ *such that* 1 *is a simple eigenvalue of its monodromy matrix. If* $\delta = [(t_0,x_0),(t_1,x_1),\ldots,(t_N,x_N)]$ *is a discrete closed curve, denote by* p_i *the degree* $2d+1$ *Hermite interpolation of the Taylor series of the trajectory segment on mesh interval* i *of* δ, *by* τ_i *the midpoint (in time) of mesh interval* i *and set* $H_i = (p_i'(\tau_i) - f(p_i(\tau_i)))/\sqrt{d}$. *If* d *is sufficiently large,* δ *is close to* γ, *and the mesh of* δ *is sufficiently fine, then the system of equations* $H_i = 0$ *with the* x_i *constrained to lie on a set of cross-sections to* γ *form a regular system of equations whose solutions converge uniformly to* γ *as* $d \to \infty$.

We give only a brief outline for the proof of this theorem. There are two aspects of the proof; namely, the analysis outlined above of how the Hermite polynomials approximate trajectory segments and demonstration that the Jacobian of the system of equations is a regular, square matrix. Denoting the degree d Taylor series approximations to γ at the points of δ by q_i, the magnitude of $(p_i'(\tau_i) - f(p_i(\tau_i)))/\sqrt{d}$ is

comparable to $|q_{i+1}(\tau_i) - q_i(\tau_i)|$. This is the leading order term of the asymptotic expansion of $(p_i'(\tau_i) - f(p_i(\tau_i)))/\sqrt{d}$ in d. Since we assume that each mesh interval is contained within the domain of convergence of the Taylor series for the trajectories at its endpoints, there is a $0 < \rho < 1$ so that $|q_i(\tau_i) - \phi(x_i, \tau_i - t_i)|$ decreases at a rate faster than ρ^d. Therefore, $|\phi(x_{i+1}, \tau_i - t_{i+1}) - \phi(x_i, \tau_i - t_i)|$ decreases at a rate faster than ρ^d as d increases, implying that the solutions of $H_i = 0$ converge uniformly to points of γ as $d \to \infty$. The Jacobian has a block matrix structure of the same kind as a symmetric multiple shooting algorithm since the leading order term of $(p_i'(\tau_i) - f(p_i(\tau_i)))/\sqrt{d}$ approaches $|\phi(x_{i+1}, \tau_i - t_{i+1}) - \phi(x_i, \tau_i - t_i)|$. The regularity of the system of equations $H_i = 0$ follows from the regularity of the Jacobian of symmetric multiple shooting with phase points restricted to a set of cross-sections to the periodic orbit.

3. Continuation and bifurcation. Introduction of parameters prompts two extensions to the algorithms of the previous section, namely,

1. computing families of solutions with changing parameters, and
2. computing bifurcations: parameter values at which the stability properties of the periodic orbits change.

This section discusses these two topics. *Continuation methods*, algorithms for computing curves of solutions to a system of k equations in $k+1$ variables have been extensively studied [1, 26]. They can be used in a straightforward way with the periodic orbit algorithms described above. The principle underlying continuation methods stems from the implicit function theorem. If a system of equations $F(x_1, \ldots, x_{k+1}) = 0$ has maximal rank, the set of solutions is a smooth curve whose tangent direction is the kernel of DF. The solution curve can be parametrized either by arc length or, locally, by a suitable coordinate x_i. Points on the solution curve can be calculated by simple predictor-corrector methods using an Euler step along the curve as a predictor and a Newton iteration in a hyperplane of R^{k+1} as a corrector. Methods for adapting the step length along the solution curves have been incorporated into computer packages for continuation [26, 10, 22]. For computing equilibria or periodic orbits of a system of ordinary differential equations, one of the coordinates in the system of equations $F(x_1, \ldots, x_{k+1}) = 0$ is a system parameter, called the *active parameter*.

Local bifurcations of periodic orbits involve a change of stability along a smooth family of periodic orbits or the collapse of a family of periodic orbits at a point of equilibrium Hopf bifurcation [18]. Global bifurcations of periodic orbits are associated with the period of an orbit becoming unbounded, either by approaching an orbit homoclinic to a saddle or by approaching a saddle-node in cycle bifurcation [18] at which there is a saddle-node equilibrium point having a trajectory that approaches the equilibrium point in both forward and backward time. Here we consider only local bifurcations of periodic orbits, i.e, those involving a change of stability in the periodic orbit. The defining equations for such bifurcations can be formulated most easily in terms of the Jacobian J of the return map for an orbit. A codimension 1 bifurcation occurs if J has an eigenvalue of modulus 1 and suitable nondegeneracy conditions are satisfied. There are three cases: saddle-node bifurcation with eigenvalue 1, period doubling (also called flip) bifurcation with eigenvalue -1, and Hopf (also called torus) bifurcation in the case of a complex pair of eigenvalues of modulus 1.

The principal algorithmic question in computing bifurcations is the formulation of defining equations. Since the defining equations are expressed in terms of the Jacobian J of the return map or the monodromy matrix of the periodic orbit, it is important to compute these matrices accurately. Our work seeks to connect the

accurate calculation of periodic orbits and their return maps with the best algorithms from linear algebra for computing defining equations for bifurcation. Where possible, we seek to reduce the defining equations to a single equation expressing the singularity of a matrix. Theoretically, this is easily done for eigenvalues ± 1: the determinants of the matrices $J \mp I$ are defining equations for the saddle-node and period doubling bifurcations. For Hopf bifurcation, the tensor product matrix $J \otimes J$ acting on skew-symmetric tensors has eigenvalues that are pairwise products of distinct eigenvalues of J. Therefore, the determinant of the matrix $J \otimes J - I \otimes I$ gives a defining equation for Hopf bifurcation. Methods for constructing defining equations for Hopf bifurcation of equilibria were described by Guckenheimer, Myers, and Sturmfels [19].

While determinants of matrices give defining equations for determining whether a matrix A is singular, there are better methods. The smallest singular value of A is a better conditioned and more reliable measure than $\det A$ of the distance of A from the set of singular matrices. Bordered matrix methods [15] provide an efficient way to compute a quantity proportional to the smallest singular value using Gaussian elimination with partial pivoting. If A has corank 1, then for generic vectors B and C, the bordered matrix

$$\begin{pmatrix} A & B \\ C^t & 0 \end{pmatrix}$$

is nonsingular. In this case, the equations

$$\begin{pmatrix} A & B \\ C^t & 0 \end{pmatrix} \begin{pmatrix} V \\ G \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix},$$

$$\begin{pmatrix} W^t & G \end{pmatrix} \begin{pmatrix} A & B \\ C^t & 0 \end{pmatrix} = \begin{pmatrix} 0 & 1 \end{pmatrix}$$

have unique solutions V, W, G with G a scalar that is proportional to the smallest singular value of A [15]. For computing saddle-node bifurcations, we shall choose matrices A related to the Jacobian DF_*^d of our system of periodic orbit equations F_*^d. It has a block structure, and is nonsingular if and only if 1 is not an eigenvalue for the return map of γ. Therefore, we border DF_*^d and add the equation $G = 0$ as the augmenting defining saddle-node bifurcation.

In applying Newton's method to the system

$$\begin{pmatrix} F_*^d \\ G \end{pmatrix} = 0$$

we need partial derivatives of F_*^d and G with respect to x_i, t_i, and λ. Derivatives of G can be reduced to computation of second derivatives of DF_*^d in this situation [15, 14]. For $z \in \{t_i, x_i, \lambda\}$,

$$G_z = -W^t \left(\frac{\partial DF_*^d}{\partial z} \right) V.$$

4. Implementations and adaptation. This section describes implementations of the algorithms described in previous sections. MATLAB 5 [24] was used to program the higher level parts of the algorithms, and ADOL-C 1.7 [16] was used to compute the Taylor series expansions of trajectories with automatic differentiation. We modified

a function in ADOL-C that computes the derivatives of the Taylor series coefficients with respect to the phase space variables so that it also computes derivatives of the Taylor series coefficients with respect to parameters. Summing the derivatives of these Taylor series coefficients gives the Jacobian of the flow map and the derivative of the flow map with respect to parameters. We have investigated several variations of the algorithms. As with numerical integration of initial value problems, a single algorithm is unlikely to be optimal for all problems and individual preferences are likely to persist in the use of different algorithms. Here we describe three algorithms, each of which appears to work well on at least some of the test problems described in the next section.

4.1. Forward multiple shooting with adaptive step lengths. The first algorithm whose implementation we describe is forward multiple shooting. The input for the algorithm is $\delta = [(t_0, x_0), (t_1, x_1), \ldots, (t_N, x_N)]$, a discrete closed curve that approximates a periodic orbit. The desired output is a new discrete closed curve $\pi = [(s_0, y_0), (s_1, y_1), \ldots, (s_N, y_N)]$ that approximates the periodic orbit with high precision. This is achieved by computing the approximate flow map $p_{t_{i+1}-t_i}(x_i)$ using the Taylor polynomial of ϕ at x_i and forming the map

$$F_f^d(\delta) = (e_0, \ldots, e_{N-1}), \qquad e_i = p_{t_{i+1}-t_i}(x_i) - x_{i+1}$$

that evaluates the difference between a point of δ and the flow from the previous point on the discrete closed curve. Restricting to the subspace defined by $x_0 = x_N$ and $t_0 = 0$, there are still N more independent variables than equations in F_f^d. We want to apply Newton's method to F_f^d. This is done by requiring that the Newton updates are orthogonal to the N vectors $f(x_i)\partial x_i + \partial t_i$ for $0 < i < N$ and $f(x_0)\partial x_0 - \sum_{i=1}^{N-1} \partial t_i$. Concretely, we use coordinates $(x_0, \ldots, x_{N-1}, t_1, \ldots, t_N)$ and add N rows to DF_f^d to obtain the matrix

$$J_o = \begin{pmatrix}
D_1 & -I & & & & v_1 & & & & & \\
 & D_2 & -I & & & -v_2 & v_2 & & & & \\
 & \cdots & \cdots & & & & \cdots & \cdots & & & \\
 & & \cdots & \cdots & & & & \cdots & \cdots & & \\
 & & & D_{N-1} & -I & & & & -v_{N-1} & v_{N-1} & \\
-I & & & & D_N & & & & & -v_N & v_N \\
v_0^t & & & & & -1 & -1 & \cdots & -1 & -1 & \\
 & v_1^t & & & & 1 & & & & & \\
 & & \cdots & & & & \cdots & & & & \\
 & & & v_{N-2}^t & & & & & 1 & & \\
 & & & & v_{N-1}^t & & & & & 1 &
\end{pmatrix}.$$

We iterate the Newton map that subtracts the vector

$$J_o^{-1}(e_0, \ldots, e_{N-1}, 0, \ldots, 0)$$

from $(x_0, \ldots, x_{N-1}, t_1, \ldots, t_N)$ to converge to a discrete closed curve that approximates the periodic orbit.

The accuracy of the solution to equations $F_f^d = 0$ as an approximation to the periodic orbit depends upon the accuracy of the time steps of numerical integration. These are computed by evaluation of Taylor polynomials with a fully adaptive step length procedure. The Taylor polynomials themselves and their Jacobians are computed with the programs `forode` and `accode` in the computer package ADOL-C [16].

These automatic differentiation algorithms use no finite difference calculations and typically produce Taylor series of high degree that are close to the numerical precision of IEEE-754 floating point arithmetic. Step lengths are chosen based upon two criteria:

1. the estimated radius of convergence of the Taylor series, and
2. agreement between the tangent vector to the Taylor polynomial and the vector field at the end of the step.

Our implementation of the first criterion assumes that the rate of growth of the computed coefficients a_i of the Taylor series is a good estimate of the radius of convergence of the Taylor series. The quantity $\rho = \sup |a_i|^{-1/i}$ over some range of degrees i is used to estimate the radius of convergence. Assuming that the degree j term in the Taylor series of the trajectory is bounded by ρ^j, we estimate a step length h with the property that the remainder of the Taylor series is smaller than the numerical precision of the computations along the step. If the size of the estimated step is larger than the time to the next mesh point, we reduce it to end at the time of the next mesh point. With this choice of step we hope that the accuracy of the computed step is within unit precision of the floating point arithmetic from the exact value. As an a posteriori check on the accuracy of the numerical trajectory, its tangent vector at the end of the step is compared with the value of the vector field at this point. If the difference between these exceeds a specified (relative or absolute) tolerance, the step size is reduced (we used a factor of 0.7) and the test repeated at the end point of the step of reduced length. If no step larger than a minimum specified step length satisfied the desired bound for the difference between tangent vector and vector field, then the algorithm halted with an error message. The map F_f^d is computed by numerically integrating each mesh point to the time of the next mesh point in this manner. As the numerical integration is done, the Jacobian of the flow is also computed, using the derivatives of the Taylor series coefficients with respect to the phase space variables, as computed with ADOL-C.

The motivation for using multiple shooting rather than single shooting stems largely from situations in which the norm of the flow map along parts of a trajectory become so large that the computation of the flow maps are horribly ill-conditioned. In extreme situations like those illustrated in the canard example of the next section, some trajectory segments within a periodic orbit are so unstable that a numerical integration cannot follow them without extreme floating point precision. We need flow maps from one mesh point to the next to be sufficiently well conditioned that small changes in the initial point of the trajectory produce small changes in its final point. We use a mesh refinement algorithm to produce such flow maps. The algorithm monitors the magnitude of the Jacobian of the flow map from the previous mesh point. When this magnitude exceeds a specified bound, then a new mesh point is inserted. The result is a discrete closed curve with imposed bounds on the magnitude of the Jacobian of the flow map from the beginning to the end of each trajectory segment.

In addition to mesh refinement, we also implement an algorithm for mesh coarsening. The strategy which is used is the following. The first point of the mesh is kept. If the last mesh point which is kept has index i, its trajectory is integrated to the time of the next mesh point. If $|p(x_i, t_{i+1} - t_i) - x_{i+1}|$ is larger than a chosen tolerance or the norm of the Jacobian of p is larger than a chosen bound, then the mesh point (t_{i+1}, x_{i+1}) is kept. If neither of these conditions fails, the numerical integration is continued to the first time t_{i+k+1} at which either $|p(x_i, t_{i+k+1} - t_i) - x_{i+k+1}|$ is larger than its tolerance or the Jacobian of p is larger than its bound. The mesh point

(t_{i+k}, x_{i+k}) with index $i+k$ is then retained and the numerical integration restarted from x_{i+k}. Finally, (t_N, x_N) is retained so that we have a discrete closed curve.

Our implementation of this forward shooting algorithm is embedded in a simple continuation framework. A parameter λ is designated as the active parameter, and an additional column giving the derivatives of F_f^d with respect to λ is added to the matrix JA, producing a matrix J_a. Computation of these derivatives required modifications to the program `accode` in ADOL-C. As the active parameter is varied, the equations $F_f^d = 0$ have an $N+1$ dimensional manifold of solutions. The curve on the solution manifold that satisfies the constraints imposed above has tangent vector lying in the null space J_a, spanned by the unit vector C. This null space is computed with MATLAB, and the discrete closed curve $[(t_0, x_0), \ldots, (t_N, x_N); \lambda]$ is augmented by adding a small multiple of C to the last computed discrete closed curve to obtain the initial seed for the next Newton iteration. When computing periodic orbits in the canard family, it is necessary to allow the parameter to vary during the Newton iteration since the amplitude of periodic orbits is extremely sensitive to parameter values. Thus we add one more row to J_a, the transpose of C, and solve for Newton updates of the variables $(x_0, \ldots, x_{N-1}, t_1, \ldots, t_N, \lambda)$. The code to implement this algorithm on top of ADOL-C is very small. Including continuation of the periodic orbits with a varying parameter, it comprises less than 600 lines of MATLAB m-files, and two C++ files to evaluate the vector field and provide the interface between MATLAB and ADOL-C used in computing Taylor series of trajectories.

4.2. Hermite polynomial interpolation. The Hermite polynomial global algorithm is somewhat more complicated than the multiple shooting algorithm described above for three reasons:

1. The algorithm constrains the phase space variables to lie on cross-sections and uses the time increments $s_i = t_i - t_{i-1}$, $0 < i \leq N$, as independent variables.
2. The calculation of the Hermite polynomials from the Taylor series at the ends of a mesh interval is more sensitive to round-off errors for high degree interpolation.
3. The defining equations for the approximate periodic orbit and their Jacobians are more complicated than for the multiple shooting algorithms.

The algorithm takes as input a discrete closed curve $\delta = [(t_0, x_0), \ldots, (t_N, x_N)]$ and seeks to compute a new discrete closed curve whose Hermite interpolation is an accurate approximation to a periodic orbit. The core of the algorithm is a Newton iteration to solve the equations $H_i = 0$ defined in our discussion of the theoretical foundations of this algorithm. We compute the Hermite polynomials by translating each mesh interval $[t_i, t_{i+1}]$ to $[-\beta, \beta]$ with $\beta = (t_{i+1} - t_i)/2$ and use a basis for the polynomials of degree $2d+1$ consisting of the polynomials

$$(1 - (t/\beta)^2)^j \qquad \text{and} \qquad (t/\beta)(1 - (t/\beta)^2)^j,\ 0 \leq j \leq d.$$

Using this normalization, the value of the interpolating polynomial and its derivative at the midpoint of the mesh interval is expressed directly as a weighted sum of the coefficients of the Taylor series at the mesh points. The Taylor series and the derivatives of the Taylor series coefficients with respect to phase space variables are computed using the routines `forode` and `accode` in the ADOL-C package. Cross-section coordinates are used in the Newton iteration. At the beginning of the Newton iteration, the vector field f is evaluated at each mesh point x_i and an orthonormal basis for the subspace Σ_i orthogonal to x_i is computed as the last $n-1$ columns of the Q factor

of the QR decomposition of the matrix $[f(x_i)\, I]$. Coordinates with respect to this basis are used to parametrize Σ_i. The Jacobian of the H_i is computed with respect to these coordinates at x_i and x_{i+1}, as well as with respect to changes in s_i, the length of mesh interval i.

The goal of the Hermite interpolation algorithm is to produce a curve p with $E(x) = |p'(x) - f(p(x))|$ smaller than a specified bound. Following convergence of the Newton iteration, we expect to meet this criterion at the endpoints and midpoints of the mesh intervals. To estimate whether the bound is satisfied uniformly along p, we evaluate E at additional points equally spaced in each mesh interval. If the desired bound on E is not met, we refine the mesh by adding to the mesh the midpoints of the mesh intervals with the largest values of E. We experimented with different criteria for determining when a mesh interval would be subdivided. The performance of the varied mesh refinement strategies appeared to be comparable to one another. We employed a different mesh coarsening strategy in this algorithm than in the forward shooting algorithm described above. Here, mesh points are deleted if the apparent errors of Hermite interpolating polynomials on the neighboring mesh intervals lie below a specified bound. We do not think that this is an optimal coarsening strategy, but it was simple to implement and its performance was satisfactory with the test problems we investigated.

4.3. Saddle-node bifurcations with multiple shooting. The next algorithm we describe is a symmetric multiple shooting algorithm for finding saddle-node bifurcations of periodic orbits. The algorithm takes a discrete closed curve that approximates a periodic orbit and a starting parameter λ_0 as input. The desired output is a new discrete closed curve $[(t_0, x_0), (t_1, x_1), \ldots, (t_N, x_N)]$ and a new parameter λ which approximates with high accuracy a periodic orbit undergoing a saddle-node bifurcation.

The method is implemented with points constrained to lie in fixed cross-sections, so we define a set of cross-sections Σ_i orthogonal to the vector field at the current mesh and introduce coordinates $w_i \in R^{n-1}$ for Σ_i. Each (half) mesh interval is traversed in a single step of length $s_i = (t_{i+1} - t_i)/2$.

We compute the approximate flow maps $p_{s_i}(w_i)$ and $p_{-s_i}(w_{i+1})$ using the degree d Taylor polynomials of ϕ at w_i and w_{i+1}. These are used to form the map $F = D_s^d$

$$F(s_0, w_0, \ldots, s_{N-1}, w_{N-1}, \lambda) = (e_0, \ldots e_{N-1}), \qquad e_i = p_{s_i}(w_i) - p_{-s_i}(w_{i+1}),$$

where e_i evaluates the difference between the trajectory segments shooting forward from mesh point ith and shooting backward from mesh point $i+1$. The Jacobian DF is nonsingular if and only if the periodic orbit is regular. Thus the singularity of DF determines points of saddle-node bifurcation. We compute the singularity of DF with a bordered matrix computation. Assume that DF has only one singular value that is close to 0. We can then border DF by a single row and column to form a matrix M whose smallest singular value is far from zero. The last component of the solution v to the matrix equation $Mv = (0, \ldots, 0, 1)^t$ is comparable to the smallest singular value of the matrix DF [15]. Denote this last component of v by $G(s, w, \lambda) = 0$. In our algorithms, G is computed from M by functions that are part of MATLAB.

We next apply Newton's method to the system

$$\begin{pmatrix} F \\ G \end{pmatrix} = 0.$$

In doing this, we need to compute the Jacobian of the system, which has the following

form:

$$K = \begin{pmatrix} DF & \frac{\partial F}{\partial \lambda} \\ \frac{\partial H}{\partial (x,\delta)} & \frac{\partial G}{\partial \lambda} \end{pmatrix}.$$

Given our modifications to ADOL-C, the computation of $\frac{\partial F}{\partial \lambda}$ is straightforward. Derivatives of G at solutions to $G = 0$ satisfy $G_z = -W^t(DF_z)V$ with W^t and V left and right eigenvectors of DF and z a component of (s, w, λ). Thus we need only compute a few of the second derivatives of F to obtain G_z. Some of these have been computed with finite differences in our codes thus far. (ADOL-C cannot be called recursively, so functions whose definition invokes ADOL-C cannot be differentiated using ADOL-C.)

Once we've computed the necessary second derivatives, we assemble the Jacobian of the system, and Newton's method is straightforward. The convergence of Newton's method appears to be limited by the condition number of K. Although DF becomes singular as we approach the bifurcation point, the condition number of K is generally well behaved. The mesh is adapted periodically during these computations as follows. To add mesh points, we begin by computing the L_∞ error $\max |p'(s_j) - f(p(s_j))|$ for a set of evenly spaced points $\{r_j\} \in [-s_i, s_i]$ in each mesh interval. If, for a particular interval joining the ith and $(i+1)$st mesh points, the error is greater than a specified multiple of the average error for all intervals, we add a new mesh point at the average of the points obtained by shooting forward from the ith mesh point and shooting backward from the $(i+1)$st mesh point. To remove mesh points, we check for each mesh point the resulting error if that point is removed, and remove it if the new error is no more than a prescribed tolerance relative to the sum of the previous errors.

The length of this code is about 600 lines of MATLAB (which includes the version of the algorithm to converge to a simple periodic orbit), and three C++ files, for evaluation of the vector field and the interface between MATLAB and ADOL-C. The appendix gives a more complete description of this algorithm in terms of pseudocode that corresponds closely to our MATLAB implementation.

5. Case studies. In this section, we present comparisons of computations on several test problems chosen to evaluate different aspects of the algorithms.

5.1. A periodic orbit in an algebraic curve. The first test problem was chosen so that the location of its periodic orbit is explicitly given by an algebraic formula. The vector field is defined as

$$\begin{aligned} \dot{x} &= y - y^2 - x(x^2 - y^2 + 2y^3/3 + c), \\ \dot{y} &= x + (y - y^2)(x^2 - y^2 + 2y^3/3 + c). \end{aligned}$$

This vector field has a stable periodic orbit that lies in the zero set of the polynomial $g(x, y) = x^2 - y^2 + 2y^3/3 + c$ when $c \in (0, 1/3)$. Thus, evaluation of g along computed trajectories is an explicit measure of their accuracy. We shall call $\max |g(u(t))|$ the *residual* of a numerically computed orbit $u(t)$, where the residual map is evaluated as a discrete function at mesh points or regarded as a continuous function using a mapping of discrete closed curves into a function space.

For $c = 0.07$, we compared three ways of computing the periodic orbit: the collocation method implemented in the AUTO package, numerical integration with a fourth-order Runge–Kutta algorithm, and a multiple shooting code that uses automatic differentiation.

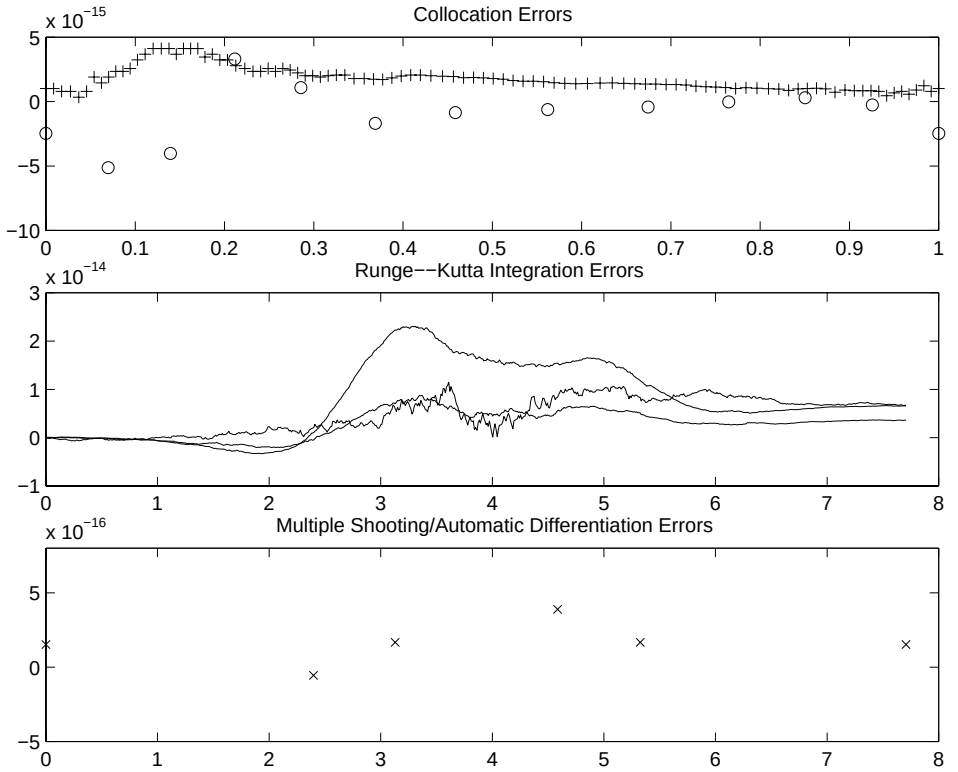


FIG. 5.1. *Three different methods have been used to compute a periodic orbit that lies in a polynomial curve in the plane. The value of the polynomial g along the periodic orbit is plotted as a function of time during one traversal of the periodic orbit. The data in the top panel were produced by AUTO and give the value of g at mesh points for (+) a mesh of* 100 *points with* 4 *collocation points per interval and for (o) a mesh of* 12 *points with* 7 *collocation points per interval. The three curves in the middle panel are the values of g at each time step of trajectories computed with a fourth-order Runge–Kutta method, using fixed time steps of* 0.00125, 0.001, *and* 0.0001 *in the three cases. The bottom panel shows the values of g produced with a multiple shooting algorithm employing automatic differentiation on a mesh with* 5 *intervals.*

The computer program AUTO [10] is widely regarded as the best available tool for the computation of periodic orbits. AUTO uses a collocation method with between 2 and 7 collocation points per mesh interval to compute orbits. The algorithm used by AUTO is superconvergent at the mesh points. The top panel of Figure 5.1 displays the value of the polynomial g at mesh points along a periodic orbit computed by AUTO. The "+" symbols were computed using 100 mesh intervals and 4 collocation points per interval. The "o" symbols were computed using 12 mesh intervals and 7 collocation points per interval. For each case of 4 and 7 collocation points per mesh interval, we tested several mesh sizes. The data in the figure have the smallest residuals we obtained.

The "standard" fourth-order Runge–Kutta algorithm [20] for the system $\dot{x} = f(x)$ with step size h is described by the following formulas that yield the approximation x_1 to $\phi_h(x_0)$:

$$k_1 = hf(x_0),$$

$$k_2 = hf(x_0 + k_1/2),$$
$$k_3 = hf(x_0 + k_2/2),$$
$$k_4 = hf(x_0 + k_3),$$
$$x_1 = x_0 + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4).$$

Setting $x_1 = \rho_h(x_0)$, $\rho_h : R^n \to R^n$ is a map that approximates the flow map ϕ_h to fourth-order in the step size h. This implies that the errors made in computing one step of the algorithm are comparable to h^5 and that the errors associated with computing a trajectory for bounded time are comparable to h^4. When h is very small, round off errors dominate the truncation errors in the finite difference calculations of derivatives of f that are implicit in the algorithm. As h is reduced, we observed that there is an optimal step size h_o for accuracy at which the round off and truncation errors appear to be balanced. This is illustrated by data from numerical experiments in which the Runge–Kutta algorithm is used to evolve the trajectory with initial point $(0, 0.2952161257895192)$ with different step sizes. The initial point lies on the periodic orbit to within round-off error, and the trajectory was computed until it returned to the y-axis near the initial point.

The apparent value of h_o is approximately 0.001. For step sizes larger than 0.001, the residual of the computed orbits decrease with h at a rate that appears consistent with the fourth-order convergence of the Runge–Kutta algorithm. For step sizes smaller than 0.001, the residual increases in an erratic manner. The second panel of Figure 5.1 shows the value of g as a function of time along Runge–Kutta trajectories with step sizes 0.00125, 0.001, and 0.0001. The smallest observed value of the residual was approximately 8×10^{-15} with step size 0.001. Since the orbit has period approximately 7.7, the numerical integration with step length 0.001 required over 7,700 steps and approximately 30,000 function evaluations. The observed optimal step size for the Runge–Kutta and mesh sizes for the AUTO calculations are consistent with predictions based upon their asymptotic orders of accuracy. The collocation method employed by AUTO is superconvergent at mesh points, with the result that the mesh points lie much closer to the desired periodic orbit than the intermediate collocation points used in the calculation. For example, the residuals of the collocation points and mesh points for our computation with seven collocation points per mesh interval and twelve mesh intervals are approximately 10^{-9} and 5×10^{-15}, respectively. The corresponding data for the orbit computed with four collocation points per mesh interval and 100 mesh intervals are 10^{-10} and 4×10^{-15}.

The bottom panel of the figure shows the results of a calculation using a symmetric multiple shooting algorithm employing automatic differentiation. There are 5 mesh intervals and the maximum value of $|g|$ is approximately 6×10^{-16}. The degree of the Taylor series polynomials used in the calculation is 16. Convergence is obtained in approximately 6 Newton steps starting with mesh points that are far from the computed solution. Thus our method produces solutions to this problem that are close to an order of magnitude more accurate than those produced by the most accurate numerical integration possible with the standard fourth-order Runge–Kutta algorithm or with the collocation method in AUTO. Given the differences in the way in which each of these algorithms was implemented, direct comparisons of their efficiency were not made. Nonetheless, it is apparent that the small mesh sizes and rapid convergence of the Taylor series method make it very efficient compared to other methods of computing periodic orbits to high accuracy.

5.2. Bifurcations in a planar vector field with multiple limit cycles. Dangelmayr and Guckenheimer [6] investigated the dynamics of the four parameter family of vector fields:

$$\begin{aligned}\dot{x} &= y,\\ \dot{y} &= -(x^3 + rx^2 + nx + m) + (b - x^2)y.\end{aligned}$$

The system describes the effect of symmetry imperfections in the unfolding of a codimension 2 bifurcation of an equilibrium with symmetry rotation by π. The dynamics of this family is surprisingly complex, with over 20 different inequivalent phase portraits. In a small region of the parameter space, there are phase portraits with a single equilibrium point and four nested limit cycles. The existence of this parameter region was deduced from an analysis of unfoldings of subsidiary codimension 3 bifurcations in Dangelmayr and Guckenheimer [6]. Parameter values at which this phase portrait occurs were first determined by Malo [23]. Malo developed simple shooting algorithms to trace curves of saddle-node bifurcations of periodic orbits. Applying these to the above system, he demonstrated that, when $r = 0.87$ and $m = -1$ the parameter region in the (n, b) plane for which the system has four nested limit cycles is a strip of width approximately 3×10^{-9}. We used this example to test our algorithms for computation and continuation of saddle-node bifurcations of periodic orbits.

Periodic orbits of the vector field with parameters $n = -1.127921667$ and $b = 0.897258546$ are displayed in Figure 5.2. The inset shows details of the three inner periodic orbits near their left intersection with the x axis. The saddle-node bifurcation points in this system were computed with the symmetric shooting method described earlier. However, the sensitivity of the problem made it a rather difficult undertaking. The shape of the three inner curves makes the computation difficult, as there is a very slow, tight turn, and a very fast, wide turn in the orbits. In order to converge from a mesh of regularly spaced points, we needed to coarsen the mesh several times before applying Newton steps. In this way, we were able to obtain representations for each of the nested periodic orbits. Each of these representations uses about 20 to 30 mesh points.

Only the middle of the three inner orbits yielded starting data sufficient to converge directly to a saddle-node bifurcation. Since we could only find one of the saddle-node bifurcations, we used a continuation strategy to find the other. We continued the first saddle-node curve with increasing n until we reached the neighborhood of an apparent cusp point. We then hoped to jump onto the second saddle-node curve and continue back to the original value of n. The computation to the cusp was quite easy: we were able to take large (size 2×10^{-6}) steps in n at first, and were able to quickly come close to the cusp point, gradually decreasing our step sizes to 10^{-8}. Once we were close to this point, we took a step of size -2.7×10^{-8}, from where the routines converged to a saddle-node bifurcation on the new curve. Following this curve back to the original value of n was significantly more difficult than computing the previous curve. We were unable to take large step sizes when we were close to the cusp because large steps converged back to the previously computed saddle-node curve. The computation was also quite sensitive to initial conditions, so we often couldn't take uniform steps. The Jacobians of the augmented systems along this curve are generally on the order of 10^{10}. Consequently, our approximations to these orbits generally have an L_∞ error on the order of 10^{-5}. Values we have computed for a portion of the saddle-node curves near Malo's parameters are given in Table 5.1.

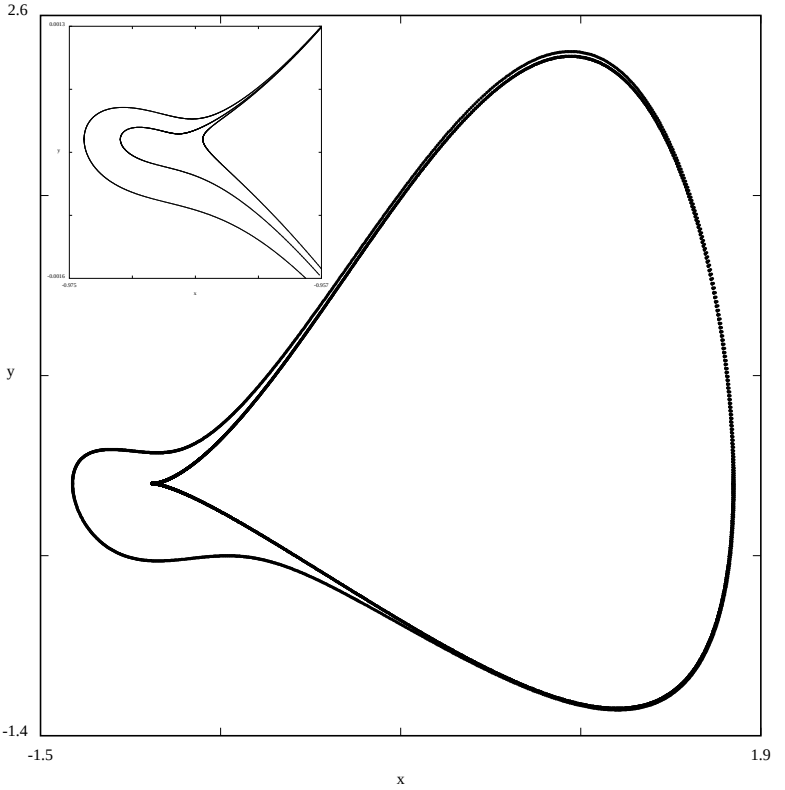


FIG. 5.2. *Four nested periodic orbits in the vector field* $y\partial_x + ((0.897258546 - x^2)y - (x^3 + 0.87x^2 + -1.127921667x - 1))\partial_y$ *The three inner orbits are not resolved in the large figure. The inset shows a small region near the left intersection of the inner orbits with the* x*-axis, bounded by the rectangle at* $[-0.975, -0.957] \times [-0.0016, 0.0013]$.

5.3. Multiple time scales and canards. The term canard describes trajectories of singularly perturbed systems that have segments which are close to unstable slow manifolds. Inspiration for the term comes from limit cycles of the dynamical system

$$\dot{x} = (y - x^2 - x^3)/\epsilon,$$
$$\dot{y} = a - x.$$

As $\epsilon \to 0$, this system has a slow manifold that approaches the cubic curve $y = x^2 + x^3$. The fast vector field is horizontal in the limit $\epsilon = 0$. The portion of the slow manifold between $x = -2/3$ and $x = 0$ is unstable, while the remainder is stable. When $a = 0$, there is an equilibrium point at the origin where the fast vector field is tangent to the slow manifold. Hopf bifurcation occurs at this point. As a decreases from 0, a family of stable periodic orbits grows rapidly with the magnitude of a. The orbits quickly stabilize as relaxation oscillations approximated by stable segments of the slow manifold and horizontal segments connecting the origin with $(-1, 0)$ and $(-2/3, 4/27)$ with $(1/3, 4/27)$. The periodic orbits of intermediate amplitude take the shape of canards in which there is a segment that follows the unstable portion of the slow manifold. An extensive analysis of the canard solutions has been undertaken from the three different perspectives of nonstandard analysis, [9], classical asymptotic

TABLE 5.1

Computed values of saddle-node bifurcation of the cubic vector field. The first column gives values of the parameter n, the second and third columns are computed values of b at the two curves of saddle-node bifurcation for the specified value of n, and the fourth column is the difference between the two values of saddle-node bifurcation.

[t!]

n	b_1	b_2	$b_2 - b_1$
-1.127921667	8.9725854413042e-01	8.9725854695863e-01	2.82821 e-09
-1.127921669	8.9725854512245e-01	8.9725854795178e-01	2.82933 e-09
-1.127921671	8.9725854611447e-01	8.9725854894492e-01	2.83045 e-09
-1.127921673	8.9725854710650e-01	8.9725854993807e-01	2.83157 e-09
-1.127921675	8.9725854809852e-01	8.9725855093121e-01	2.83269 e-09
-1.127921677	8.9725854909055e-01	8.9725855192436e-01	2.83381 e-09
-1.127921679	8.9725855008257e-01	8.9725855291751e-01	2.83494 e-09
-1.127921681	8.9725855107459e-01	8.9725855391065e-01	2.83606 e-09
-1.127921683	8.9725855206662e-01	8.9725855490380e-01	2.83718 e-09
-1.127921685	8.9725855305864e-01	8.9725855589694e-01	2.83830 e-09

analysis [12], and geometric singular perturbation theory [11]. Here we investigate the numerical computation of canards with our adaptive step length multiple shooting algorithm and compare these results with AUTO calculations.

The canard phenomenon is subtle. We recall a few of the results from the theory. Fenichel [13] established the existence of a slow manifold for positive values of ϵ as well as $\epsilon = 0$. Eckhaus [12] computed asymptotic expansions for the slow manifold. Its distance from the cubic curve $y = x^2 + x^3$ has order ϵ. Away from the turning points, trajectories flow towards or away from the cubic characteristic at exponential rates of order $1/\epsilon$. A very large region of initial points has trajectories that follow the unstable branch of the slow manifold for approximately the same distance and then jump back to a stable branch at approximately the same height. Asymptotic formulas for this critical height have been obtained. The critical height is extremely sensitive to the value of the parameter a, varying at a rate comparable to $\exp(-Q/\epsilon)$, where Q is given by the asymptotic theory. The critical height determines the size of a canard. All trajectories starting near the stable left branch of the slow manifold flow to the local maximum of the cubic characteristic and then jump to the right branch of the slow manifold. They then follow the right branch to the local minimum of the cubic curve. If the parameter a is in the range in which the canards "with heads" occur, the trajectory then climbs the unstable portion of the slow manifold to the critical height before jumping back to the left branch of the slow manifold. Thus, all trajectories that begin on the left branch of the slow manifold are swept into a small neighborhood of the canard cycle before they have completed a single circuit around the cycle. Nonetheless, tiny variations in an initial condition near the minimum of the cubic characteristic produce trajectories that separate from each other after traveling only a short distance along the unstable branch of the slow manifold.

Instability of the middle branch of the slow manifold prevents computation of canards with numerical integration. To make our discussion concrete, consider what happens with $\epsilon = 0.001$. The Jacobian of the vector field is

$$\begin{pmatrix} 1000(-2x - 3x^2) & 1000 \\ -1 & 0 \end{pmatrix}.$$

When $x \in [-1/2, -1/6]$ and $a < 0$, the x components of nearby trajectories separate at a rate at least $\exp(250t)$ while the y component of the vector field increases at a rate at most $1/2$. Thus a trajectory requires at least time $2/3$ to traverse the portion

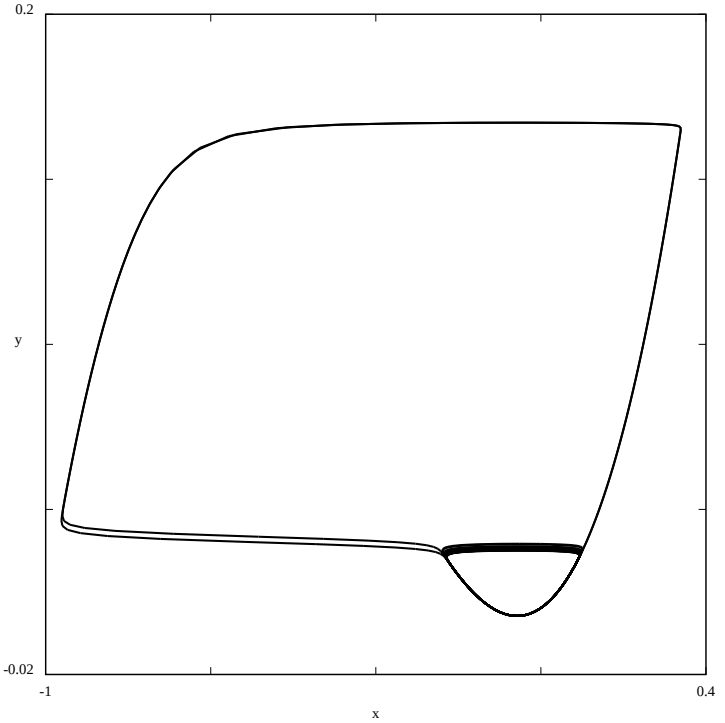


FIG. 5.3. *Fourth-order Runge–Kutta integration of the canard vector field. Due to the massive instability of the unstable branch of the slow manifold, the integration is unable to compute trajectories that follow this branch. Compare with the family of canards shown in Figure* 5.4.

of the slow manifold with $x \in [-1/2, -1/6]$. During this time, the relative separation of trajectories in the x direction increases by a factor of well over $\exp(150)$. Thus, if an initial condition has distance greater than $\exp(-150)$ from the slow manifold, it will not track it until x reaches $-1/2$. It is evident that extended precision beyond the 53 bit precision of IEEE-754 floating point arithmetic is required for this task. If we try to follow the unstable portion of the slow manifold by choosing initial conditions that lie as close together as possible on opposite sides of the manifold, then Runge–Kutta integration was observed to return to opposite sides of the unstable manifold erratically. Figure 5.3 shows such a "chaotic" trajectory. The numerical integration algorithm has qualitative behavior inconsistent with the trajectories of any planar vector field. There is no numerical trajectory that approximates the canards even crudely.

We used a forward multiple shooting method and the program AUTO to compute this family of canards. The AUTO parameters were set to use 200 mesh intervals and error tolerances of 10^{-12}. Starting at the Hopf bifurcation point, AUTO was able to compute the entire family of canard orbits. Figure 5.4 shows selected orbits from this family. We sought to evaluate the precision of the AUTO computations. To achieve this goal we took the trajectory of one canard computed in AUTO as an initial discrete closed curve for computation with an adaptive step length forward multiple

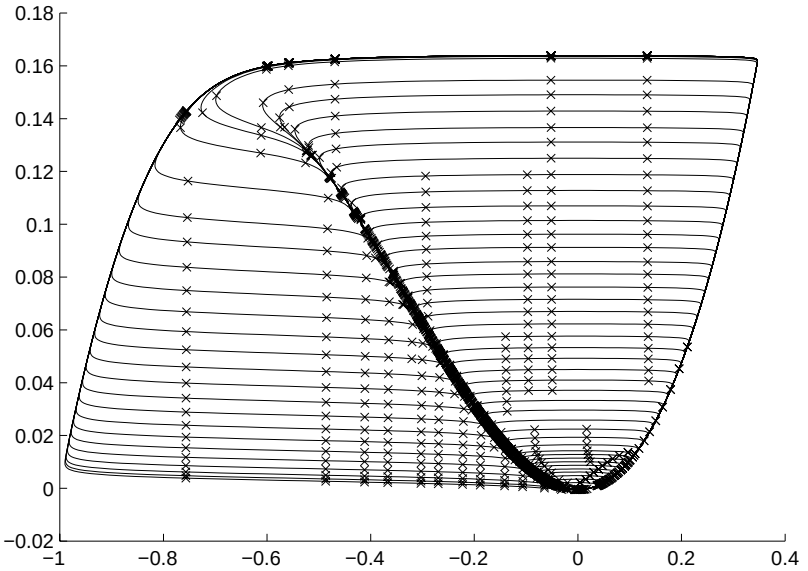


FIG. 5.4. *A family of canards computed with the forward multiple shooting algorithm described in section* 3. *The parameter* $\epsilon = 0.001$.

shooting algorithm. This algorithm first computes Taylor polynomial approximations to the trajectory segments of the discrete closed curve obtained from AUTO. These computations yield an estimate of approximately 5×10^{-9} for the maximum mismatch from the end of one trajectory segment to the initial point of the next, despite the stringent error tolerances used in the calculation and the superconvergence at mesh points of the AUTO collocation algorithm. The condition number of the Newton method Jacobian at this orbit was approximately 1×10^{50} with the parameter a fixed. The algorithm did not converge. However, fixing the period and varying a, the condition number was approximately 3.5×10^{7} and one Newton step produced an approximate discrete closed curve with the distance from the endpoint of one trajectory segment to the initial point of the next having magnitude smaller than 6×10^{-15}. The maximum length of the difference between the tangent vector to the Taylor polynomials and the vector field at the ends of the time steps is approximately 1×10^{-13}. Dividing the length of the difference by the length of the vector field produces an estimate of approximately 2×10^{-13} for the maximum deviation of the approximate periodic orbit from the flow direction.

The distance between the discrete closed curve computed by AUTO and the discrete closed curve produced by the multiple shooting algorithm is approximately 1.4×10^{-7}. Figure 5.5 displays a three-dimensional plot of the difference of the y coordinates of the two discrete trajectories. The differences between the x coordinates of the two trajectories are much smaller. The largest differences occur when the trajectory leaves the slow manifold and appear smooth enough that it is unlikely that the differences are due to round-off error. AUTO calculations with coarser meshes yield a similar pattern, with larger differences that have a similar shape along the jumps of the canard. Thus, we believe that our calculation is sufficiently precise that the difference between the two trajectories is an estimate for the distance of the mesh points computed by AUTO from the actual periodic orbit. The parameter values

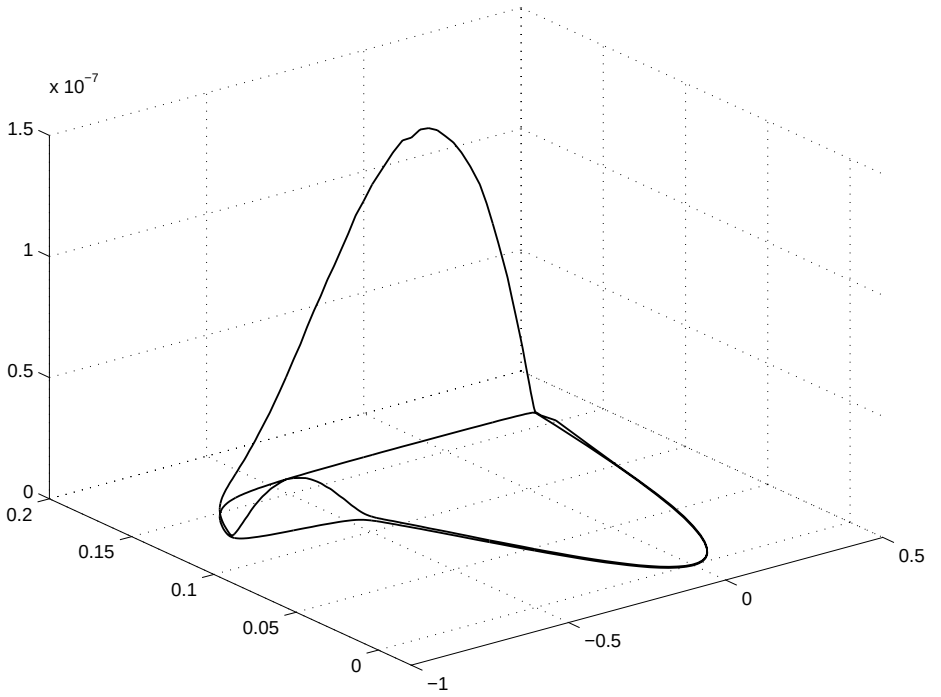


FIG. 5.5. *This three-dimensional plot gives the difference between a canard computed with AUTO and a canard computed with a Taylor series, forward shooting method. The canard is plotted in the horizontal plane and the magnitude of the difference between the two computations is plotted on the vertical axis.*

imported from AUTO for the canards had 11 significant digits. The parameter value obtained with Newton's method differs from the AUTO parameter by approximately 1.5×10^{-15}, agreeing to the last digit of the value from AUTO. We next used the period as a continuation parameter in order to track the family of canards. These calculations gave parameter values that differ in the 13th significant digit, and they do so erratically. Thus, the sensitivity of the canards to parameter variations is so great that our algorithms are unable to compute the qualitative behavior of this parameter variation. This is hardly surprising since the asymptotic theory described above estimates the parameter variation across the canard family to be smaller than 10^{-20}. Calculation of the parameter variation appears to be beyond the bounds that are readily feasible without using greater precision than IEEE-754 double precision floating point arithmetic. Our attempts to construct continuation codes for tracking the canard solutions have not achieved the robustness we seek. However, the orbits themselves appear to be computed with precision that is dominated by round-off errors as in the case of nonstiff systems.

5.4. Monodromy in a model of coupled Josephson junctions. Swift and Watanabe [30] studied the dynamics of arrays of N Josephson junction oscillators shunted by a series LRC load, a system symmetric with respect to all interchanges of the oscillators. These systems have "splay-phase" solutions in which the N oscillators all undergo the same motion but out of phase with one another. Denoting the motion

TABLE 5.2
Eigenvalues of the monodromy matrix for the Josephson junction model.

1.212564610112479e-06 ± 5.700237500982539e-08i
1.390021921820548e-06
−1.172334117548194e-03 ± 4.455213497385255e-04i
8.826221531499485e-01
1.000000000000006e+00
1.003009060195232e+00
1.149723251975266e+00 ± 5.356810539765165e-02i

of oscillator i in the splay phase state by $\phi_i(t)$, the oscillators can be numbered so that $\phi_{i+1}(t) = \phi_i(t + 2\pi/N)$. In the limit that the Stewart–McCumber parameter of these systems is zero, the splay-phase state is neutrally stable. Numerical integration of the system with other values of this parameter suggested that the splay-phase states might be neutrally stable for positive values of the Stewart–McCumber parameter as well, with a return map possibly having eigenvalue 1 with multiplicity $N - 3$. Swift and Watanabe studied the case of four coupled junctions carefully and demonstrated that the splay-phase states are represented by hyperbolic periodic orbits for generic values of the Stewart–McCumber parameter. They did so by computing the splay phase states using AUTO86 and then computing the monodromy matrix of the periodic orbit outside of AUTO. (The computation of Floquet multipliers has been greatly improved in AUTO97 compared to AUTO86, but AUTO97 was not available when Swift and Watanabe did their work.)

The vector field f that represents an array of four junctions is defined by the ten-dimensional system:

$$\begin{aligned}
\dot{x}_1 &= x_5,\\
\dot{x}_2 &= x_6,\\
\dot{x}_3 &= x_7,\\
\dot{x}_4 &= x_8,\\
\dot{x}_5 &= (I - x_5 - sin(x_1 - x_{10}))/b,\\
\dot{x}_6 &= (I - x_6 - sin(x_2 - x_{10}))/b,\\
\dot{x}_7 &= (I - x_7 - sin(x_3 - x_{10}))/b,\\
\dot{x}_8 &= (I - x_8 - sin(x_4 - x_{10}))/b,\\
\dot{x}_9 &= x_{10},\\
\dot{x}_{10} &= ((x_1 + x_2 + x_3 + x_4)/4 - rx_{10} - x_9/c)/l.
\end{aligned}$$

Since Swift and Watanabe had difficulty computing the splay phase states and their stability, we sought to confirm their results with an automatic differentiation based solver. We used the Hermite interpolation global method to compute the splay phase state, starting with linear varying phase for each oscillator. The parameter values were $I = 2.5, l = 0.75, r = 0, c = 20$, and Stewart–McCumber parameter $b = 0.2$. With 20 mesh intervals and degree 16 Taylor series approximations at the mesh points, the algorithm converged in six Newton steps to an approximate periodic orbit $p(t)$. Using the uniform norm in R^{10}, the norm of $p'(t) - f(p(t))$ appears to be smaller than 2×10^{-13}. The computed eigenvalues of the monodromy matrix are displayed in Table 5.2. The monodromy matrix of a periodic orbit has an eigenvalue 1. The precision with which this eigenvalue is computed is a measure for the accuracy of the computation of the periodic orbit and its monodromy matrix. In our calculation, the error for

this eigenvalue is 6×10^{-15}. We also note that the eigenvalue 1.003009060195232 is in excellent agreement with the perturbation analysis of Swift and Watanabe [30, Figure 8]. As a final check on the precision of these calculations, we repeated them with a coarser mesh having 10 mesh intervals. The algorithm converged in 7 Newton steps. The monodromy matrix agreed with the values in Table 5.2 above to 10 decimal places. The error in the eigenvalue 1 was approximately 5.5×10^{11}. Thus, these periodic orbit calculations seem to be converging quickly and accurately.

6. Conclusions. Numerical computation of periodic orbits of vector fields is a difficult component in understanding the qualitative structure of dynamical systems. With prevailing methods, achieving high accuracy in the direct computation of periodic orbits leads to large systems of equations through the use of large meshes. We have developed algorithms for computing periodic orbits that achieve high accuracy with small meshes through the use of high-order algorithms. A key feature of these algorithms is that increasing order does not require the introduction of larger root finding to obtain the periodic orbit. Instead, our algorithms utilize Taylor series of trajectories computed with automatic differentiation. Taylor polynomials of trajectories can be computed to arbitrary degree, and the defining equations for hyperbolic periodic orbits converge with increasing degree.

We have implemented our algorithms within MATLAB 5 together with the automatic differentiation package ADOL-C. The implementations have been tested on several problems. One of the test problems was designed to have a periodic orbit along an explicitly known polynomial curve so that the absolute accuracy of different methods for computing the periodic orbit could be tested. Our results demonstrate that the automatic differentiation based algorithms can indeed produce more accurate approximations to a periodic orbit than either numerical integration with a standard Runge–Kutta algorithm or a boundary value solver based upon collocation. We attribute the increased accuracy to the fact that there are no finite difference calculations of derivatives implicit in our methods. Moreover, the meshes used by our algorithm are coarse, reflecting the high order of the algorithms.

The three additional test problems presented in this paper were chosen for their difficulty. One of these, the planar cubic system, has limit cycles and bifurcations that are very close to one another. The second test problem shows that our methods are capable of computing canards, solutions to differential equations with multiple time scales that are not readily computed with numerical integration. The final problem tests the accuracy of computing the monodromy matrix on a higher-dimensional problem, where the desired unstable periodic orbit has a multiplier slightly larger than 1.

We are encouraged by our results. While it has been shown in the past that Taylor series methods can be competitive with other methods of numerical integration in both efficiency and accuracy [4, 5, 27], we believe that there is even more to be gained from their use in direct computation of periodic orbits as boundary value problems. The methods are very flexible with regard to mesh adaptation and conceptually simple. They are readily extended to methods that yield defining equations for bifurcations of periodic orbits. With further refinement, we look forward to our implementations becoming mature software packages that can be added to tool kits for numerical exploration of dynamical systems.

Appendix.

Main procedure: Continue saddle-node bifurcations.

Load data and parameters, and set global variables:
dim is the dimension of the phase space
data is a matrix whose rows are the mesh points (t_i, x_i)
delta is a vector containing the interval sizes (divided by 2)
nint is the number of intervals
Call procedure Find Orbit
for number of continuation points requested **do**
Call procedure Find Saddle-Node Bifurcation
Call procedure Symmetric Compute Return Map Jacobian
Save representation for the located periodic orbit: Taylor coefficients and total derivatives at the mesh points, *data*, *delta*, $ReturnMapJacobian$ and parameter values
Add continuation stepsize to the continuation parameter
if we have another point to find **then**
Call procedure Symmetric Predict Next Saddle-Node

Procedure: Find orbit.

for required number of steps **do**
Call procedure Newton Init
Call procedure Symmetric Newton Loop
Call procedure Symmetric Decimate
Call procedure Newton Init
Call procedure Symmetric Newton Loop
for required number of steps **do**
Call procedure Symmetric Refine
Call procedure Newton Init
Call procedure Symmetric Newton Loop
Let *memax* be the maximum L_∞ error over all intervals
if *memax* is small enough **then**
quit Find Orbit

Procedure: Newton init.

for all mesh points i **do**
Compute Taylor series coefficients and total derivatives (matrix collections $JMat\{i\}$) for the trajectory through the mesh point using ADOL-C
Compute an orthonormal basis for a cross-section at the mesh point by computing an orthogonal matrix whose first column is parallel to the vector field at the mesh point

---

Procedure: Symmetric Newton loop.

for required number of steps **do**

Call procedure Symmetric Evaluate
Call procedure Symmetric Newton Step

Procedure: Symmetric Evaluate.

for all mesh points i **do**
Compute Taylor series coefficients and total derivatives ($JMat\{i\}$) for the trajectories through the mesh point using ADOL-C
for all intervals i **do**
Let err(column i) be the "shooting error": the difference between shooting forward from the ith mesh point and shooting backward from the $(i+1)$st mesh point
Compute the Jacobians of the shooting error WRT the ith and $(i+1)$st mesh points
Compute the Jacobian of the shooting error WRT $delta(i)$
Form the Jacobians of the shooting error WRT the bases of cross-section variables and $delta(i)$ (matrices $D\{i\}$ and $E\{i\}$)
Let $fd\{i\}$ be the Jacobian of the approximate flow map from mesh point i to mesh point $i+1$ given by multiplying the R^n Jacobian of the flow map forward from the ith mesh point by the inverse of the R^n Jacobian of the flow map backward from the $(i+1)$st mesh point

---

Procedure: Symmetric Newton step.

Stack the columns of err into vector $errv$
Form the Jacobian J from the matrix collections D and E, with the matrices $D\{i\}$ on the diagonal and the matrices $E\{i\}$ on the superdiagonal
Solve $J \cdot Nstep = errv$ for the Newton update $Nstep$ (in cross-section coordinates)
Transform the Newton update back to R^n coordinates and update $data$ and $delta$

Procedure: Symmetric decimate.

Call procedure Symmetric Interpolate Full
Let $adddata$ be a $2nint \times (dim+1)$ array {$adddata$ will contain the decimated data}
Set $j = 1$, $id = 1$
while $id \leq nint$ **do**
$adddata$(row j) = $data$(row id)
if $id < nint$ **then**
if adjacent L_∞ errors are small enough **then**
Calculate the L_∞ error if mesh point $id+1$ is removed
if the error is small enough **then**
increment id, effectively removing mesh point $id+1$
Increment id and j
Save the last mesh point in $adddata$(row j)
Replace $data$ by $adddata$ and reset $delta$ and $nint$

Procedure: Symmetric interpolate full.

for all intervals i **do**
Let $ppts$ be an evenly spaced set of times in $[-delta(i), delta(i)]$
Let $xpts$ be the values of the shooting polynomial for the times of $ppts$
Let $xdpts$ be the values of the derivative of the shooting polynomial at the points $xpts$
Let $vpts$ be the values of the vector field at the points of $xpts$
Let $errpts$ be the differences between $vpts$ and $xdpts$
Let $emax(i)$ be the largest absolute value in $errpts$ (estimated L_∞ error)
Let $eavg$ be the average estimated L_∞ error over all of the intervals

---

Procedure: Symmetric refine.

Call procedure Symmetric Interpolate Full
Set $adddata$ to be a $2nint \times (dim+1)$ array {$adddata$ will contain the refined data}
Set $j = 1$
for $ir = 1$ to $nint$ **do**
$adddata$(row j) = $data$(row ir)
if $\frac{emax(ir)}{eavg}$ is too large **then**
Let $adddata$(row $j+1$) be the average of the forward trajectory from mesh point ir and the backward trajectory from mesh point $ir+1$
Increment j
Increment j
Save the last mesh point in $adddata$(row j)
Replace $data$ by $adddata$ and reset $delta$ and $nint$

Procedure: Find saddle-node bifurcation.

for required number of steps **do**
Call procedure Newton Init
Call procedure Symmetric Newton Saddle-Node Loop
Call procedure Symmetric Decimate
Call procedure Newton Init
Call procedure Symmetric Newton Saddle-Node Loop
for required number of steps **do**
Call procedure Symmetric Refine
Call procedure Newton Init
Call procedure Symmetric Newton Saddle-Node Loop
Let $memax$ be the largest L_∞ error over all intervals
Call procedure Symmetric Compute Return Map Jacobian
if ($ReturnMapJacobian$ is close enough to 1) and ($memax$ is small enough) **then**
quit Find Saddle-Node Bifurcation

Procedure: Symmetric Newton saddle-node loop.

for required number of steps **do**

Call procedure Symmetric Newton Saddle-Node Init
Call procedure Create Bordered Matrix
Call procedure Symmetric Newton Saddle-Node Step

Procedure: Symmetric Newton saddle-node init.

for all mesh points i **do**
Compute Taylor series coefficients and total derivatives ($JMat\{i\}$) for the trajectory through the mesh point using ADOL-C
Compute an orthonormal basis for a cross-section at the mesh point by computing an orthogonal matrix whose first column is parallel to the vector field at the mesh point
for all intervals i **do**
Let err(column i) be the "shooting error": the difference between shooting forward
from the ith mesh point and shooting backward from the $(i+1)$st mesh point
Compute the Jacobians of the shooting error WRT the ith and $(i+1)$st mesh points
Compute the Jacobian of the shooting error WRT $delta(i)$
Form the Jacobians of the shooting error WRT the bases of cross-section variables and $delta(i)$ (matrices $D\{i\}$ and $E\{i\}$)
Let $fd\{i\}$ be the Jacobian of the approximate flow map from mesh point i to mesh point $i+1$ given by multiplying the R^n Jacobian of the flow map forward
from the ith mesh point by the inverse of the R^n Jacobian of the flow map backward from the $(i+1)$st mesh point

---

Procedure: Create bordered matrix.

Create matrix M of size $(nint \cdot dim + 1) \times (nint \cdot dim + 1)$
Put J in the upper left $nint \cdot dim \times nint \cdot dim$ corner of M
Put a 1 in the upper right and lower left corners of M
if M is singular **then**
exit all procedures {something is seriously wrong}
Solve system $M\,[V^t\ G]^t = [0\ 0\ ...\ 0\ 1]^t$ for vector V and scalar G
Solve system $[W^t\ G']\,M = [0\ 0\ ...\ 0\ 1]$ for vector W and scalar G'
(as a check, $G' = G$)

Procedure: Symmetric Newton saddle-node step

Let $Kbar$ be an $(nint \cdot dim + 1) \times (nint \cdot dim + 1)$ matrix
Put matrix J in the upper left $nint \cdot dim \times nint \cdot dim$ block of $Kbar$
Stack the columns of err into vector $errv$
for all mesh points i **do**
Compute $\frac{\partial(JMat\{i\})}{\partial(activeparam)}$ using finite differencing and store the results
for all intervals i **do**
Compute the Hessians of the shooting error WRT the active parameter and the

ith
and $(i+1)$st mesh points (in cross-section coordinates), using $\frac{\partial(JMat)}{\partial(activeparam)}$
for all cross-section variables **do**
Compute the Hessians of the shooting error WRT the cross-section variable and
the ith and $(i+1)$st mesh points (in cross-section coordinates)
Compute the Hessian of the shooting error WRT $delta(i)$ and the ith and $(i+1)$st mesh points (in cross-section coordinates)
Compute the Hessian of the shooting error WRT $delta(i)$
Compute the Hessian of the shooting error WRT $delta(i)$ and the active parameter
Let $bigKz$ be an $nint \cdot dim \times nint \cdot dim$ matrix
Fill $bigKz$ with the Hessians that involve the active parameter and either a mesh point
or $delta(i)$ for some i
Set $Kbar(nint \cdot dim + 1, nint \cdot dim + 1) = -W^t \cdot bigKz \cdot V$
for all mesh points i **do**
for all cross-section variables k at mesh point i **do**
Fill Kz with the Hessians that involve cross-section variable k at mesh point i
Set $Kbar(nint \cdot dim + 1, (i-1) \cdot dim + k) = -W^t \cdot Kz \cdot V$
Fill Kz with the Hessians that involve $delta(i)$
Set $Kbar(nint \cdot dim + 1, (i-1) \cdot dim) = -W^t \cdot Kz \cdot V$
for all intervals i **do**
Compute the Jacobian of the shooting error WRT the active parameter
Insert this vector into the appropriate rows of the last column of $Kbar$
Solve $Kbar \cdot Nstep = [errv^t \mid G]^t$ for Newton update $Nstep$
Transform the Newton update to R^n coordinates and update $data$, $delta$ and the active parameter

---

Procedure: Symmetric compute return map jacobian.

Set $Monodromy$ to be a $dim \times dim$ identity matrix
for all intervals i **do**
Multiply $Monodromy$ on the left by $fd\{i\}$
Transform $Monodromy$ to $ReturnMapJacobian$ by multiplying on the left and right by the matrices to transform to cross-section coordinates

---

Procedure: Symmetric predict next saddle-node.

Call procedure Symmetric Newton Saddle-Node Init
Call procedure Create Bordered Matrix
Let $Kbar$ be an $(nint \cdot dim + 1) \times (nint \cdot dim + 1)$ matrix
Put matrix J in the upper left $nint \cdot dim \times nint \cdot dim$ block of $Kbar$
Stack the columns of err into vector $errv$
for all mesh points i **do**
Compute $\frac{\partial(JMat\{i\})}{\partial(activeparam)}$ using finite differencing and store the results
Compute $\frac{\partial(JMat\{i\})}{\partial(contparam)}$ using finite differencing and store the results
for all intervals i **do**

Compute the Hessians of the shooting error WRT the active parameter and the ith
and $(i+1)$st mesh points (in cross-section coordinates), using $\frac{\partial(JMat)}{\partial(activeparam)}$
Compute the Hessians of the shooting error WRT the cont parameter and the ith
and $(i+1)$st mesh points (in cross-section coordinates), using $\frac{\partial(JMat)}{\partial(contparam)}$
for all cross-section variables **do**
Compute the Hessians of the shooting error WRT the cross-section variable and
the ith and $(i+1)$st mesh points (in cross-section coordinates)
Compute the Hessian of the shooting error WRT $delta(i)$ and the ith and $(i+1)$st mesh points (in cross-section coordinates)
Compute the Hessian of the shooting error WRT $delta(i)$
Compute the Hessian of the shooting error WRT $delta(i)$ and the active parameter
Compute the Hessian of the shooting error WRT $delta(i)$ and the cont parameter
Let $bigKz$ be an $nint \cdot dim \times nint \cdot dim$ matrix
Fill $bigKz$ with the Hessians that involve the active parameter and either a mesh point
or a $delta(i)$ for some i
Set $Kbar(nint \cdot dim + 1, nint \cdot dim + 1) = -W^t \cdot bigKz \cdot V$
Fill $bigKz$ with the Hessians that involve the cont parameter and either a mesh point
or a $delta(i)$ for some i
Set $errv(nint \cdot dim + 1) = -W^t \cdot bigKz \cdot V$
for all mesh points i **do**
for all cross-section variables k at mesh point i **do**
Fill Kz with the Hessians that involve cross-section variable k at mesh point i
Set $Kbar(nint \cdot dim + 1, (i-1) \cdot dim + k) = -W^t \cdot Kz \cdot V$
Fill Kz with the Hessians that involve $delta(i)$
Set $Kbar(nint \cdot dim + 1, (i-1) \cdot dim) = -W^t \cdot Kz \cdot V$
for all intervals i **do**
Compute the Jacobian of the shooting error WRT the active parameter
Insert this vector into the appropriate rows of the last column of $Kbar$
Compute the Jacobian of the shooting error WRT the cont parameter
Insert this vector into the appropriate rows of the vector $errv$
Solve $Kbar \cdot Nstep = [errv^t \mid G]^t$ for Newton update $Nstep$
Transform the Newton update to R^n coordinates and update $data$, $delta$ and the active parameter

Acknowledgments. The hospitality of the Institute for Mathematics and its Applications is gratefully acknowledged. We would like to thank Kurt Lust and Sebius Doedel for their helpful comments. We would also like to thank the referees for their helpful comments and corrections to the manuscript.

REFERENCES

[1] J. Alexander and J. Yorke, *The homotopy continuation method: Numerically implementable topological procedures*, Trans. Amer. Math. Soc., 242 (1978), pp. 271–284.
[2] U. Ascher, R. Mattheij, and R. Russell, *Numerical Solution of Boundary Value Problems*, Prentice Hall, Englewood Cliffs, NJ, 1988.

[3] I. Babuška and M. Suri, *The P and H-P versions of the finite element method, basic principles and properties*, SIAM Rev., 36 (1994), pp. 578–632.
[4] Y. F. Chang and G. Corliss, *Solving ordinary differential equations using Taylor series*, ACM Trans. Math. Software, 8 (1982), pp. 114–444.
[5] G. Corliss, A. Griewank, P. Henneberger, G. Kirlinger, E. Potra, and H. J. Stetter, *High-order stiff ODE solvers via automatic differentiation and rational prediction, Numerical analysis and its applications*, Lecture Notes in Comput. Sci. 1196, Springer-Verlag, Berlin, 1997, pp. 114–125.
[6] G. Dangelmayr and J. Guckenheimer, *On a four parameter family of planar vector fields*, Arch. Ration. Mech. Anal., 97 (1987), pp. 321–352.
[7] P. Deuflhard, *Recent advances in multiple shooting techniques*, in Computational Techniques for Ordinary Differential Equations, I. Gladwell and D. K. Sayers, eds., Academic Press, New York, 1980, pp. 217–272.
[8] H.-J. Diekhoff, P. Lory, H.-J. Oberle, H.-J. Pesch, P. Rentrop, and R. Seydel, *Comparing routines for the numerical solution of initial value problems of ordinary differential equations in multiple shooting*, Numer. Math., 27 (1977), pp. 449–469.
[9] M. Diener, *Canards et bifurcations*, in Mathematical Tools and Models for Control, Systems Analysis and Signal Processing, Vol. 3, Toulouse, Paris, CNRS, 1981/1982, pp. 289–313.
[10] E. Doedel, AUTO, Available via ftp at://ftp.cs.concordia.ca/pub/doedel/auto.
[11] F. Dumortier and R. Roussarie, *Canard cycles and center manifolds*, Mem. Amer. Math. Soc., 121 (1996).
[12] W. Eckhaus, *A standard chase on French ducks*, Lecture Notes in Math. 985, 1983, pp. 449-494.
[13] N. Fenichel, *Persistence and smoothness of invariant manifolds for flows*, Indiana Univ. Math. J., 21 (1971), pp. 193-226.
[14] W. Govaerts, J. Guckenheimer, and A. Khibnik, *Defining functions for multiple Hopf bifurcations*, SIAM J. Numer. Anal., 34 (1997), pp. 1269–1288.
[15] W. Govaerts and J. D. Pryce, *A singular value inequality for block matrices*, Linear Algebra Appl., 125 (1989), pp. 141–148.
[16] A. Griewank, D. Juedes, and J. Utke, *ADOL-C: A Package for the Automatic Differentiation of Algorithms Written in C/C++, Version* 1.7, Argonne National Laboratory, Argonne, IL, 1996.
[17] J. Guckenheimer and W. G. Choe, *Computing periodic orbits with high accuracy*, Comput. Methods Appl. Mech. and Engrg., 170 (1999), pp. 331–341.
[18] J. Guckenheimer and P. Holmes, *Nonlinear Oscillations Dynamical Systems, and Bifurcation of Vector Fields*, Appl. Math. Sci. 42, Springer-Verlag, New York, 1983.
[19] J. Guckenheimer, M. Myers, and B. Sturmfels, *Computing Hopf bifurcations* I, SIAM J. Numer. Anal., 34 (1997), pp. 1–21.
[20] E. Hairer, S. P. Nørsett, and G. Wanner, *Solving Ordinary Differential Equations* I, *Nonstiff problems*, 2nd ed., New York, Springer-Verlag, Springer Ser. Comput. Math. 8, 1993.
[21] E. Hairer and G. Wanner, *Solving Ordinary Differential Equations* II, *Stiff and Different Algebraic Problems*, Springer-Verlag, Berlin, Springer Ser. Comput. Math. 14, 1991.
[22] V. Kuznetsov and V. Levitin, Available via ftp at CONTENT, //ftp.cwi.nl/pub/CONTENT.
[23] S. Malo, *Rigorous Computer Verification of Planar Vector Field Structure*, thesis, Cornell University, Ithaca, NY, 1993.
[24] MATLAB is a product of The MathWorks, Inc.
[25] D. Morrison, J. Riley, and J. Zancanaro, *Multiple shooting method for two-point boundary value problems*, Comm. ACM, 5 (1962), pp. 613–614.
[26] W. Rheinboldt, *Solution of nonlinear equations and continuation methods*, SIAM J. Numer. Anal., 17 (1980), pp. 221–237.
[27] P. Rentrop, *A Taylor series method for the numerical solution of two-point boundary value problems*, Numer. Math., 31 (1979), pp. 359–375.
[28] J. Shen and G. Strang, *Asymptotic analysis of Daubechies polynomials*, Proc. Amer. Math. Soc., 124 (1996), pp. 3819–3833.
[29] J. Shen and G. Strang, personal communication, Cornell University, Ithaca, NY, 1998.
[30] S. Watanabe and J. Swift, *Stability of periodic solutions in series arrays of Josephson junctions with internal capacitance*, J. Nonlinear Sci., 7 (1997), pp. 503–536.

SIAM J. SCI. COMPUT.
Vol. 22, No. 3, pp. 986–1007
© 2000 Society for Industrial and Applied Mathematics

NUMERICAL DISCRETIZATION OF ENERGY-TRANSPORT MODELS FOR SEMICONDUCTORS WITH NONPARABOLIC BAND STRUCTURE*

PIERRE DEGOND†, ANSGAR JÜNGEL‡, AND PAOLA PIETRA§

Abstract. The energy-transport models describe the flow of electrons through a semiconductor crystal, influenced by diffusive, electrical, and thermal effects. They consist of the continuity equations for the mass and the energy, coupled with Poisson's equation for the electric potential. These models can be derived from the semiconductor Boltzmann equation.

This paper consists of two parts. The first part concerns the modeling of the energy-transport system. The diffusion coefficients and the energy relaxation term are computed in terms of the electron density and temperature, under the assumptions of nondegenerate statistics and nonparabolic band diagrams. The equations can be rewritten in a drift-diffusion formulation which is used for the numerical discretization.

In the second part, the stationary energy-transport equations are discretized using the exponential fitting mixed finite element method in one space dimension. Numerical simulations of a ballistic diode are performed.

Key words. mixed finite element method, exponential fitting, nonparabolic band structure, semiconductors

AMS subject classifications. 65N30, 65C20, 78A35

PII. S1064827599360972

1. Introduction.

Semiconductor devices can be simulated by means of the semiconductor Boltzmann equation, which is usually numerically solved by employing the Monte Carlo method. However, this method is too costly and time consuming to model real problems in semiconductor production mode where simulation results are needed in hours or minutes. Acceptable accuracy can be reached by solving macroscopic semiconductor models derived from the Boltzmann equation. The simplest models are drift-diffusion models which consist of the mass continuity equation for the charge carriers and a definition for the particle current density (see, e.g., [25]). These models, however, are not accurate enough for submicron device modeling, owing to temperature effects, for instance.

The energy-transport equations consist of the conservation laws of mass and energy, together with constitutive relations for the particle and energy currents, and are able to model temperature effects in submicron devices. Since the energy-transport equations are of parabolic type, the numerical solution needs less effort than the hy-

*Received by the editors September 10, 1999; accepted for publication (in revised form) March 27, 2000; published electronically September 7, 2000. The first and second authors acknowledge support from the DAAD-PROCOPE Program; the second and third author are partly supported by the DAAD-Vigoni Program and by the Program on Charged Particles Kinetics at the Erwin Schrödinger Institute, Vienna. All three authors acknowledge support from the TMR Project "Asymptotic Methods in Kinetic Theory," grant ERB-FMBX-CT97-0157. The second author acknowledges support from the Gerhard-Hess Program of the Deutsche Forschungsgemeinschaft.

http://www.siam.org/journals/sisc/22-3/36097.html

†Université P. Sabatier Toulouse 3, UFR MIG, 118 route de Narbonne, F-31062 Toulouse Cedex, France (degond@mip.ups-tlse.fr).

‡Fachbereich Mathematik, Technische Universität Berlin, Straße des 17. Juni 136, D-10623 Berlin, Germany. Current address: Fachbereich Mathematik und Statistik, Universität Konstanz, Fach D193, 78457 Konstanz, Germany (juengel@fmi.uni-konstanz.de).

§Istituto di Analisi Numerica, C.N.R., Via Abbiategrasso 209, I-27100 Pavia, Italy (pietra@dragon.ian.pv.cnr.it).

drodynamic models. Moreover, the energy-transport equations can be written in a drift-diffusion formulation, therefore the numerical effort is comparable to the drift-diffusion models.

In this paper a numerical scheme for energy-transport models is presented and numerical results for a one-dimensional ballistic diode are given. The originality of this paper consists of three facts: First, we compute explicitly, for rather general band diagrams, diffusion coefficients and the energy relaxation term in terms of the electron density and temperature. For nonparabolic bands in the sense of Kane, the coefficients can be computed analytically. The resulting model is *completely* derived from the Boltzmann equation. Second, we show that *any* energy-transport model, derived from the Boltzmann equation via the *spherical harmonic expansion* (SHE) model under rather weak assumptions on the semiconductor band structure, allows a drift-diffusion formulation. Finally, based on the drift-diffusion formulation, we discretize and solve the equations by means of mixed finite elements and point out the differences of various models used in the physical literature.

The first part of this paper is concerned with the computation of the diffusion coefficients and the energy relaxation term, assuming general nonparabolic band diagrams and Boltzmann statistics (section 2). In [5] the energy-transport equations are derived from the semiconductor Boltzmann equation by means of the Hilbert expansion method. First, the SHE model is obtained in the diffusion limit, under the assumption of dominant elastic scattering. Then, through a diffusion approximation, respectively making electron-electron or phonon scattering large, the energy-transport equations are derived from the SHE model. The stationary energy-transport model reads as follows:

$$-\operatorname{div} J_1 = 0, \tag{1.1}$$

$$-\operatorname{div} J_2 = -J_1 \cdot \nabla V + W(n,T), \tag{1.2}$$

$$J_1 = L_{11}\Big(\nabla\frac{q\mu}{k_BT} - \frac{q\nabla V}{k_BT}\Big) + L_{12}\nabla\Big(-\frac{1}{k_BT}\Big), \tag{1.3}$$

$$qJ_2 = L_{21}\Big(\nabla\frac{q\mu}{k_BT} - \frac{q\nabla V}{k_BT}\Big) + L_{22}\nabla\Big(-\frac{1}{k_BT}\Big), \tag{1.4}$$

$$\varepsilon_s\Delta V = q(n-C). \tag{1.5}$$

The variables are the chemical potential μ, the electron temperature T, and the electric potential V. Furthermore, J_1, J_2 are the particle and energy current densities, respectively. The physical constants are the elementary charge q, the Boltzmann constant k_B, and the semiconductor permittivity ε_s. The electron density n depends on μ and T. For instance, for Boltzmann statistics and parabolic bands, the relation $n = N_iT^{3/2}\exp(q\mu/k_BT)$ with $N_i > 0$ holds. The space dependent function $C = C(x)$ is the doping profile, $L_{ij} = L_{ij}(n,T)$ are the diffusion coefficients, and $W = W(n,T)$ is the energy relaxation term. These equations hold in the (bounded) semiconductor domain Ω, and they have to be complemented with mixed Dirichlet–Neumann boundary conditions

$$\begin{aligned} n = n_D, \quad T = T_D, \quad V = V_D \text{ on } \Gamma_D, \\ J_1\cdot\nu = J_2\cdot\nu = \nabla V\cdot\nu = 0 \text{ on } \Gamma_N, \end{aligned}$$

modeling the Ohmic contacts Γ_D and the insulating boundary parts Γ_N. Then, $\partial\Omega = \Gamma_D \cup \Gamma_N$ and $\Gamma_D \cap \Gamma_N = \emptyset$ must be satisfied. The exterior normal unit vector on $\partial\Omega$ is denoted by ν.

The mathematical analysis of (1.1)–(1.5) has been studied recently in [11, 12, 13, 18] (see also [1, 16]). The existence and uniqueness of solutions to both the stationary and the time-dependent equations have been proved. In the physical literature, the energy-transport equations have been investigated numerically for several years [2, 9, 10, 22, 33, 34, 35], using parabolic band structure and Boltzmann statistics. Nonparabolic and non-Maxwellian distribution effects are discussed in [9, 34], but no comparisons of energy-transport models with parabolic and nonparabolic band diagrams have been performed.

In section 2 we compute the diffusion coefficients L_{ij}, the electron density n, the internal energy E, and the energy relaxation term W in terms of μ and T. We assume that the energy-band diagram of the semiconductor crystal is spherically symmetric and monotone in the modulus of the wave vector $\vec{k}$, that nondegenerate Boltzmann statistics can be used, and that a momentum relaxation time τ can be defined by $\tau(\varepsilon) \sim \varepsilon^{-\beta} N(\varepsilon)^{-1}$, where ε is the energy, $N(\varepsilon)$ denotes the density of states, and $\beta > -2$ is a parameter. Then, using the general formulas for the coefficients and densities from [5], we get more explicit expressions than those of [5], involving the energy-band function $\varepsilon(\vec{k})$ and depending on the temperature T (see section 2.2).

Furthermore, we get *analytical* expressions under the additional assumption of nonparabolic bands in the sense of Kane [20]:

$$\varepsilon(1+\alpha\varepsilon) = \frac{\hbar^2}{2m_0}|\vec{k}|^2,$$

where $\hbar$ is the reduced Planck constant, m_0 the effective electron mass, and $\alpha > 0$ the nonparabolicity parameter.

The second part of this paper is concerned with the numerical discretization of the energy-transport equations (section 3). An important observation is that the current densities can be written in a drift-diffusion formulation of the form

$$J_i = \nabla g_i(n,T) - g_i(n,T)\frac{\nabla V}{T}, \quad i = 1,2, \tag{1.6}$$

where g_1 and g_2 are nonlinear functions of n and T. This formulation is valid for any current densities coming from a SHE model involving Boltzmann statistics (see section 2.3); it is the basis for our numerical discretization. For constant temperature, the expression (1.6) reduces to the standard drift-diffusion current definition.

The continuity equations (1.6) are discretized in one space dimension with a variant of the mixed exponential fitting scheme, which has been developed and studied in [6, 7, 8, 24] for the linear drift-diffusion equations and extended to a nonlinear drift-diffusion model in [19]. The most important features of these schemes are the current conservation (the current is introduced as an independent variable and continuity is directly imposed) and the ability of well-approximating solutions with steep gradient. (The scheme introduces exponentials of electric potential differences, which automatically account for the diffusion-dominant part and the drift-dominant part of the operator.) Moreover, the ideas of the discretization are not intrinsically monodimensional, and a two-dimensional extension is currently under investigation. The current discretization (in one-dimensional) can be seen as a nonlinear Scharfetter–Gummel discretization [31]. Other generalizations of the Scharfetter–Gummel discretization can be found, for instance, in [15, 28, 29].

The numerical scheme is applied to the simulation of a one-dimensional n^+nn^+ ballistic diode, which is a simple model of the channel of a MOS transistor. In the

numerical simulations, we also use nonparabolic energy bands. The numerical experiments are performed by employing two energy-transport models, the Lyumkis and the Chen models, which are defined by different momentum relaxation time functions. These two models are already used in the physical literature but only for parabolic band diagrams.

The numerical results show that the spurious velocity overshoot spike at the anode junction becomes smaller in the nonparabolic band case, compared to the parabolic case, and almost vanishes for the Chen model. The question of velocity overshoot peaks is addressed in [14].

2. Formulation of the model.

2.1. Scaling of the equations. We bring (1.1)–(1.5) into a scaled and dimensionless form. Let C_m be the maximal value of the doping profile, ℓ^* the diameter of the device, μ_0 the low-field mobility constant, T_0 the lattice temperature, and $U_T = k_B T_0/q$ the thermal voltage. Using the scaling

$$n \to C_m n, \quad C \to C_m C, \quad T \to T_0 T, \quad V \to U_T V, \quad \mu \to U_T \mu, \quad x \to \ell^* x,$$
$$J_1 \to (q\mu_0 U_T C_m/\ell^*) J_1, \quad J_2 \to (q\mu_0 U_T^2 C_m/\ell^*) J_2,$$
$$L_{ij} \to ((qU_T)^{i+j-1}\mu_0 C_m) L_{ij}, \quad W \to (q\mu_0 U_T^2 C_m/\ell^{*2}) W,$$

we get the system

$$-\operatorname{div} J_1 = 0, \tag{2.1}$$
$$-\operatorname{div} J_2 = -J_1 \cdot \nabla V + W, \tag{2.2}$$
$$J_1 = L_{11}\Big(\nabla\frac{\mu}{T} - \frac{\nabla V}{T}\Big) + L_{12}\nabla\Big(-\frac{1}{T}\Big), \tag{2.3}$$
$$J_2 = L_{21}\Big(\nabla\frac{\mu}{T} - \frac{\nabla V}{T}\Big) + L_{22}\nabla\Big(-\frac{1}{T}\Big), \tag{2.4}$$
$$\lambda^2 \Delta V = n - C, \tag{2.5}$$

where $\lambda^2 = \varepsilon_s U_T/(qC_m\ell^{*2})$ denotes the square of the scaled Debye length.

2.2. General nonparabolic band diagrams. In this subsection we reformulate the diffusion coefficients for the energy-transport model (2.1)–(2.5), as derived in [5], and we make precise our assumptions on the energy relaxation term. We assume in this and the following subsections that all physical variables and parameters are in scaled form. In order to get more explicit expressions for the coefficients L_{ij} in terms of n, T (or μ, T), we have to impose some physical assumptions:

(H1) The energy-band diagram ε of the semiconductor crystal is spherically symmetric and a strictly monotone function of the modulus $k = |\vec{k}|$ of the wave vector $\vec{k}$. Therefore, the Brillouin zone equals $\mathbb{R}^3$ and $\varepsilon : \mathbb{R} \to \mathbb{R}$, $k \mapsto \varepsilon(k)$.

(H2) A momentum relaxation time can be defined by

$$\tau(\varepsilon) = \big(\phi_0(2N_0+1)\varepsilon^\beta N(\varepsilon)\big)^{-1}, \qquad \beta > -2, \quad \phi_0 > 0, \tag{2.6}$$

where $N(\varepsilon) = 4\pi k^2/|\varepsilon'(k)|$ is the density of states of energy $\varepsilon = \varepsilon(k)$ [5, formula (III.31)] and N_0 is the phonon occupation number [4, section 4].

(H3) The electron density n and the internal energy E are given by nondegenerate Boltzmann statistics.

The assumptions (H1)–(H2) are imposed in order to get simpler expressions for the variables. In the physical literature, the values $\beta = 0$ [9] and $\beta = 1/2$ [22] have been used in the case of parabolic band structure (see section 2.4). The nondegeneracy assumption (H3) is valid for semiconductor devices with a doping concentration which is below 10^{19}cm^{-3}. Almost all devices in practical applications satisfy this condition.

Under these assumptions, the diffusion coefficients are given by

$$L_{ij} = L_{ij}(\mu, T) = e^{\mu/T} \int_0^\infty d(\varepsilon)\varepsilon^{i+j-2} e^{-\varepsilon/T} d\varepsilon, \tag{2.7}$$

where

$$d(\varepsilon) = \frac{4\pi}{3}\tau(\varepsilon)|\varepsilon'(k)|k^2 \quad \text{and} \quad \varepsilon = \varepsilon(k)$$

(see [5, formulas (IV.17), (III.33)]). We refer to [5] for more general expressions for the diffusion coefficients under weaker assumptions.

Further, due to assumption (H3), we have [5, formula (IV.16)]

$$n = n(\mu, T) = e^{\mu/T} \int_0^\infty e^{-\varepsilon/T} N(\varepsilon) d\varepsilon, \tag{2.8}$$

$$E = E(\mu, T) = e^{\mu/T} \int_0^\infty \varepsilon e^{-\varepsilon/T} N(\varepsilon) d\varepsilon. \tag{2.9}$$

Let $\gamma(\varepsilon) = k^2$ be the inverted $\varepsilon(k)$ relation. Then $N(\varepsilon) = 2\pi\gamma(\varepsilon)^{1/2}\gamma'(\varepsilon)$ and, using (2.6),

$$d(\varepsilon) = \frac{8\pi}{3}\tau(\varepsilon)\frac{\gamma(\varepsilon)^{3/2}}{\gamma'(\varepsilon)} = \frac{4}{3\phi_0(2N_0+1)}\frac{\gamma(\varepsilon)}{\varepsilon^\beta \gamma'(\varepsilon)^2},$$

which yields

$$L_{ij} = \frac{4}{3\phi_0(2N_0+1)} e^{\mu/T} \int_0^\infty \varepsilon^{i+j-\beta-2} \frac{\gamma(\varepsilon)}{\gamma'(\varepsilon)^2} e^{-\varepsilon/T} d\varepsilon$$

or

$$L_{ij} = T^{i+j-\beta-1} e^{\mu/T} P_\beta(T, i+j) \tag{2.10}$$

with

$$P_\beta(T, \ell) = \frac{4}{3\phi_0(2N_0+1)} \int_0^\infty u^{\ell-\beta-2} \frac{\gamma(Tu)}{\gamma'(Tu)^2} e^{-u} du.$$

The electron density and the internal energy read (see (2.8), (2.9))

$$n = 2\pi e^{\mu/T} \int_0^\infty \gamma(\varepsilon)^{1/2}\gamma'(\varepsilon) e^{-\varepsilon/T} d\varepsilon,$$

$$E = 2\pi e^{\mu/T} \int_0^\infty \varepsilon\gamma(\varepsilon)^{1/2}\gamma'(\varepsilon) e^{-\varepsilon/T} d\varepsilon,$$

or

$$n = Te^{\mu/T} Q(T, 0), \qquad E = T^2 e^{\mu/T} Q(T, 1) \tag{2.11}$$

with

$$Q(T,\ell) = 2\pi \int_0^\infty u^\ell \gamma(Tu)^{1/2} \gamma'(Tu) e^{-u} du.$$

The energy relaxation term is given by

$$W = \int_0^\infty S_1(e^{(\mu-\varepsilon)/T}) \varepsilon d\varepsilon,$$

where S_1 is the phonon collision operator [5, formula (IV.18)]. In the Fokker–Planck approximation, we can write this operator as (see [32])

$$S_1(e^{(\mu-\varepsilon)/T}) = \frac{\partial}{\partial \varepsilon} \left\{ \delta(\varepsilon) \left[\left(1 + T_0 \frac{\partial}{\partial \varepsilon} \right) e^{(\mu-\varepsilon)/T} \right] \right\},$$

where $\delta(\varepsilon) = \phi_0 \varepsilon^\beta N(\varepsilon)^2$, $\beta > -1$, and $T_0 = 1$ is the (scaled) ambient temperature. With the definition of $\delta(\varepsilon)$, the above expression can be simplified:

$$\begin{aligned} W &= -e^{\mu/T} \int_0^\infty \delta(\varepsilon) e^{-\varepsilon/T} \left(1 - \frac{T_0}{T} \right) d\varepsilon \\ &= \phi_0 e^{\mu/T} T^\beta (T_0 - T) \int_0^\infty u^\beta N(Tu)^2 e^{-u} du \\ &= 4\pi^2 \phi_0 e^{\mu/T} T^\beta (T_0 - T) \int_0^\infty \gamma(Tu) \gamma'(Tu)^2 u^\beta e^{-u} du. \end{aligned}$$

Introducing

$$R_\beta(T) = \int_0^\infty \gamma(Tu) \gamma'(Tu)^2 u^\beta e^{-u} du, \tag{2.12}$$

the energy relaxation term can be written as

$$W = \frac{3}{2} \frac{n(T_0 - T)}{\tau_\beta(T)},$$

with the temperature-dependent relaxation time

$$\tau_\beta(T) = \frac{3}{8\pi^2 \phi_0} \frac{T^{1-\beta} Q(T,0)}{R_\beta(T)}. \tag{2.13}$$

2.3. A drift-diffusion formulation for the current densities. A remarkable observation is that the current densities J_1 and J_2 can be written in a drift-diffusion formulation of the type

$$J_1 = \nabla g_1(n,T) - g_1(n,T) \frac{\nabla V}{T}, \tag{2.14}$$

$$J_2 = \nabla g_2(n,T) - g_2(n,T) \frac{\nabla V}{T}. \tag{2.15}$$

(Here and in the following, the gradient ∇ always means differentiation with respect to the space variable.) Indeed, in the general case the current densities are given by

$$J_i = \int_0^\infty d(\varepsilon) \left(\nabla e^{(\mu-\varepsilon)/T} + \nabla V \frac{\partial}{\partial \varepsilon} e^{(\mu-\varepsilon)/T} \right) \varepsilon^{i-1} d\varepsilon, \quad i = 1, 2. \tag{2.16}$$

This relation holds true under weak assumptions (see [5] for details) and in particular under the assumptions (H1)–(H3) of section 2.2.

From (2.16) we get

$$J_i = \nabla \int_0^\infty d(\varepsilon) e^{(\mu-\varepsilon)/T} \varepsilon^{i-1} d\varepsilon - \frac{\nabla V}{T} \int_0^\infty d(\varepsilon) e^{(\mu-\varepsilon)/T} \varepsilon^{i-1} d\varepsilon,$$

which equals (2.14), (2.15), respectively, setting

$$g_1 = \int_0^\infty d(\varepsilon) e^{(\mu-\varepsilon)/T} d\varepsilon, \qquad g_2 = \int_0^\infty d(\varepsilon) e^{(\mu-\varepsilon)/T} \varepsilon d\varepsilon.$$

The functions g_1 and g_2 can be computed in terms of n and T, under the assumptions (H1)–(H3) of section 2.2. Indeed, by (2.7), we get $g_1 = L_{11}$ and $g_2 = L_{21}$, and using (2.10) and (2.11), we can write

$$g_1(n,T) = \frac{P_\beta(T,2)}{Q(T,0)} T^{-\beta} n, \qquad g_2(n,T) = \frac{P_\beta(T,3)}{Q(T,0)} T^{1-\beta} n \tag{2.17}$$

or

$$g_1(n,T) = \mu_\beta^{(1)}(T) T n, \qquad g_2(n,T) = \mu_\beta^{(2)}(T) T^2 n$$

with the temperature-dependent mobilities

$$\mu_\beta^{(i)}(T) = \frac{P_\beta(T,i+1)}{Q(T,0)} T^{-1-\beta}, \quad i = 1,2. \tag{2.18}$$

We can write the stationary energy-transport model in the drift-diffusion formulation either in the variables n, T, and V or in the variables g_1, g_2, and V. In both cases only the current density relations change. In the former case we have

$$J_1 = \nabla(\mu_\beta^{(1)}(T) T n) - \mu_\beta^{(1)}(T) n \nabla V,$$
$$J_2 = \nabla(\mu_\beta^{(2)}(T) T^2 n) - \mu_\beta^{(2)}(T) T n \nabla V,$$

and in the latter case

$$J_i = \nabla g_i - \frac{g_i}{T(g_1,g_2)} \nabla V, \quad i = 1,2.$$

The electron density is given in terms of g_1 and g_2 by (see (2.17))

$$n(g_1,g_2) = \frac{Q(T(g_1,g_2),0)}{P_\beta(T(g_1,g_2),2)} T(g_1,g_2)^\beta g_1. \tag{2.19}$$

The energy relaxation term in the variables g_1 and g_2 now writes (recall that $T_0 = 1$)

$$W = \frac{3}{2\tau_\beta(T)T} \left(\frac{g_1}{\mu_\beta^{(1)}(T)} - \frac{g_2}{\mu_\beta^{(2)}(T)} \right). \tag{2.20}$$

In order to compute the electron temperature in terms of g_1 and g_2, we have to invert the following function (see (2.17)):

$$f(T) \stackrel{\text{def}}{=} \frac{P_\beta(T,3)}{P_\beta(T,2)} T = \frac{g_2}{g_1}. \tag{2.21}$$

This is possible if and only if the derivative of f is positive for all $T > 0$. The following lemma shows that this is true if and only if the diffusion matrix (L_{ij}) is positive definite. Now, this property has to be satisfied in order to get a well-posed mathematical problem.

LEMMA 2.1. *Let the hypotheses* (H1)–(H3) *hold. Then*

$$f'(T) = \frac{\det(L_{ij})}{(Tg_1)^2}. \tag{2.22}$$

Proof. Using the relation

$$TP'_\beta(T,\ell-1) = P_\beta(T,\ell) - (\ell-\beta-2)P_\beta(T,\ell-1),$$

which can be proved by integration by parts, we obtain

$$f'(T) = P_\beta(T,2)^{-2}[P_\beta(T,4)P_\beta(T,2) - P_\beta(T,3)^2].$$

Then, from the formulas

$$\det(L_{ij}) = e^{2\mu/T}T^{4-2\beta}[P_\beta(T,4)P_\beta(T,2) - P_\beta(T,3)^2]$$

and $n = Q(T,0)Te^{\mu/T}$ (see (2.10) and (2.11)), it follows that

$$f'(T) = \left(\frac{Q(T,0)T^{\beta-1}}{P_\beta(T,2)n}\right)^2 \det(L_{ij}) = \frac{\det(L_{ij})}{(Tg_1)^2}. \qquad \square$$

For later reference, we rewrite the complete energy-transport model in the (g_1, g_2, V) formulation:

$$-\text{div } J_1 = 0, \tag{2.23}$$

$$-\text{div } J_2 = -J_1\cdot\nabla V + W, \tag{2.24}$$

$$J_1 = \nabla g_1 - \frac{g_1}{T}\nabla V, \tag{2.25}$$

$$J_2 = \nabla g_2 - \frac{g_2}{T}\nabla V, \tag{2.26}$$

$$\lambda^2\Delta V = n - C(x) \quad \text{in } \Omega, \tag{2.27}$$

subject to the mixed Dirichlet–Neumann boundary conditions

$$g_1 = g_{D,1}, \quad g_2 = g_{D,2}, \quad V = V_D \text{ on } \Gamma_D, \tag{2.28}$$

$$J_1\cdot\nu = J_2\cdot\nu = \nabla V\cdot\nu = 0 \text{ on } \Gamma_N, \tag{2.29}$$

where we have set $g_{D,i} = g_i(n_D, T_D)$, $i = 1, 2$. The functions n and W depend on g_1 and g_2 according to (2.19) and (2.20), respectively. The dependence of T on g_1 and g_2 is given by the nonlinear equation (2.21).

2.4. A nonparabolic band approximation. In this section we compute the diffusion coefficients and the energy relaxation term for nonparabolic band diagrams in the sense of Kane. Related models incorporating nonparabolic bands can be found in [9, 36]. Moreover, we show that for the parabolic band approximation, we get the same relations as in the physical literature [9, 22].

The nonparabolic band structure in the sense of Kane [20] is defined as follows:

(H4) Let the energy $\varepsilon(k)$ satisfy

$$\varepsilon(1+\alpha\varepsilon) = \frac{k^2}{2m^*}.$$

The constant m^* is the (scaled) effective electron mass given by $m^* = m_0 k_B T_0/\hbar^2 k_0^2$, where m_0 is the unscaled effective mass, k_0 is a typical wave vector, and $\alpha > 0$ is the (scaled) nonparabolicity parameter. Notice that we get a parabolic band diagram if $\alpha = 0$.

The assumption (H4) implies $\gamma(Tu) = 2m^*Tu(1+\alpha Tu)$, and by introducing the functions

$$p_\beta(\alpha T,\ell) = \int_0^\infty \frac{1+\alpha Tu}{(1+2\alpha Tu)^2} u^{\ell-\beta-1}e^{-u}du,$$
$$q(\alpha T,\ell) = \int_0^\infty (1+\alpha Tu)^{1/2}(1+2\alpha Tu)u^{1/2+\ell}e^{-u}du,$$

we can rewrite P_β and Q as (see section 2.2)

$$P_\beta(T,\ell) = \frac{2}{3\phi_0(2N_0+1)m^*}Tp_\beta(\alpha T,\ell),$$
$$Q(T,\ell) = 2\pi(2m^*)^{3/2}T^{1/2}q(\alpha T,\ell).$$

Therefore, the electron density and internal energy from (2.11) become

$$n = N(T)T^{3/2}e^{\mu/T}, \quad E = \frac{q(\alpha T,1)}{q(\alpha T,0)}Tn,$$

where $N(T) = 2\pi(2m^*)^{3/2}q(\alpha T,0)$. For the mobilities (2.18) we get the expressions

$$\mu_\beta^{(i)}(T) = \mu_0\frac{p_\beta(\alpha T,i+1)}{q(\alpha T,0)}T^{-1/2-\beta}, \quad i=1,2.$$

Here, the mobility constant μ_0 is given by

$$\mu_0 = \left(3\pi\phi_0(2N_0+1)m^*(2m^*)^{3/2}\right)^{-1}.$$

Furthermore, introducing

$$r_\beta(\alpha T) = \int_0^\infty (1+\alpha Tu)(1+2\alpha Tu)^2u^{1+\beta}e^{-u}du,$$

we obtain (see (2.12))

$$R_\beta(T) = (2m^*)^3 r_\beta(T)T,$$

and the energy relaxation time (2.13) becomes

$$\tau_\beta(T) = \tau_0\frac{3q(\alpha T,0)}{2r_\beta(\alpha T)}T^{1/2-\beta},$$

where

$$\tau_0 = \left(2\pi\phi_0(2m^*)^{3/2}\right)^{-1}.$$

Notice that the function r_β is in fact a polynomial:

$$r_\beta(\alpha T) = \Gamma(\beta+2) + 5\Gamma(\beta+3)\alpha T + 8\Gamma(\beta+4)(\alpha T)^2 + 4\Gamma(\beta+5)(\alpha T)^3.$$

The symbol Γ denotes the Gamma function defined by

$$\Gamma(s) = \int_0^\infty u^{s-1}e^{-u}du, \quad s > 0.$$

(Here we use the hypothesis $\beta > -2$.)

Finally, the energy relaxation term (2.20) can be rewritten as

$$W = \frac{T^{2\beta-1}}{\mu_0\tau_0} r_\beta(\alpha T)\left(\frac{g_1}{p_\beta(\alpha T, 2)} - \frac{g_2}{p_\beta(\alpha T, 3)}\right).$$

In the parabolic band approximation case ($\alpha = 0$) the above expressions simplify. Since $q(0,0) = \Gamma(3/2) = \sqrt{\pi}/2$ and $q(0,1) = \Gamma(5/2) = 3\sqrt{\pi}/4$, we get for the electron density and the internal energy the well-known relations

$$n = (2\pi m^*)^{3/2}T^{3/2}e^{\mu/T}, \quad E = \frac{3}{2}Tn.$$

In order to compute the mobilities and the energy relaxation time, we have to specify the parameter β. As mentioned in section 2.2, in the literature the values $\beta = 1/2$ (used by Chen et al. [9]) and $\beta = 0$ (used by Lyumkis et al. [22]) have been employed.

First let $\beta = 1/2$. Then $p_{1/2}(0,2) = \sqrt{\pi}/2$ and $p_{1/2}(0,3) = r_{1/2}(0) = 3\sqrt{\pi}/4$ and therefore,

$$\mu_{1/2}^{(1)}(T) = \mu_0 T^{-1}, \quad \mu_{1/2}^{(2)}(T) = \frac{3}{2}\mu_0 T^{-1}, \quad \tau_{1/2}(T) = \tau_0.$$

Hence, we get the same current density relations and the same energy relaxation term as Chen et al. in [9]:

$$\begin{aligned} J_1 &= \mu_0\Big(\nabla n - \frac{n}{T}\nabla V\Big), \\ J_2 &= \frac{3}{2}\mu_0\Big(\nabla(nT) - n\nabla V\Big), \\ W &= \frac{3}{2}\,\frac{n(T_0 - T)}{\tau_0}. \end{aligned}$$

The energy-transport model with the above relations will be called the *Chen model*.

When $\beta = 0$, we have $p_0(0,2) = r_0(0) = 1$, $p_0(0,3) = 2$, and

$$\mu_0^{(1)}(T) = \frac{2\mu_0}{\sqrt{\pi}}T^{-1/2}, \quad \mu_0^{(2)}(T) = \frac{4\mu_0}{\sqrt{\pi}}T^{-1/2}, \quad \tau_0(T) = \frac{3\sqrt{\pi}}{4}\tau_0 T^{1/2},$$

so that the current densities and the energy relaxation term become

$$\begin{aligned} J_1 &= \frac{2\mu_0}{\sqrt{\pi}}\Big(\nabla(nT^{1/2}) - \frac{n}{T^{1/2}}\nabla V\Big), \\ J_2 &= \frac{4\mu_0}{\sqrt{\pi}}\Big(\nabla(nT^{3/2}) - nT^{1/2}\nabla V\Big), \\ W &= \frac{2}{\sqrt{\pi}}\,\frac{n(T_0 - T)}{\tau_0 T^{1/2}}. \end{aligned}$$

The energy transport equations with these expressions will be called the *Lyumkis model*.

We conclude this section with a remark on the choice of the parameters. In order to determine the energy-transport model completely, the parameters α, β, ϕ_0, N_0, and k_0 have to be chosen. The mobility constant μ_0 depends on ϕ_0, N_0, and k_0 (the dependence on k_0 comes in via m^*), and the constant τ_0 depends on ϕ_0 and k_0. Instead of choosing the parameters ϕ_0, N_0, and k_0, we prescribe μ_0 and τ_0 whose values (depending on the semiconductor material) can be derived from physical experiments.

3. Numerical approximation. In the following we describe in detail the discretization of the one-dimensional energy flux continuity equations (2.24), (2.26) by means of an exponential fitting mixed finite element method. The discretization of (2.23), (2.25) is similar but simpler (since the zeroth order term and the right-hand side of (2.23) are zero). The Poisson equation (2.27) is discretized with a P_1 finite element scheme. Consequently, in the following, V denotes a piecewise linear function and V_x its (piecewise constant) derivative. The exponential fitting mixed finite element method introduced for the drift-diffusion continuity equation (cf. [6, 7, 8, 24]) can be sketched as follows: (i) transformation of the problem by means of the Slotboom variable to a symmetric form; (ii) discretization of the symmetric form with mixed finite elements (consequently, the flux is introduced as an independent variable); (iii) suitable discrete change of variable to rewrite the equations in terms of the original variables g_2. Due to the nonconstant electron temperature, a Slotboom variable does not exist in the present case. As a starting point of the discretization scheme we define a "local" Slotboom variable, assuming that the temperature is a prescribed piecewise constant function defined in the global iteration process. We refer to the end of the section for an explicit choice of the procedure. A related idea has been used in [19] for the discretization of the nonlinear drift-diffusion continuity equation.

More precisely, introduce a partition $0 = x_0 < x_1 < \cdots < x_N = 1$ of $(0,1)$ and set $I_i = (x_{i-1}, x_i)$, $h_i = x_i - x_{i-1}$ for $i = 1, \ldots, N$, and $h = \max_i h_i$. We denote by $\overline{T}$ the piecewise constant approximation of the temperature (see (3.22) for the precise definition). The equations to be solved are then

$$J_2 = (g_2)_x - g_2 V_x/\overline{T}, \tag{3.1}$$

$$-(J_2)_x + \overline{c}_2 g_2 = -J_1 V_x + \overline{c}_1 g_1, \tag{3.2}$$

where we set

$$\overline{c}_\ell = \frac{3}{2\overline{T}\tau_\beta(\overline{T})\mu_\beta^{(\ell)}(\overline{T})},$$

for $\ell = 1, 2$, and, for simplicity of notation, we denote again by J_ℓ, g_ℓ, for $\ell = 1, 2$, the variables.

In each interval I_i, "local" Slotboom variables are introduced by

$$y_2 = e^{-V/\overline{T}} g_2 \quad \text{in } I_i, \tag{3.3}$$

and (3.1) and (3.2) are written in the interval I_i as

$$e^{-V/\overline{T}} J_2 - (y_2)_x = 0, \tag{3.4}$$

$$-(J_2)_x + \overline{c}_2 e^{V/\overline{T}} y_2 = -J_1 V_x + \overline{c}_1 g_1. \tag{3.5}$$

A similar idea for the transformation of the energy-transport equations has been used in [17]. Jerome [15] and Jerome and Shu [16] have employed a slightly different Slotboom transformation by introducing $\phi(x) = \int_0^x V_x(s)/T(s)ds$.

To define the mixed finite element scheme we follow [24], where a monotonic scheme for the two-dimensional current continuity equation in the presence of zeroth order term has been developed. The finite dimensional space for the flux variable contains functions of $L^2(\Omega)$, which are in each interval polynomials of the form $\sigma(x) = a_i + b_i P_i(x)$, with a_i, b_i constant and $P_i(x)$ a second order polynomial uniquely defined in the interval I_i as follows. Let $P(x)$ be the second order polynomial with the following properties:

$$\int_0^1 P(x)dx = 0, \quad P(0) = 0, \quad P(1) = 1, \tag{3.6}$$

that is, $P(x) = 3x^2 - 2x$. Moreover, it holds that $\int_0^1 P'(x)dx = 1, \int_0^1 P(x)^2 dx = \frac{2}{15}$. We define $P_i(x)$ (depending on V) by

$$P_i(x) = -P\left(\frac{x_i - x}{h_i}\right) \qquad \text{if } i_{min} = i - 1, \tag{3.7}$$

$$P_i(x) = P\left(\frac{x - x_{i-1}}{h_i}\right) \qquad \text{if } i_{min} = i, \tag{3.8}$$

where i_{min} is the point of minimum of the potential $V(x)$ in the interval I_i. We shall denote by V_{min} its minimum value. Notice that the minimum is always attained at one end point of the interval, since V is linear in I_i. If $V(x)$ is constant in I_i, we define $P_i(x) = P(\frac{x - x_{i-1}}{h_i})$.

Let us introduce the following finite dimensional spaces:

$$\begin{aligned} X_h &= \{\sigma \in L^2(\Omega) \,:\, \sigma(x) = a_i + b_i P_i(x) \text{ in } I_i,\ i = 1, \ldots, N\}, \\ W_h &= \{\xi \in L^2(\Omega) \,:\, \xi \text{ is constant in } I_i,\ i = 1, \ldots, N\}, \\ \Lambda_{h,\chi} &= \{q \text{ is defined at the nodes } x_0, \ldots, x_N\, q(x_0) = \chi(0),\ q(x_N) = \chi(1)\}. \end{aligned}$$

The mixed-hybrid approximation of (3.1)–(3.2) is then as follows:

Find $J_2^h \in X_h$, $\overline{g}_2^h \in W_h$, $g_2^h \in \Lambda_{h,g_{D,2}}$, such that

$$\sum_{i=1}^N \left(\int_{I_i} A_i J_2^h \sigma + \int_{I_i} B_i \overline{g}_2^h \sigma_x - \left[e^{-V/\overline{T}} g_2^h \sigma \right]_{x_{i-1}}^{x_i} \right) = 0, \tag{3.9}$$

$$\sum_{i=1}^N \left(-\int_{I_i} (J_2^h)_x \xi + \int_{I_i} \overline{c}_2 \overline{g}_2^h \xi \right) = \sum_{i=1}^N \int_{I_i} (-J_1^h V_x + \overline{c}_1 \overline{g}_1^h)\xi, \tag{3.10}$$

$$\sum_{i=1}^N [q J_2^h]_{x_{i-1}}^{x_i} = 0 \tag{3.11}$$

for all $\sigma \in X_h$, $\xi \in W_h$, $q \in \Lambda_{h,0}$. $J_1^h \in X_h$ is the approximation of the current density J_1, $\overline{g}_1^h \in W_h$ is the piecewise constant approximation of g_1, stemming from the discretization of the current continuity equation (see (3.19)–(3.21) below). In the first equation A and B denote the piecewise constant functions (approximation of

$e^{-V/\overline{T}}$) defined in each interval I_i by

$$A|_{I_i} = A_i \overset{\text{def}}{=} \frac{1}{h_i}\int_{x_{i-1}}^{x_i} e^{-V(s)/\overline{T}}ds, \quad i=1,\ldots,N,$$
$$B|_{I_i} = B_i \overset{\text{def}}{=} e^{-V_{min}/\overline{T}}, \qquad\qquad i=1,\ldots,N.$$

J_2^h is an approximation of the energy flux J_2, $\overline{g}_2^h$ is a piecewise constant approximation of g_2, and g_2^h is an approximation of g_2 at the nodes (see [3, 23]). The first equation is obtained from a weak version of (3.4), using integration by parts and summation over all I_i together with the inverse of the Slotboom transformation (3.3). Notice that the discrete inverse transformation is not the same for the variables $\overline{g}_2^h$ and g_2^h. We refer to [24] for a detailed discussion on the need of different approximations of the exponential function due to (possibly) large values of V_x. The second equation is a discrete weak version of (3.2), obtained from (3.5), where $e^{V/\overline{T}}$ is approximated by B^{-1} and the discrete inverse Slotboom transformation for $\overline{g}_2^h$ is used. The third equation implies the continuity of J_2^h at the nodes.

The variables J_2^h and $\overline{g}_2^h$ can be eliminated a priori by static condensation, leading to a final algebraic system in the variables g_2^h only. We write $J_2^h \in X_h$ as

$$J_2^h(x) = J_{2,i}^0 + J_{2,i}^1 P_i(x) \quad \text{for } x \in I_i, \tag{3.12}$$

for some constants $J_{2,i}^0$, $J_{2,i}^1$, $i=1,\ldots,N$. Set $\overline{g}_{2,i} = \overline{g}_2^h|_{I_i}$, $g_{2,i} = g_2^h(x_i)$, $V_i = V(x_i)$, and $q_i = q(x_i)$. Taking $\sigma \in X_h$ such that $\sigma = 1$ in I_i and $\sigma = 0$ elsewhere in (3.9) gives

$$h_i A_i J_{2,i}^0 = e^{-V_i/\overline{T}} g_{2,i} - e^{-V_{i-1}/\overline{T}} g_{2,i-1}.$$

The integral in the definition of A_i can be computed explicitly and we arrive after elementary computations at

$$J_{2,i}^0 = \frac{V_i - V_{i-1}}{2\overline{T}} \coth\left(\frac{V_i - V_{i-1}}{2\overline{T}}\right)\frac{g_{2,i} - g_{2,i-1}}{h_i} - \frac{g_{2,i} + g_{2,i-1}}{2\overline{T}}\frac{V_i - V_{i-1}}{h_i}. \tag{3.13}$$

This discretization can be seen as a nonlinear Scharfetter–Gummel scheme (cf. [6]). The constants $J_{2,i}^1$ are computed by using (3.10) once $\overline{g}_{2,i}$ is given. Indeed, taking $\xi = 1$ in I_i and $\xi = 0$ elsewhere in (3.10), it follows that

$$J_{2,i}^1 = \overline{c}_2 h_i \overline{g}_{2,i} - h_i r_i, \tag{3.14}$$

where we set $r_i = \frac{1}{h_i}\int_{I_i}(-J_1^h V_x + \overline{c}_1 \overline{g}_1^h)dx$. Taking now $\sigma \in X_h$ such that $\sigma = P_i(x)$ in I_i and $\sigma = 0$ elsewhere in (3.9), we obtain

$$\frac{2}{15} h_i A_i J_{2,i}^1 = -e^{-V_{min}/\overline{T}}\overline{g}_{2,i} + e^{-V_{min}/\overline{T}} g_{2,i_{min}}. \tag{3.15}$$

Using (3.14) and (3.15), we can eliminate $J_{2,i}^1$ and get

$$\overline{g}_{2,i}^h = (\overline{\gamma}\ \overline{c}_2 + 1)^{-1}(\overline{\gamma} r_i + g_{2,i_{min}}), \tag{3.16}$$

where $\overline{\gamma} = \frac{2}{15} h_i^2 A_i e^{V_{min}}$. Replacing (3.16) into (3.14) we get $J_{2,i}^1$ in terms of $g_{2,i}$:

$$J_{2,i}^1 = \frac{\overline{c}_2 h_i}{\overline{\gamma}\ \overline{c}_2 + 1} g_{2,i_{min}} - \frac{h_i}{\overline{\gamma}\ \overline{c}_2 + 1} r_i. \tag{3.17}$$

Finally, (3.11), with $q_i = 1$ and $q_k = 0$ for all $k \neq i$, gives

$$J^0_{2,i} + J^1_{2,i} P_i(x_i) = J^0_{2,i+1} + J^1_{2,i+1} P_{i+1}(x_i), \qquad i = 1, \ldots, N. \tag{3.18}$$

We recall that, due to definition (3.6)–(3.8), $P_i(x_i) = 0$ ($P_{i+1}(x_i) = 0$, respectively) if the minimum of V on I_i is in x_{i-1} (x_{i+1}, respectively); otherwise it is $P_i(x_i) = 1$ ($P_{i+1}(x_i) = -1$, respectively). Using the expression (3.13) for $J^0_{2,i}$ and (3.17) for $J^1_{2,i}$, the last equation (3.18) can be written in terms of the variables $g_{2,i}$ only, giving rise to a tridiagonal algebraic system, with the (positive) contribution of the zeroth order term appearing only in the diagonal entry. Then the energy flux J^h_2 is computed locally in each interval by (3.12) and $\overline{g}^h_2$ is computed locally by (3.16).

Discretizing the current continuity equation (2.23), (2.25) with the same scheme and applying the (simpler) static condensation procedure (J^h_1 is piecewise constant in this case), we obtain

$$J^0_{1,i} = J^0_{1,i+1}, \qquad i = 1, \ldots, N, \tag{3.19}$$

with

$$J^0_{1,i} = \frac{V_i - V_{i-1}}{2\overline{T}} \coth\left(\frac{V_i - V_{i-1}}{2\overline{T}}\right) \frac{g_{1,i} - g_{1,i-1}}{h_i} - \frac{g_{1,i} + g_{1,i-1}}{2\overline{T}} \frac{V_i - V_{i-1}}{h_i}. \tag{3.20}$$

Moreover, the analogue of (3.16) gives the upwind expression

$$\overline{g}_{1,i} = g_{1,i_{min}}, \qquad i = 1, \ldots, N. \tag{3.21}$$

In order to complete the scheme, we still have to specify how the piecewise constant temperature $\overline{T}$ is defined. The temperature is defined implicitly in terms of g_1 and g_2 according to the nonlinear equation (2.21). Lemma 2.1 shows that this equation can be solved uniquely. Numerically, we define $\overline{T}$ in each interval I_i as the approximate solution of

$$\frac{\overline{g}_{2,i}}{\overline{g}_{1,i}} = f(\overline{T}_i), \qquad i = 1, \ldots, N, \tag{3.22}$$

with $\overline{T}_i \overset{\text{def}}{=} \overline{T}|_{I_i}$ and $\overline{g}_{1,i}$, $\overline{g}_{2,i}$ given by the mixed scheme (see (3.21), (3.16)). The nonlinear equation reduces to a linear one when $\alpha = 0$ (parabolic band). For $\alpha > 0$ a single iteration of the (scalar) Newton scheme is sufficient to obtain $\overline{T}$ with an accuracy of 10^{-8}, when the initial guess is the temperature at the previous global iteration procedure step. Moreover, f' can be explicitly computed by (2.22).

For the discretization of the two-dimensional problem, one can freeze the temperature, defined in the global iteration procedure, and use the ideas of [24].

In contrast to the strongly coupled equations (2.1)–(2.4) in the variables μ/T and $-1/T$, the two continuity equations (2.23) and (2.24) are *weakly* coupled through the temperature (which varies only slowly during the iterations). Consequently, we defined the global iteration procedure as follows. The temperature is frozen at the previous iteration step, and a full Newton method is used to solve the nonlinear system in g_1, g_2, and V. At each iteration, the temperature is updated according to equation (3.22). The associated linear system is solved by using a GMRES solver. Finally, we remark that a Gummel-type iteration procedure can be employed (instead of the Newton method) in the parabolic band case.

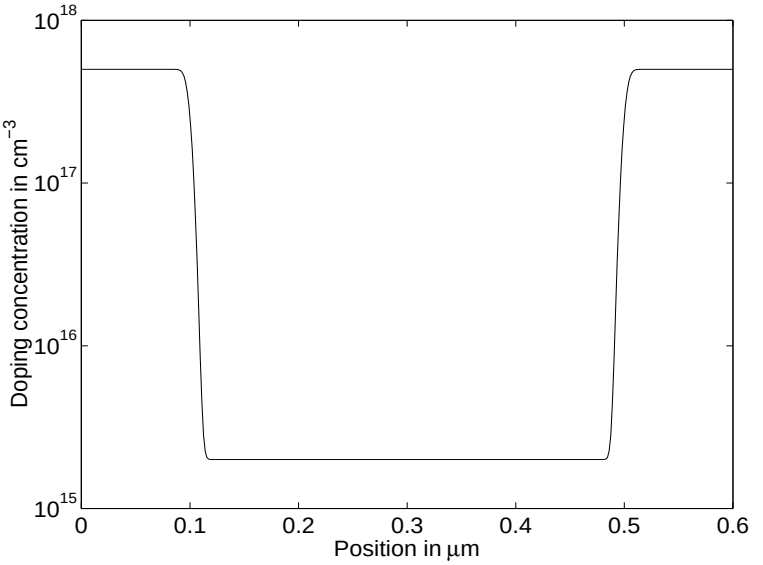


FIG. 4.1. *Doping concentration in the n^+nn^+ diode.*

4. Numerical results. As a numerical example we present the simulation of a one-dimensional n^+nn^+ ballistic silicon diode which is a simple model for the channel of a MOS transistor. The semiconductor domain is given by the interval $\Omega = (0, \ell^*)$ with $\ell^* > 0$. In the n^+-regions the maximal doping concentration is $5 \cdot 10^{17}\,\mathrm{cm}^{-3}$; in the n-channel the minimal doping profile is $2 \cdot 10^{15}\,\mathrm{cm}^{-3}$. The doping profile is shown in Figure 4.1. The length of the n^+-regions is $0.1\,\mu$m, whereas the length of the channel region equals $0.4\,\mu$m.

The numerical values of the physical parameters (for a silicon diode) are given in Table 4.1.

On the boundary points $x = 0$ and $x = \ell^*$ we assume that the (unscaled) total space charge $C - n$ vanishes and that the (unscaled) temperature has the ambient temperature

$$n(0) = n(\ell^*) = c_1, \qquad T(0) = T(\ell^*) = T_0, \qquad V(0) = 0, \quad V(\ell^*) = -U,$$

where $U > 0$ is the applied voltage. We take the value $U = 1.5\,\mathrm{V}$. The unscaled relaxation time τ_0 and the low-field mobility μ_0 depend on ϕ_0 and k_0 (see section 2.4). These parameters are chosen such that τ_0 and μ_0 take the values shown in Table 4.1. We have chosen the data such that our results can be compared to the numerical results of the literature (see, e.g., [9, 27, 34]).

We perform numerical results for a uniform mesh of 100 nodes. In Figure 4.2 we present the electron temperature for vanishing and nonvanishing nonparabolicity parameter α using Lyumkis's model. As expected the temperature in the n-channel is high; i.e., the electrons are "hot." The maximal temperature for $\alpha = 0$ is $T = 3970\,\mathrm{K}$ and $T = 3240\,\mathrm{K}$ for $\alpha = 0.5\,(\mathrm{eV})^{-1}$. The corresponding thermal energies are $E_{th} = \frac{3}{2}k_BT = 0.51\,\mathrm{eV}$ and $E_{th} = 0.42\,\mathrm{eV}$, respectively. Therefore, the temperature is reduced due to the nonparabolicity effects. Similar results can be observed by employing Chen's model (Figure 4.3). Here, the maximal temperature (thermal energy) values are $T = 2330\,\mathrm{K}$ ($E_{th} = 0.30\,\mathrm{eV}$) for $\alpha = 0$ and $T = 1610\,\mathrm{K}$

TABLE 4.1
Physical parameters.

Parameter	Physical meaning	Numerical value
q	elementary charge	$1.6 \cdot 10^{-19}\,\mathrm{As}$
ε_s	permittivity constant	$10^{-12}\,\mathrm{AsV^{-1}cm^{-1}}$
μ_0	(low field) mobility constant	$1.5 \cdot 10^3\,\mathrm{cm^2V^{-1}s^{-1}}$
U_T	thermal voltage at $T_0 = 300\,\mathrm{K}$	$0.026\,\mathrm{V}$
ℓ^*	length of the device	$0.6\,\mu\mathrm{m}$
ℓ_0	length of the n^+ region	$0.1\,\mu\mathrm{m}$
c_0	doping concentration in the n region	$2 \cdot 10^{15}\,\mathrm{cm^{-3}}$
c_1	doping concentration in the n^+ region	$5 \cdot 10^{17}\,\mathrm{cm^{-3}}$
τ_0	energy relaxation time	$0.4 \cdot 10^{-12}\,\mathrm{s}$
α	nonparabolicity parameter	$0.5\,(\mathrm{eV})^{-1}$

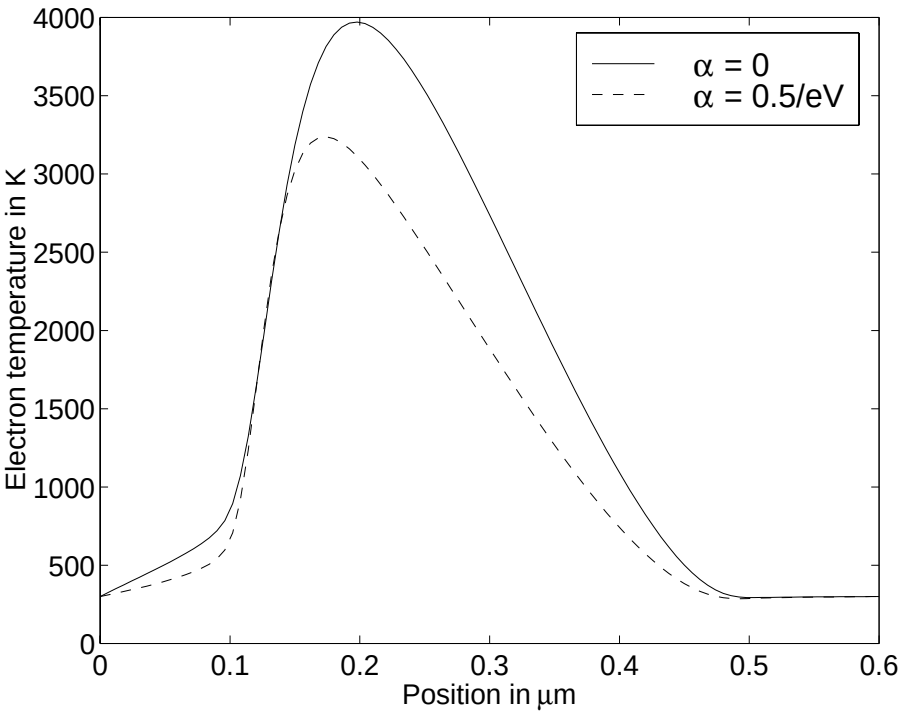


FIG. 4.2. *Electron temperature versus position in a ballistic diode using Lyumkis's model.*

($E_{th} = 0.21\,\mathrm{eV}$) for $\alpha = 0.5\,(\mathrm{eV})^{-1}$. The *effective* scaled relaxation time in the Lyumkis model is $(3\sqrt{\pi}/4)\tau_0\sqrt{T}$ and τ_0 in the Chen model. Therefore, the effective relaxation time in the Chen model is smaller than that in the Lyumkis model, and we expect that the maximal temperature in the Chen model is smaller than in the Lyumkis model. This observation follows from the fact that in the vanishing relaxation-time limit, the temperature relaxes to the lattice temperature, and it is confirmed by our numerical experiments.

In Figure 4.4 the electron mean velocity for the two different values of the nonparabolicity parameter α using Lyumkis's model is shown. The mean velocity u is

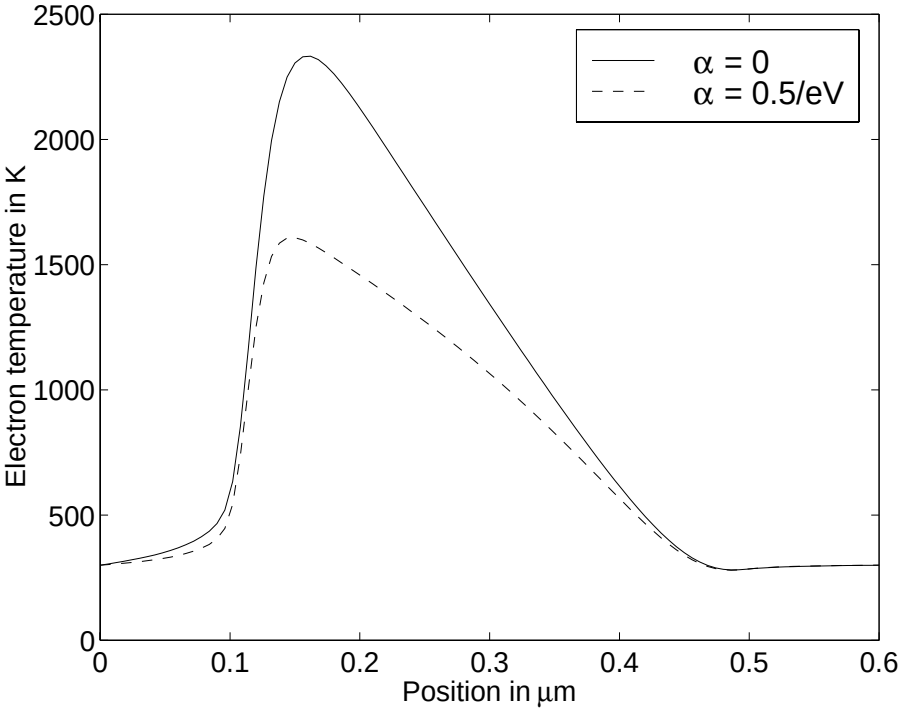


FIG. 4.3. *Electron temperature versus position in a ballistic diode using Chen's model.*

defined by $u = J_1/(qn)$. The spurious velocity overshoot peak at the left junction becomes smaller for nonvanishing nonparabolicity parameter. The maximal mean velocity for $\alpha = 0$ is $u = 2.92 \cdot 10^7$ cm/s and $u = 1.51 \cdot 10^7$ cm/s for $\alpha = 0.5\,(\mathrm{eV})^{-1}$. The same effect can be observed using Chen's model (see Figure 4.5), where the spurious velocity overshoot spike almost vanishes for $\alpha = 0.5\,(\mathrm{eV})^{-1}$. The maximal velocities are $u = 1.44 \cdot 10^7$ cm/s for $\alpha = 0$ and $u = 1.25 \cdot 10^7$ cm/s for $\alpha = 0.5\,(\mathrm{eV})^{-1}$.

The mean velocities for the nonparabolic case, using Chen's or Lyumkis's models, are compared to the mean velocity from the standard drift-diffusion model in Figure 4.6. In the latter model, no velocity saturation effects are taken into account, i.e., the drift-diffusion model equals the energy-transport equations in the case of *constant* temperature. The velocity overshoot from the drift-diffusion model is much larger than for the energy-transport equations (Figure 4.6). This can be explained by the fact that the total energy of the energy-transport model is composed of the thermal *and* the kinetic energy, whereas the total energy of the drift-diffusion model is determined only by the kinetic energy.

In Figure 4.7 we present the current-voltage characteristics for the different energy-transport models. The particle current density J_1 is always smaller in nonparabolic band situations. Its dependence on the applied voltage U seems to be sublinear. Indeed, in the voltage range $U \in [0.5\mathrm{V}, 1.5\mathrm{V}]$, the dependence of J_1 on U is approximately $J_1 \sim U^\gamma$, where γ is between 0.88 and 1, depending on the model (see Table 4.2). We remark that increasing the number of nodes does not change the values of the current.

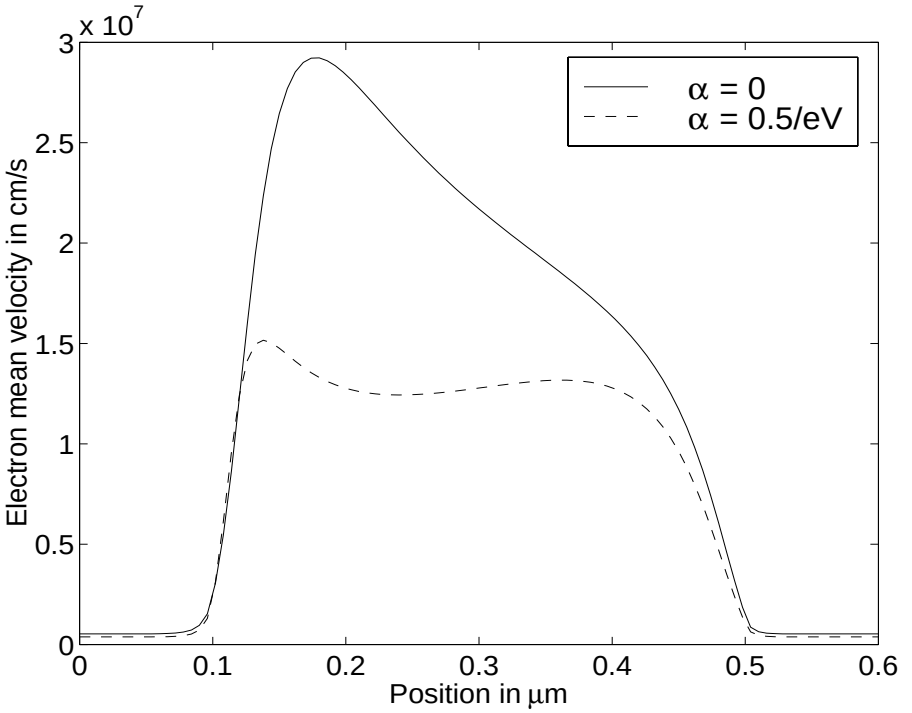


FIG. 4.4. *Electron mean velocity versus position in a ballistic diode using Lyumkis's model.*

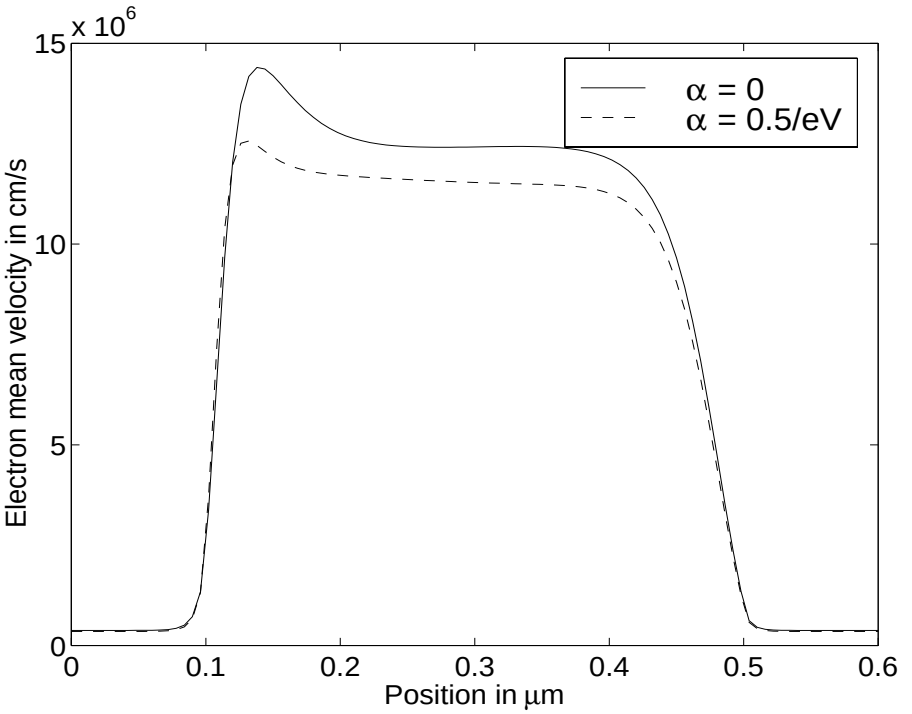


FIG. 4.5. *Electron mean velocity versus position in a ballistic diode using Chen's model.*

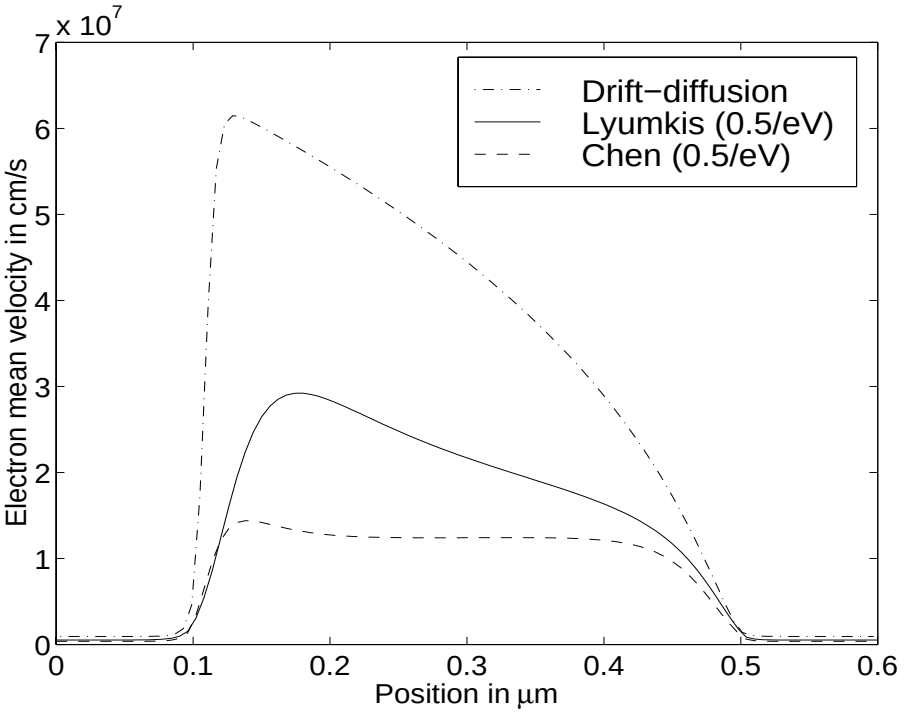


FIG. 4.6. *Electron mean velocity versus position in a ballistic diode.*

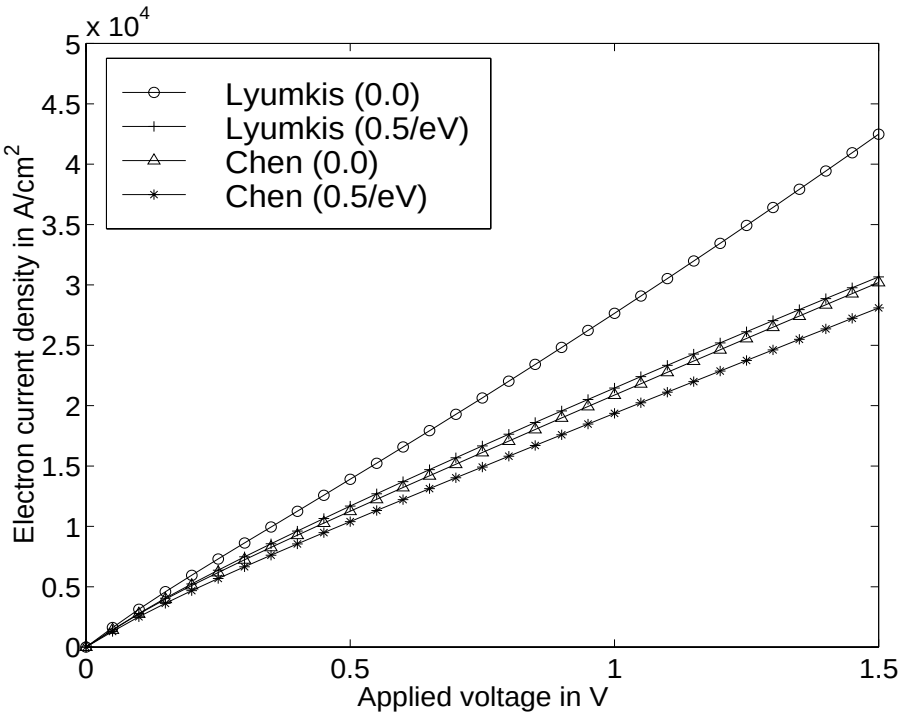


FIG. 4.7. *Current-voltage characteristics for a ballistic diode.*

TABLE 4.2
Slopes of the logarithmic current-voltage curves for $U \in [0.5\,V, 1.5\,V]$.

Model	slope
Lyumkis: $\alpha = 0.0$	1.00
Lyumkis: $\alpha = 0.5/\text{eV}$	0.88
Chen: $\alpha = 0.0$	0.90
Chen: $\alpha = 0.5/\text{eV}$	0.90

TABLE 4.3
RE for the variables.

N	RE for ψ_h	RE for T_h	RE for n_h	RE for $g_{2,h}$
50	$7.23 \cdot 10^{-2}$	$8.63 \cdot 10^{-2}$	$2.37 \cdot 10^{-2}$	$3.13 \cdot 10^{-2}$
100	$3.47 \cdot 10^{-2}$	$4.16 \cdot 10^{-2}$	$6.52 \cdot 10^{-3}$	$9.72 \cdot 10^{-3}$
200	$1.69 \cdot 10^{-2}$	$1.89 \cdot 10^{-2}$	$2.26 \cdot 10^{-3}$	$3.76 \cdot 10^{-3}$
400	$8.0 \cdot 10^{-3}$	$7.56 \cdot 10^{-3}$	$8.08 \cdot 10^{-4}$	$1.45 \cdot 10^{-3}$

In Table 4.3 we report on the relative errors for the computed variables using the Chen model in the parabolic band situation. As a reference solution we choose the computed solution with a mesh of $N+1 = 1201$ nodes. The relative error (RE) for ψ_h is defined by $\|\psi_h - \psi\|_{H^1}/\|\psi\|_{H^1}$, where ψ_h is the discrete solution with discretization parameter $h = 1/N$. From Table 4.3 it can be seen immediately that the order of convergence for the relative error for ψ_h is one, which is expected. The relative errors for T_h, n_h, $g_{2,h}$ are defined as above but in the L^2-norm. Since $n = g_1$ in the Chen model, we do not need to report on the error for g_1. The orders of convergence for T_h, n_h, and $g_{2,h}$ are 1.23, 1.62, and 1.48, respectively. Similar results are obtained by employing the Lyumkis model and nonparabolic band situations, except that the order of convergence for n_h is smaller due to the nonlinear dependence of n_h on T_h.

5. Conclusions. In this paper we have derived energy-transport models for semiconductors for general nonparabolic band diagrams. The diffusion coefficients and the energy relaxation term can be written analytically in terms of the electron density and the temperature if nonparabolic bands in the sense of Kane are considered. The energy-transport models are completely derived from the semiconductor Boltzmann equation. There appear two parameters: the nonparabolicity parameter α and the parameter in the definition of the momentum relaxation time β. For parabolic bands ($\alpha = 0$), we recover two models already studied in the literature: the so-called Lyumkis model ($\beta = 0$) [22] and the so-called Chen model ($\beta = 1/2$) [9].

Thanks to a drift-diffusion formulation valid for a large class of energy-transport models, we presented a mixed exponential fitting finite element discretization of the stationary equations and numerical experiments of a ballistic diode in one space dimension. It turns out that the spurious velocity overshoot peak is smaller in Chen's model than in Lyumkis' model and for nonparabolic bands compared to parabolic ones. Furthermore, the spurious peak almost vanishes in the nonparabolic Chen model. This shows that the energy-transport models describe the charge flow of electrons in a ballistic diode with reasonable accuracy.

REFERENCES

[1] W. Allegretto and H. Xie, *Nonisothermal semiconductor systems*, in Comparison Methods and Stability Theory, Lecture Notes in Pure and Appl. Math. 162, X. Liu and D. Siegel, eds., Marcel Dekker, New York, 1994.

[2] Y. Apanovich, P. Blakey, R. Cottle, E. Lyumkis, B. Polsky, A. Shur, and A. Tcherniaev, *Numerical simulations of submicrometer devices including coupled nonlocal transport and nonisothermal effects*, IEEE Trans. Electron Devices, 42 (1995), pp. 890–897.

[3] D. N. Arnold and F. Brezzi, *Mixed and nonconforming finite element methods: Implementation, postprocessing and error estimates*, RAIRO Model Math. Anal. Numer., 19 (1985), pp. 7–32.

[4] N. Ben Abdallah, P. Degond, P. Markowich, and C. Schmeiser, *High field approximations of the spherical harmonics expansion model for semiconductors*, Z. Angew. Math. Phys., to appear.

[5] N. Ben Abdallah and P. Degond, *On a hierarchy of macroscopic models for semiconductors*, J. Math. Phys., 37 (1996), pp. 3308–3333.

[6] F. Brezzi, L. Marini, and P. Pietra, *Méthodes d'éléments finis mixtes et schéma de Scharfetter-Gummel*, C. R. Acad. Sci. Paris Ser. I Math., 305 (1987), pp. 599–604.

[7] F. Brezzi, L. D. Marini, and P. Pietra, *Numerical simulation of semiconductor devices*, Comput. Methods Appl. Mech. Engrg., 75 (1989), pp. 493–514.

[8] F. Brezzi, L. D. Marini, and P. Pietra, *Two-dimensional exponential fitting and applications to drift-diffusion models D*, SIAM J. Numer. Anal., 26 (1989), pp. 1342–1355.

[9] D. Chen, E. Kan, U. Ravaioli, C. Shu, and R. Dutton, *An improved energy transport model including nonparabolicity and non-Maxwellian distribution effects*, IEEE Electron Device Letters, 13 (1992), pp. 26–28.

[10] D. Chen, E. Sangiorgi, M. Pinto, E. Kan, U. Ravaioli, and R. Dutton, *Analysis of spurious velocity overshoot in hydrodynamic simulations*, NUPAD IV, 1992, pp. 109–114.

[11] P. Degond, S. Génieys, and A. Jüngel, *An existence and uniqueness result for the stationary energy-transport model in semiconductor theory*, C. R. Acad. Sci. Paris Ser. I Math., 324 (1997), pp. 29–34.

[12] P. Degond, S. Génieys, and A. Jüngel, *A system of parabolic equations in nonequilibrium thermodynamics including thermal and electrical effects*, J. Math. Pures Appl., 76 (1997), pp. 991–1015.

[13] P. Degond, S. Génieys, and A. Jüngel, *A steady-state system in nonequilibrium thermodynamics including thermal and electrical effects*, Math. Methods Appl. Sci., 21 (1998), pp. 1399-1413.

[14] A. Gnudi, F. Odeh, and M. Rudan, *Investigation of non-local transport phenomena in small semiconductor devices*, Europ. Trans. Telecommun., 1 (1990), p. 307.

[15] J. Jerome, *Analysis of Charge Transport. A Mathematical Study of Semiconductor Devices*, Springer, Berlin, 1996.

[16] J. Jerome and C.-W. Shu, *Energy transport systems for semiconductors: Analysis and simulation*, First World Congress of Nonlinear Analysts 192, Walter de Gruyter, Berlin, 1995, pp. 3835–3846.

[17] A. Jüngel, *The energy-transport model for semiconductors: Some analytical and numerical results*, in Proceedings of the International Workshop on "Recent Progress in the Mathematical Theory on Vlasov-Maxwell Equations," Paris, 1997, pp. 140–158.

[18] A. Jüngel, *Regularity and uniqueness of solutions to a system of parabolic equations in nonequilibrium thermodynamics*, Nonlinear Anal., to appear.

[19] A. Jüngel and P. Pietra, *A discretization scheme of a quasi-hydrodynamic semiconductor model*, Math. Modél. Methods Appl. Sci., 7 (1997), pp. 935–955.

[20] E. Kane, *Band structure of indium-antimonide*, J. Phys. Chem. Solids, 1 (1957), pp. 249–261.

[21] T. Kerkhoven and Y. Saad, *On acceleration methods for coupled nonlinear elliptic systems*, Numer. Math., 60 (1992), pp. 525–548.

[22] E. Lyumkis, B. Polsky, A. Shur, and P. Visocky, *Transient semiconductor device simulation including energy balance equation*, COMPEL, 11 (1992), pp. 311–325.

[23] L.D. Marini and P. Pietra, *An abstract theory for mixed approximations of second order elliptic problems*, Mat. Apl. Comput., 8 (1989), pp. 219–239.

[24] L.D. Marini and P. Pietra, *New mixed finite element schemes for current continuity equations*, COMPEL, 9 (1990), pp. 257–268.

[25] P. A. Markowich, *The Stationary Semiconductor Device Equations*, Springer, Vienna, 1986.

[26] P. A. Markowich, C. A. Ringhofer, and C. Schmeiser, *Semiconductor Equations*, Springer, Berlin, 1990.

[27] A. Marrocco, P. Montarnal, and B. Perthame, *Simulation of the energy-transport and simplified hydrodynamic models for semiconductor devices using mixed finite elements*, in Proceedings ECCOMAS 96, John Wiley, London, 1996.

[28] F. Odeh, M. Rudan, and J. White, *Numerical solution of the hydrodynamic model for a one-dimensional semiconductor device*, COMPEL, 6 (1987), pp. 151–170.

[29] M. Ruadan and F. Odeh, *Multi-dimensional discretization scheme for the hydrodynamic model of semiconductor devices*, COMPEL, 5 (1986), pp. 149–183.

[30] P. Raviart and J. Thomas, *A mixed finite element method for second order elliptic equations*, in Mathematical Aspects of the Finite Element Method, Lecture Notes in Math. 606, Springer, Berlin, 1977, pp. 292–315.

[31] D. Scharfetter and H. Gummel, *Large signal analysis of a Silicon Read diode oscillator*, IEEE Trans. Electron Devices, ED16 (1969), pp. 64–77.

[32] C. Schmeiser and A. Zwirchmayr, *Elastic and drift-diffusion limits of electron-phonon interaction in semiconductors*, Math. Modél. Methods Appl. Sci., 8 (1998), pp. 37-53.

[33] K. Souissi, F. Odeh, and A. Gnudi, *A note on current discretization in the hydromodel*, COMPEL, 10 (1991), pp. 475–485.

[34] K. Souissi, F. Odeh, H. Tang, and A. Gnudi, *Comparative studies of hydrodynamic and energy transport models*, COMPEL, 13 (1994), pp. 439–453.

[35] P. Visocky, *A method for transient semiconductor device simulation using hot-electron transport equations*, Proceedings of the Nasecode X Conference., J. Miller, ed., Boole Press, Dublin, 1994, pp. 101–104.

[36] D. Woolard, M. Stroscio, M. Littlejohn, R. Trew, and H. Grubin, *A new nonparabolic hydrodynamic model with quantum corrections*, in Computational Electronics: Semiconductor Transport and Device Simulation, K. Hess, ed., Kluwer Academic Press, Boston, 1991, pp. 59–62.

SIAM J. SCI. COMPUT.
Vol. 22, No. 3, pp. 1008–1015
© 2000 Society for Industrial and Applied Mathematics

REMARKS ON HIGH-RESOLUTION SPLIT SCHEMES COMPUTATION*

M. BEN-ARTZI†, J. FALCOVITZ†, AND U. FELDMAN†

Abstract. The high-resolution generalized Riemann problem (GRP) conservation laws scheme for compressible flows combined with Strang-type operator splitting is applied to computing an initial value problem having a discontinuous initial data. Imperfect representation of the initial data on the Cartesian grid, where the smooth curve of discontinuity is approximated by a jagged line, gives rise to spurious waves when using high-resolution integration with operator splitting. The nature of these waves is clarified by comparison to a one-dimensional model. We demonstrate that it is not the operator splitting that gives rise to these waves, but rather the better quality of the hyperbolic (one-dimensional) solver, which is not degraded by the operator splitting. It is expected that this property of retaining sharp features of initial data will also be produced by other second-order conservation laws schemes.

AMS subject classifications. 65M06, 76M20

Key words. compressible flow, shock waves, high-resolution computation, GRP method, Godunov-type scheme, irregular cells

PII. S1064827599345248

1. Introduction.

Consider the numerical solution of compressible time-dependent flow in two space dimensions, where the discontinuous initial data consists of two uniform states, U_L, U_R, separated by a curve $\mathcal{L}$. The fluid is initially at rest; i.e., all velocities vanish. The solutions are computed via the generalized Riemann problem (GRP) conservation laws scheme [1, 2, 3], using a Cartesian grid of square cells. When $\mathcal{L}$ intersects grid cells, the initial data specified in those cells cannot conform exactly to the given initial data. Thus we used three methods of initialization to clarify the effect of these alternates on the numerical solution. It was found that computation results were quite sensitive to the particular procedure by which the initial cell data was adjusted to approximately represent the exact initial data. By comparing the 2-D solutions with equivalent quasi-1-D solutions, using high-resolution and low-resolution methods, we conclude that the spurious waves result from the high-resolution property of the method that is not degraded by two-dimensional (2-D) operator splitting. Indeed, in the low-resolution one-dimensional (1-D) computation those waves were smeared out.

The purpose of this note is to present our findings in this regard and to suggest that caution be exercised when computing multidimensional flows with discontinuous initial data. It is noted in passing that the problem of averaging discontinuous data in cells is encountered also in multimaterial Eulerian schemes, where uniform velocity and pressure are maintained in mixed cells, following finite-difference integration of the conservation laws.

In section 2 we discuss the three alternate initialization methods. In section 3 we give a brief account of the governing equations, the GRP conservation laws

*Received by the editors July 7, 1999; accepted for publication (in revised form) March 23, 2000; published electronically September 15, 2000. This research was supported by the German-Israeli Foundation for Scientific Research and Development GIF grant I-318-195.06/93.

http://www.siam.org/journals/sisc/22-3/34524.html

†Institute of Mathematics, The Hebrew University, Jerusalem 91904, Israel (mbartzi@math.huji.ac.il, ccjf@math.huji.ac.il, ccuf@math.huji.ac.il). The first author was partially supported by the Israel Science Foundation.

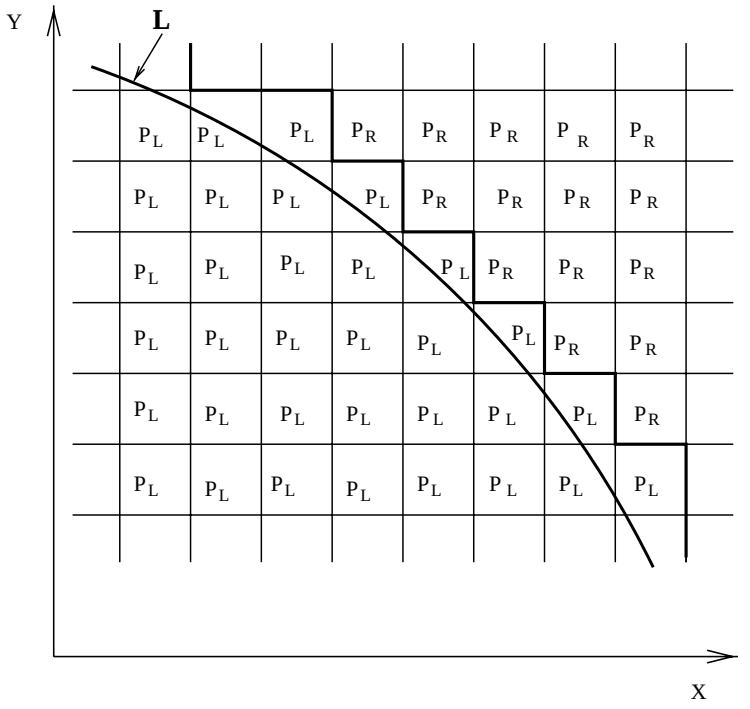


FIG. 1. *Discontinuous initial data. Line L approximated in a stepwise manner.*

scheme, and the "splitting strategy." Section 4 describes the computation results for an exploding high-pressure cylinder, demonstrating the previously outlined findings.

2. Initialization of discontinuous data. In this study, the data in cells intersected by $\mathcal{L}$ were initialized as follows. The density was always obtained by a "conservative averaging" of the exact discontinuous data (i.e., area-weighted average of the two densities ρ_L, ρ_R per cell).

The pressure, however, was obtained by one of the following approximation procedures:

(A) Initial cell pressure is set to either one of the two pressures p_L, p_R. This amounts to a "pressure-jump-preserving" stepwise approximation of $\mathcal{L}$ (as illustrated in Figure 1), where $\mathcal{L}$ is approximated by a "jagged" curve, thereby introducing "hot spots" in cells having high pressure and low density.

(B) Cell pressure is obtained by a "conservative averaging" procedure identical to the density initialization outlined above (see the schematic illustration in Figure 2). This procedure is consistent with the conservation laws (the pressure for a perfect gas being proportional to the energy per unit volume), at the sacrifice of spreading the pressure jump over one or two cells.

(C) Initial cell pressure in intersected cell "A," where the density is ρ_A, is set to be equal to p_A, so that (ρ_L, p_L) and (ρ_A, p_A) lie on the same isentropic curve in the (ρ, p) plane. We take (ρ_L, p_L) to be the "high-pressure state"; i.e., $p_L > p_R$.

3. Outline of the numerical method. Assuming an inviscid compressible fluid and an ideal gas equation of state, the 2-D flow is governed by the laws for

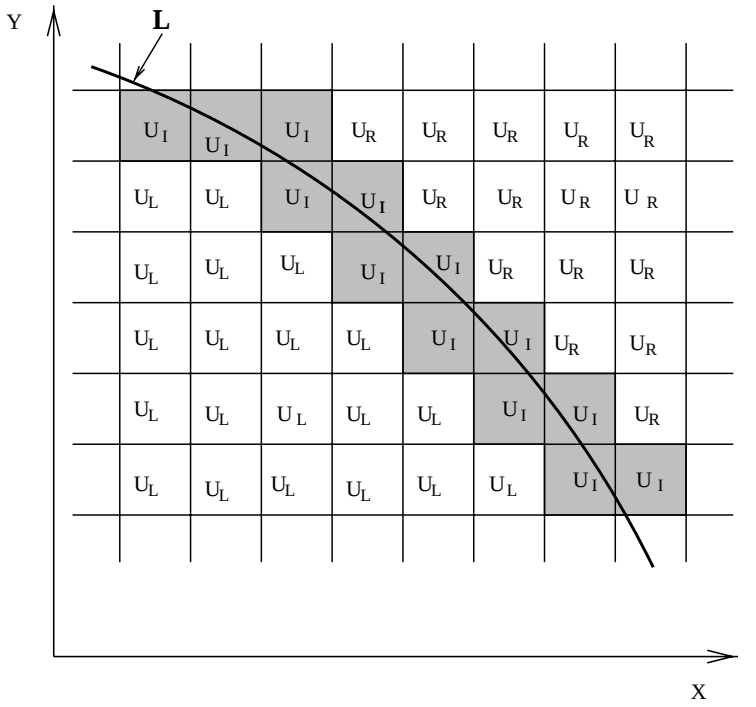


FIG. 2. *Initial data approximated by intermediate state* U_I.

conservation of mass, momentum, and energy, expressed in Cartesian coordinates (x, y) as

$$\partial_t U + \partial_x F(U) + \partial_y G(U) = 0,$$

$$U(x,y,t) = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho E \end{bmatrix}, \quad F(U) = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ u(\rho E + p) \end{bmatrix}, \quad G(U) = \begin{bmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ v(\rho E + p) \end{bmatrix}, \tag{1}$$

$$p = (\gamma - 1)\rho e, \quad \gamma = \text{constant } > 1, \tag{2}$$

$$e = E - \frac{1}{2}(u^2 + v^2). \tag{3}$$

In (1) we denote by ρ, p, e, E, u, v the density, pressure, specific-energy, specific total energy, and (x, y)-velocity components, respectively.

The 2-D finite-difference approximation to (1) is formulated as a "Strang-type" operator splitting [6], using the GRP scheme [1, 2, 3] as the 1-D finite-difference operator. Our point is that this basic algorithm gives a very accurate result, even under the difficult conditions where the initial data is discontinuous at a curved surface cutting obliquely across the Cartesian grid, as will be demonstrated by comparison to an equivalent 1-D solution. The splitting procedure can be outlined as follows.

The system (1) is split into the two simpler systems,

$$\partial_t U + \partial_x F(U) = 0, \tag{4i}$$

$$\partial_t U + \partial_y G(U) = 0. \tag{4ii}$$

Loosely speaking, the system (4) is taken to mean that the evolution of an initial state U_o by (1) over a short time interval Δt can be approximated by evolving U_o first subject to (4i) (over time Δt) obtaining a state U_1, then evolving U_1 in accordance with (4ii) again over time Δt.

Let $L_x(\Delta t)$, $L_y(\Delta t)$, $L(\Delta t)$ denote finite-difference approximation operators for the integration by a time-step Δt of (4i), (4ii), (1), respectively. Then the "Strang-type" operator sequence

$$L(\Delta t) = L_x\left(\frac{1}{2}\Delta t\right) L_y(\Delta t) L_x\left(\frac{1}{2}\Delta t\right) \tag{5}$$

is a second-order finite-difference approximation to (1).

The 1-D operators $L_x(\Delta t), L_y(\Delta t)$ are given by the "GRP solvers" [1, 2, 3]. The basic idea (in terms of L_x) is the following. The grid consists of the sequence of points $x_{i+1/2} = (i+1/2)\Delta x$, $i = 0, 1, 2, \ldots, i_{\max}$, where Δx is the grid spacing and the cell i is the interval $x_{i-1/2} < x < x_{i+1/2}$. $U(x, y, t)$ (for a fixed y) is approximated at time $t = t_n = n\Delta t$ by $U^n(x, y)$, a piecewise linear distribution in cells, having the average value U_i^n in cell i. The finite-difference GRP solver L_x, yielding $\{U_i^{n+1}\}_{i=1}^{i_{\max}}$ in terms of $\{U_i^n\}_{i=1}^{i_{\max}}$ is explicitly given by

$$U_i^{n+1} = U_i^n - \frac{\Delta t}{\Delta x}\left[F(U)_{i+1/2}^{n+1/2} - F(U)_{i-1/2}^{n+1/2}\right], \tag{6}$$

where the time-centered fluxes $F(U)_{i+1/2}^{n+1/2}$ are determined analytically from solutions to GRPs that arise at the cell interfaces $x_{i+1/2}$. More specifically, these solutions are obtained as exact (up to second-order) solutions to (4i) when the initial data is given by the piecewise linear distribution $U^n(x, y)$. We refer the reader to [1, 2, 3] for a comprehensive account of the GRP analysis and the resulting scheme, and to [4] for an overview of GRP methods and their diverse applications.

4. Numerical examples. We consider a case where the initial value problem could be reduced to a flow problem with cylindrical symmetry (at least up to a certain time). Thus the solution could be obtained by an accurate (quasi-) 1-D GRP code.

Turning to the sample problem, we set the data as follows. The fluid is an ideal gas having $\gamma = 1.4$ and it is initially at rest everywhere. The curve $\mathcal{L}$ is the circle centered at $(x, y) = (0, 0)$, having radius $R = 50$. It encircles the high-pressure state $(\rho_L, p_L) = (10, 20)$, while the low-pressure state outside $\mathcal{L}$ is $(\rho_R, p_R) = (1, 1)$. The computational domain is the square $(0 < x < 100, 0 < y < 100)$, which is divided into a grid of 100×100 square cells. The integration was carried out with a constant time step $\Delta t = 0.16$, up to the final time of $T = 12.5$. Compliance with the Courant–Friedrichs–Lewy stability condition [5] throughout the computation was verified.

As a reference calculation, we used the quasi-1-D GRP scheme (with the same grid spacing of $\Delta r = 1$) to compute the flow profiles, as functions of the radial coordinate r. The pressure and density profiles are shown in Figure 3. They represent a sharp outgoing shock wave.

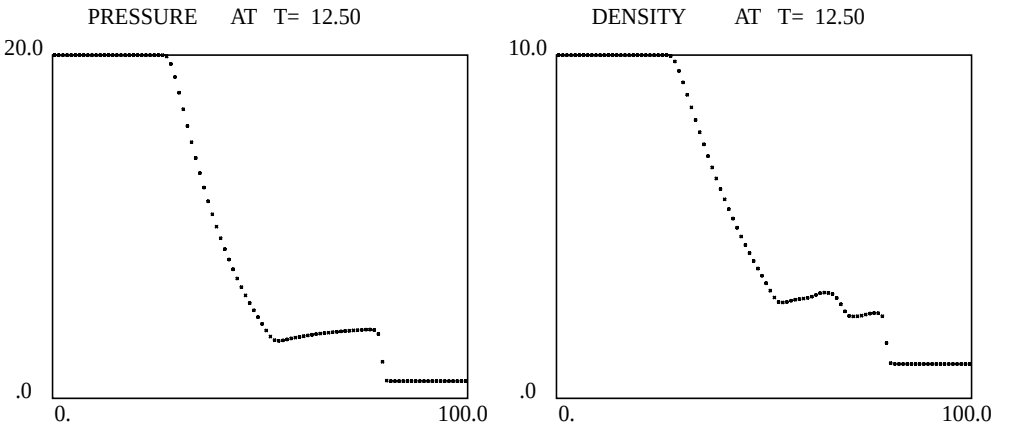


FIG. 3. *One-dimensional computation results.*

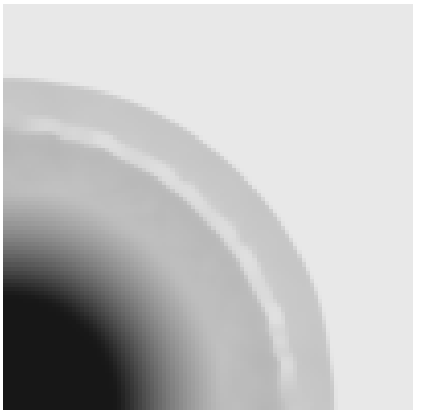


FIG. 4. *Density—Gray scale plot on* (x, y) *plane. Initial conditions type* (A).

Taking the initial data as in case (A) of section 2 (corresponding to Figure 1), and using the 2-D code as outlined above, we obtain the density distribution shown in Figure 4. We show only the density distribution, since the pressure is continuous at the contact discontinuity (see Figure 3) and does not show the spurious wave. The amount of gray shading is proportional to the density. In Figure 5 we display the cross section of the same profile along the diagonal. It is noted that all 1-D and 2-D plots are drawn to the same scale. The diagonal profile, having a mesh size larger by a factor of $\sqrt{2}$, resolves the same flow region by proportionately fewer cells.

On the other hand, by taking the initial data as in cases (B) or (C) of section 2, the results were in excellent agreement with the quasi-1-D computation (Figure 3). In Figure 6 we show the density distribution, using the initial data as in case (C) (i.e., in boundary cells the values (ρ, p) lie on the isentropic curve through (ρ_L, p_L)). Figure 7 shows the cross section of that profile along the diagonal.

The discrepancy between the results shown in Figures 4–7 can be explained as follows. In any boundary cell (namely, a cell intersected by the initial contact discontinuity), the value of the density is determined uniquely by mass conservation. Thus the only "degree of freedom" is expressed in the initial pressure value. Let $\bar{\rho}$ be the

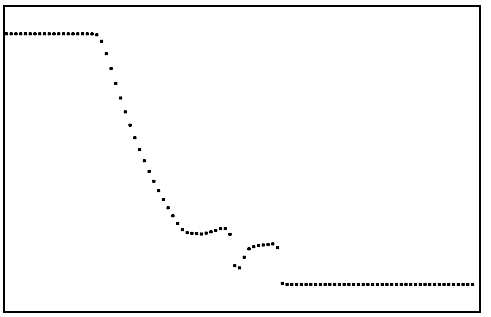


FIG. 5. *Density—Diagonal cross section. Initial conditions type* (A).

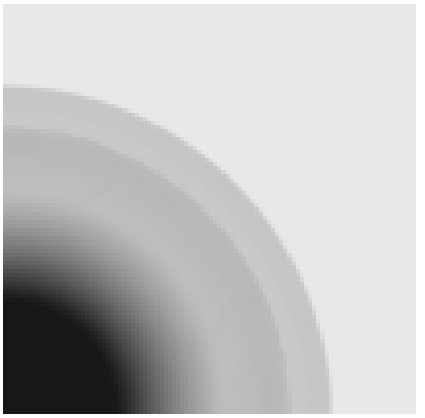


FIG. 6. *Density—Gray scale plot on* (x, y) *plane. Initial conditions type* (C).

density in that cell, and let $p_L, \bar{p}$ be the values determined by the strategies (A), (C), respectively. Clearly $p_L > \bar{p}$. Thus in case (A) the boundary cell constitutes a "hot spot," having entropy (hence temperature) higher than its neighboring cells. This "high-temperature" shell (along the contact discontinuity) is then propagated by the flow, as shown in Figures 4 and 5.

To support this explanation, we show in Figure 8 the results of a quasi-1-D calculation, using the same initial data as that used in Figure 3, except for one change, as follows. In the cell immediately to the right of the initial jump, where formerly (Figure 3) p_L, ρ_L were set, we "planted" density $\bar{\rho} = \frac{1}{2}(\rho_L + \rho_R)$ and $\bar{p} = p_L$, thus generating again a "hot spot." The results in Figure 8 are now in agreement with those of Figure 5. In particular, we stress that the split algorithm did not smooth out this spurious wave in the second-order computation.

We further underline this point by repeating the 2-D computation using the (first-order accurate) Godunov method, with type (A) initial conditions. These are the initial conditions that produced the "dip" in the density in the second-order accurate computation (see Figure 5). The results are shown as a density distribution along the diagonal in Figure 9. By comparing Figure 9 to Figure 5, it is evident that the first-order scheme does not retain the sharp features in the initial data as well as the second-order scheme, since the "dip" in the density distribution has disappeared. This should not be taken to mean that the first-order (Godunov) scheme is superior to the second-order (GRP) scheme. Rather, it demonstrates that due to its low

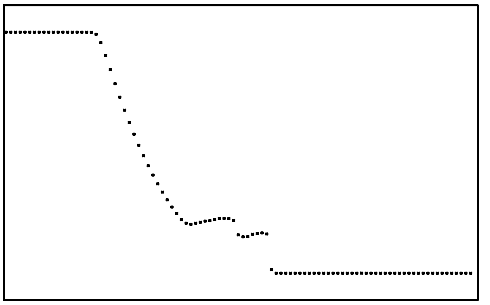


FIG. 7. *Density—Diagonal cross section. Initial conditions type* (C).

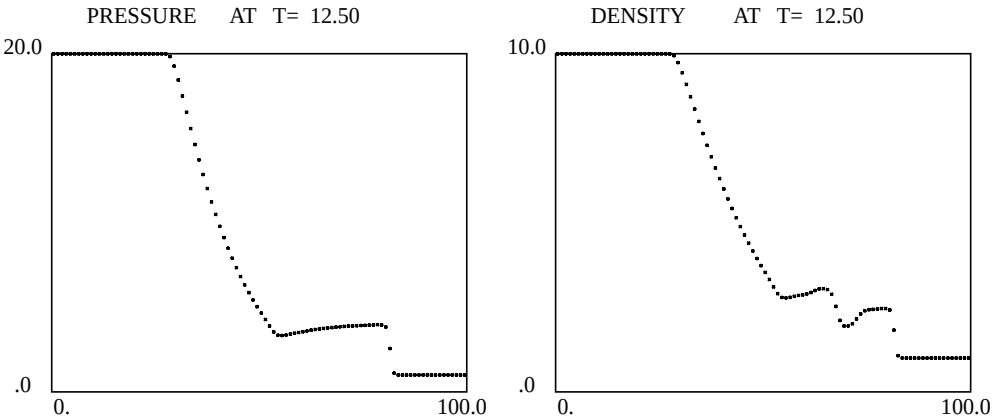


FIG. 8. *One-dimensional computation with data similar to type* (A).

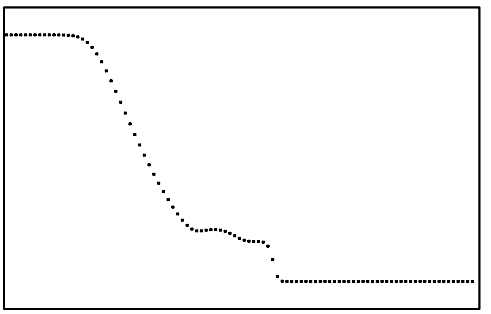


FIG. 9. *First-order density diagonal cross section. Initial conditions type* (A).

resolution quality, the Godunov scheme smears out the narrow "hot spot" that had been deliberately introduced in the initial data.

We may therefore summarize this study by pointing out the significance of approximating discontinuous initial data. In cases where geometric restrictions make it impossible to retain the exact initial data, some singular features may accompany the contact discontinuity curve in the ensuing flow. In particular, we note that these features are not smoothed out by the operator-splitting strategy associated with the high-resolution GRP scheme. It is expected that this property of retaining sharp fea-

tures of initial data will not be restricted to our GRP scheme. Other high-resolution schemes of the extended Godunov type may well exhibit similar characteristics.

REFERENCES

[1] M. Ben-Artzi and J. Falcovitz, *A second-order Godunov-type scheme for compressible fluid dynamics*, J. Comput. Phys., 55 (1984), pp. 1–32.

[2] M. Ben-Artzi and J. Falcovitz, *A high-resolution upwind scheme for quasi* 1*-D flows*, in Numerical Methods for the Euler Equations of Fluid Dynamics, F. Angrand, A. Dervieux, J.A. Desideri, and R. Glowinski, eds., SIAM, Philadelphia, 1985, pp. 66–83.

[3] M. Ben-Artzi and J. Falcovitz, *An upwind second-order scheme for compressible duct flows*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 744–768.

[4] J. Falcovitz and M. Ben-Artzi, *Recent developments of the GRP method*, JSME Internat. J. Ser. B, 38 (1995), pp. 497–517.

[5] R.D. Richtmyer and K.W. Morton *Difference Methods for Initial Value Problems*, Interscience, New York, 1967.

[6] G. Strang, *On the construction and comparison of difference schemes*, SIAM J. Numer. Anal., 5 (1968), pp. 506–517.

SIAM J. SCI. COMPUT.
Vol. 22, No. 3, pp. 1016–1035
© 2000 Society for Industrial and Applied Mathematics

A TIME-REVERSIBLE, REGULARIZED, SWITCHING INTEGRATOR FOR THE N-BODY PROBLEM*

ANNE KVÆRNØ† AND BEN LEIMKUHLER‡

Abstract. This article describes a gravitational N-body integration algorithm conserving linear and angular momentum and time-reversal symmetry. Forces are dynamically partitioned based on interbody separation, so that the long-range forces are evaluated relatively rarely, and close approaches are treated by an efficient regularization technique. The method incorporates an automatic stepsize adjustment based on a Sundman time-transformation. Although the scheme is formally second order, the most intensive computations (the close-approach dynamics) are executed at higher order, thus improving the overall accuracy. Numerical experiments indicate that the method can effectively treat few-body gravitational problems with two-body close approaches, and it compares favorably with other schemes presented in the literature.

Key words. N-body problems, Hamiltonian systems, time-reversible discretization, smooth switching functions

AMS subject classification. 65L05

PII. S1064827599355566

1. Introduction.

The N-body problem of celestial mechanics is described by a Hamiltonian,

$$\mathscr{H}(\boldsymbol{p},\boldsymbol{q}) = T(\boldsymbol{p}) + f(\boldsymbol{q}),$$

where $T : \mathbb{R}^{3N} \to \mathbb{R}$ and $f : \mathbb{R}^{3N} \to \mathbb{R}$ are smooth kinetic and potential energy functions defined in terms of the individual momenta $\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \boldsymbol{p}_N \in \mathbb{R}^3$ and positions $\boldsymbol{q}_1, \boldsymbol{q}_2, \ldots, \boldsymbol{q}_N \in \mathbb{R}^3$ of the bodies by

$$T(\boldsymbol{p}) = T(\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \boldsymbol{p}_N) := \sum_{i=1}^{N} \frac{|\boldsymbol{p}_i|^2}{2m_i},$$

and

$$f(\boldsymbol{q}) = f(\boldsymbol{q}_1, \boldsymbol{q}_2, \ldots, \boldsymbol{q}_N) := -\sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \frac{Gm_i m_j}{r_{ij}}, \qquad r_{ij} = |\boldsymbol{q}_i - \boldsymbol{q}_j|,$$

where G is the gravitational constant and m_i represents the mass of the ith body.

The N-body problem is the seminal problem of dynamical systems, and study of its solutions, ergodic properties, and topological structure continues to generate a

*Received by the editors May 4, 1999; accepted for publication (in revised form) March 29, 2000; published electronically September 15, 2000.
http://www.siam.org/journals/sisc/22-3/35556.html

†Department of Mathematical Sciences, Norwegian University of Science and Technology, N-7491 Trondheim, Norway (Anne.Kvarno@math.ntnu.no). This author worked on this project during a sabbatical visit at the University of Kansas and was supported by the Norwegian Research Council through the SYNODE-II project, contract 127582/410.

‡Department of Mathematics and Computer Science, University of Leicester, Leicester LE1 7RH, United Kingdom (B.Leimkuhler@mcs.le.ac.uk). Part of this work was performed at the Mathematical Sciences Research Institute (MSRI), Berkeley, CA, in 1998. Research at MSRI is supported in part by NSF grant DMS 9701755. The work of the second author was also supported by NSF grant DMS 9627330.

great deal of mathematical and physical interest (see [18] and other recent references therein). Its numerical simulation also remains of terrific importance. This problem is solved routinely in studies of stellar and planetary dynamics [37, 27, 1], and related problems arise in quasi-classical studies of atomic systems [29, 10, 32].

While many numerical schemes for the N-body problem have been developed over the years [13, 1], these codes may exhibit deficiencies in very long time integration or in scattering studies. One approach to improving the qualitative behavior of numerical simulation methods is to incorporate some of the many geometric properties of the phase flow, such as time-reversal symmetry, symplectic structure, and integrals such as angular momentum. Methods which preserve symmetries and invariants are sometimes referred to as *geometric* or *mechanical* integrators [30, 21, 3]. Symplectic and symmetric algorithms for smooth N-body trajectories (without close approaches) have been successfully used for long-term simulations of the solar system [37].

Close approaches introduce both theoretical and computational challenges for the Coulombic N-body problem. In the approach described here, we assume that there is potentially a large number of bodies, but at any given time only a few bodies are engaged in very close approaches. None of the geometric algorithms can be expected to preserve structure or provide substantial efficiency improvements under frequent discontinuous switchings (such as may be used for close encounters). This problem was discovered and explained by Calvo and Sanz-Serna [4] in the context of variable stepsize symplectic integration: *frequent changes in the timestepping map lead to a deterioration of the numerical stability of geometric methods.* For variable stepsizes, it turns out to be more practical to use a Sundman or Poincaré-type time-transformation to rescale the vector field or Hamiltonian in such a way that (i) geometric structure is preserved, and (ii) the new system can be solved with a fixed timestep. Articles by Stoffer [33], Hut et al. [17], and Leimkuhler with collaborators [15, 14] developed the use of time-reversible variable stepsizes for N-body problems. More recently several articles have indicated how regularization methods can be incorporated in a reversible or symplectic framework [23, 20, 16].

Another approach to variable stepsize integration, based on a hierarchical splitting, was developed by Skeel and Biesiadecki [31], and we exploit this idea in a somewhat different way to define a *smooth switching* of the close-approaching bodies. We then separate the weaker interactions by splitting, and we identify potential close approaches using *Verlet neighbor lists* [35], partitioning the bodies into small groups. The multiple timestepping framework of molecular dynamics (r-RESPA [34]) allows some control of the amount of computation performed between evaluations of the long-range forces. A Sundman time transformation is also incorporated in the manner of [14, 20].

For problems with only two-body close approaches, much of the computational work following splitting will be in the resolution of isolated pairs. The switches somewhat complicate the dynamics of the two-body pairs (they are no longer pure Kepler problems). We describe an efficient technique for recovering their dynamics, based on a reduced coordinate set, a regularization, and an efficient implementation of a higher order implicit Gauss–Legendre integrator.

When three or more body interactions in close approach must be taken into consideration (in the localized subsystem), a system of relative variables can be introduced and a regularization developed in terms of two-body pairs. In the standard approach based on Kustaanheimo–Stiefel transformation, the coupling of variables in the regularized Hamiltonian makes high-accuracy geometric integrators very costly.

We suggest instead an approach based on relative coordinates, splitting, and two-body pair integration.

Some other methods similar to our own have recently appeared in the astronomical literature. To date, no other writers appear to have successfully combined smooth switching of close encounters, coordinate regularization, and variable stepsize time-reversible integration. Chambers [5] uses a C^1 switch and a splitting into weak and strong terms. The strong terms are integrated using a fixed stepsize method, which, though not itself reversible, is used with a small enough timestep so that these terms can be viewed as exactly resolved. We believe that this approach is inefficient compared to our own, since no time or coordinate regularization is used. Moreover, we find that the radius of the switch, which is critical to the efficiency of the algorithm, can be substantially reduced in our scheme, compared to the observations of Chambers, due to our use of variable stepsize. To implement variable stepsize, Duncan, Levison, and Lee [7] follow the idea of Skeel and Biesiadecki [31], using a hierarchy of shells (inducing a C^1 or C^3 splitting) to reduce stepsize according to body separation. This method is time-reversible—also symplectic—but cannot treat extreme close approaches, for which it is well known that some sort of coordinate regularization is needed. An article by Rauch and Holman [28] considers several techniques (some of which are not geometric integrators) which incorporate switches, together with a time-only regularization ("Poincaré transformation") as in Waldvogel [36], Zare and Szebehely [39], and Mikkola [25]. In numerical experiments, our method generally appears to perform well in comparison with these different alternatives when close approaches are present. Athough very different timestep costs make precise efficiency comparisons difficult, some preliminary comparative discussion may be found in the section on numerical experiments.

The remainder of the article is arranged as follows. In section 2, we introduce the smoothly switched vector field splitting and show how it can be combined with multiple timestepping and the reversible adaptive Verlet method. Section 3 describes how the bodies are partitioned for the purposes of efficient propagation, and section 4 considers in detail the computation of the switched two-body dynamics for close approaching pairs. Section 5 then discusses efficient regularization groups of more than two bodies. Finally, section 6 includes numerical experiments with a Kepler problem and several variants of the "two fixed-points problem" as well as some discussion of comparisons with existing methods presented recently in the astronomical literature.

In summary, we propose a new N-body integrator that combines a time-reversible adaptive stepsize integrator with smooth switches to isolate close approaching bodies. Two-body close encounters are solved by a new efficient, regularized, high order Gauss–Legendre integrator. If more than two bodies are involved in a close interaction, we suggest a new approach to split these into two-body problems. Our integrator could easily be adapted to the Coulombic problems arising in quasi-classical simulations of atomic systems [29, 10, 32, 20], to systems with multiple fixed bodies, or to systems subject to applied fields or arbitrary perturbing potentials, using additional splittings.

2. Smooth switches, splitting, multiple timestepping, and reversible adaptive timestepping. We use the term *smooth switch* to describe a real (backward sigmoidal) function χ which smoothly passes from one to zero in some finite subinterval of $\mathbb{R}_+$:

$$\chi \in C^k(\mathbb{R}_+, [0,1]), \qquad \chi(r) = 1, \quad r < r_-, \qquad \chi(r) = 0, \quad r \geq r_+, \qquad \chi'(r) \leq 0.$$

Piecewise polynomial switches of any desired smoothness can be constructed as follows. First define a polynomial

$$p(x) = \left(1 - \left(\frac{x}{h}\right)^2\right)^{k+1}.$$

This function is positive in $(-h, h)$ and is easily seen to satisfy the conditions $p(0) = 1$, $p^{(l)}(h) = p^{(l)}(-h) = 0$, $l = 0, 1, \ldots, k$. Normalizing the integral of this polynomial on the interval $[-h, h]$ and subtracting from unity results in a new polynomial,

$$q(x) = 1 - \frac{\int_{-h}^{x} p(\xi)d\xi}{\int_{-h}^{h} p(\xi)d\xi}.$$

Now for given r_- and r_+ we set $h = (r_+ - r_-)/2$, $\bar{r} = (r_+ + r_-)/2$; then the function

$$\chi(r) = \begin{cases} 1, & r \le r_-, \\ q(r - \bar{r}), & r_- \le r < r_+, \\ 0, & r \ge r_+ \end{cases} \tag{2.1}$$

is a C^{k+1} switch (See Figure 2.1). The fact that we can choose χ to be piecewise polynomial will be found to have some positive ramifications for the efficiency of our code.

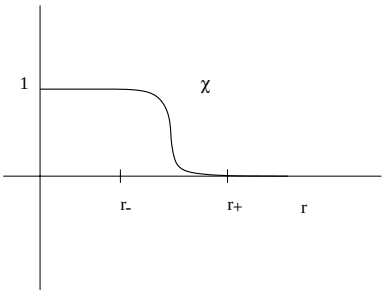


FIG. 2.1. *A smooth switch.*

The purpose of the smooth switch is to limit the high-accuracy resolution of trajectories to close approaching bodies, while reducing the number of force evaluations for weak interactions. We decompose each pair interaction in the gravitational potential using smooth switches:[1]

$$f = \underbrace{-\sum_{i=1}^{N-1}\sum_{j=i+1}^{N} \chi(r_{ij})\frac{Gm_im_j}{r_{ij}}}_{f_{\text{loc}}} + \underbrace{-\sum_{i=1}^{N-1}\sum_{j=i+1}^{N} (1 - \chi(r_{ij}))\frac{Gm_im_j}{r_{ij}}}_{f_{\text{weak}}}.$$

In a two-body problem, there are three distinct regimes in the dynamics of f_{loc}. For separations $r \le r_-$, the dynamics are pure Keplerian. For $r \ge r_+$, the bodies

[1]For simplicity, we here suppose that the parameters r_-, r_+ of each switch are all identical. In many cases, it may be more practical to choose the parameters of the switch dependent on the masses of the particles of the pair. In the related classical atomic model, the switch might also depend on the product of the charges of the particle pair.

are in free motion. For $r \in (r_-, r_+)$, the motion is determined by the (somewhat artificial) dynamics of the switch, integrable dynamics in a decaying force field.

To design a one-step method, we utilize a Hamiltonian splitting of the form $\mathscr{H} = \mathscr{H}_{\rm loc} + f_{\rm weak}$, where

$$\mathscr{H}_{\rm loc} := T + f_{\rm loc}.$$

Let $\mathbf{\Phi}_{t,\mathscr{H}}$ represent the t- flow map on Hamiltonian $\mathscr{H}$. We construct a splitting method using multiple timestepping [34, 9] with m inner substeps:

$$\mathbf{\Phi}_{\Delta t,\mathscr{H}} \approx \hat{\mathbf{\Phi}}_{\Delta t,\mathscr{H}} := \mathbf{\Phi}_{\Delta t, f_{\rm weak}} \left[\hat{\mathbf{\Phi}}_{\frac{1}{m}\Delta t,\mathscr{H}_{\rm loc}} \right]^m.$$

(In most cases, we would take $m = 1$, but the incorporation of multiple timestepping adds some flexibility.) Here $\hat{\mathbf{\Phi}}_{\Delta t,\mathscr{H}_{\rm loc}}$ is a time-reversible approximation method for solving the localized gravitational problem, as described in the next section.

The method $\hat{\mathbf{\Phi}}_{\Delta t,\mathscr{H}}$ is a first order fixed stepsize integrator, and it is not time-reversible. We use this method as the basis for constructing a second order variable stepsize reversible integrator according to the prescription of [14]:

$$\begin{aligned} \begin{pmatrix} \boldsymbol{q}^{n+1/2} \\ \boldsymbol{p}^{n+1/2} \end{pmatrix} &= \hat{\mathbf{\Phi}}_{\frac{1}{2}\Delta t_n,\mathscr{H}} \begin{pmatrix} \boldsymbol{q}^{n} \\ \boldsymbol{p}^{n} \end{pmatrix}, \\ \frac{1}{\Delta t_n} + \frac{1}{\Delta t_{n+1}} &= \frac{2}{g_{n+1/2}\Delta t}, \\ \begin{pmatrix} \boldsymbol{q}^{n+1} \\ \boldsymbol{p}^{n+1} \end{pmatrix} &= \hat{\mathbf{\Phi}}^*_{\frac{1}{2}\Delta t_{n+1},\mathscr{H}} \begin{pmatrix} \boldsymbol{q}^{n+1/2} \\ \boldsymbol{p}^{n+1/2} \end{pmatrix}, \end{aligned} \tag{2.2}$$

where

$$g_{n+1/2} = g(\boldsymbol{q}^{n+1/2}, \boldsymbol{p}^{n+1/2})$$

and $g(\boldsymbol{q}, \boldsymbol{p})$ is the time-reparameterization function, a smooth, positive, scalar-valued function which is invariant under $\boldsymbol{p} \to -\boldsymbol{p}$ (see below). The adjoint method needed above is easily computed since $\hat{\mathbf{\Phi}}_{\Delta t,\mathscr{H}}$ is the composition of symmetric maps:

$$\hat{\mathbf{\Phi}}^*_{\Delta t,\mathscr{H}} = \left[\hat{\mathbf{\Phi}}_{\frac{1}{m}\Delta t,\mathscr{H}_{\rm loc}} \right]^m \mathbf{\Phi}_{\Delta t, f_{\rm weak}}.$$

Although the method is formally second order (symmetric methods have even order [11]), we expect to utilize a higher order integrator for the local interaction dynamics, as described in the next and following sections. Viewed as a second order method, the scheme is therefore expected to have a relatively small leading error constant compared to a method such as Störmer–Verlet, which does nothing special to integrate close approaching bodies. If desired, a third order splitting method could be used in place of $\hat{\mathbf{\Phi}}_{\Delta t,\mathscr{H}}$, resulting in a fourth order method overall. (See [22] for a discussion of how to construct such splittings.)

We anticipate that the considerations for the choice of time-transformation will be similar to those discussed in [2, 20]. To ensure stable integration near close encounters, we need a transformation of the form $g \sim \min_{ij} r_{ij}^2$, where $\min_{ij} r_{ij}$ represents the smallest pair separation. The use of such an approach is not efficient for even moderately difficult trajectories; in the presence of very close encounters the progress

of the integration will practically be halted. Thus, we adopt the control suggested in [20],

$$g_{\min} = \frac{1}{1 + (\min_{ij} r_{ij})^{-3/2}}. \tag{2.3}$$

In some cases, it may be desirable to have a smooth control, in which case we could use

$$g_{\mathrm{sm}} = \frac{1}{1 + \left(\sum_{i<j} r_{ij}^{-3m/2}\right)^{1/m}},$$

where m is a positive integer [2].

3. Integration of $\mathscr{H}_{\mathbf{loc}}$. In this and the following section, we describe the integrator for $\mathscr{H}_{\mathrm{loc}}$. We will assume that the solution is desired on the time interval $[0, \Delta t]$, that initial positions $\boldsymbol{q}^0$ and momenta $\boldsymbol{p}^0$ are provided, and that new values $\boldsymbol{q}^1$ and $\boldsymbol{p}^1$ are to be computed.

Because of the use of switching functions, a relatively small number of the interaction terms are active in $\mathscr{H}_{\mathrm{loc}}$ at any given timestep; most of the particles simply drift linearly with fixed momenta. We introduce the concept of a *local interaction graph* G based on the nonzero potential interactions between the bodies. This graph will be used to limit the local computation (in the relatively expensive regularization variables) to minimal subgroups of bodies. However, there is some question about how to define G. Clearly the vertices of G are the body indices, but what should we take for the edges?

Given vertices i and j, it is evidently not sufficient to add the edge $\bar{ij}$ only when $\|\boldsymbol{q}_i^0 - \boldsymbol{q}_j^0\| \le r_+$, since, during the timestep, the two bodies may move closer to each other and thus have a close interaction by the end of the timestep. Moreover, the one-step methods we propose for integrating the few-body problems will evaluate the forces at an intermediate point in the subinterval. Still it is not sufficient to simply link i and j based on the initial, final, and intermediate steps, since it is necessary to know these interactions *at the start* of the timestep (after all, the whole purpose of constructing the graph is to restrict the computation to local groups!). To define the interaction graph, we therefore need a very cheap method to predict, based only on the initial positions and velocities, which particles have the potential to have a local interaction during the current timestep. The same technique used to construct *Verlet neighbor lists* [35] in molecular dynamics can be used for this purpose.

To identify potential close approaches, we proceed as follows for the ith body (being careful to work with squared distances to avoid computation of extra square roots). We iterate over the bodies with index $j > i$. Define

$$\boldsymbol{\Delta q} = \boldsymbol{q}_i^0 - \boldsymbol{q}_j^0, \qquad \boldsymbol{\Delta p} = \boldsymbol{p}_i^0 - \boldsymbol{p}_j^0.$$

We first compute $\sigma^0 = |\boldsymbol{\Delta q}|^2$. Clearly if $\sigma^0 < r_+^2$, we should add edge $\bar{ij}$ to G, but because our method is heuristic (based only on linear trajectories), we include a safety factor $\beta > 1$ and check instead $\sigma^0 < \beta r_+^2$. Next, we compute the time of closest approach of the linear trajectories through $(\boldsymbol{q}_i^0, \boldsymbol{p}_i^0)$ and $(\boldsymbol{q}_j^0, \boldsymbol{p}_j^0)$:

$$t^* = -\frac{\boldsymbol{\Delta q} \cdot \boldsymbol{\Delta p}}{|\boldsymbol{\Delta p}|^2}.$$

If $t^* \in [0, \Delta t]$, then we compute the squared separation between these two trajectories at t^*, $\sigma^* = |\boldsymbol{\Delta q} + t^* \boldsymbol{\Delta p}|^2$, and check if $\sigma^* < \beta r_+^2$. Finally, we compute the separation at the right endpoint of the time interval, $\sigma^1 = |\boldsymbol{\Delta q} + \Delta t \boldsymbol{\Delta p}|^2$, adding the edge if this is less than βr_+^2.

After updating the interaction graph, we subdivide it into L disjoint connected components $\Omega_1, \Omega_2, \ldots, \Omega_L$. These components will generally consist of one or several indices. The cost of resolving the close approaches rises very rapidly with the number of bodies, while the attainable accuracy diminishes, but close approaches of three or more bodies are rare. In the next section we focus on the case of two-body approaches. Later, we will describe a method for handling higher-body collisions.

4. Fast integrator for the switched problem: Two-body case. The computation of the close encounters is typically the most costly part of the calculation, so substantial algorithmic effort is warranted in the interest of overall efficiency. In this section we describe a technique for two-body approaches based on polar coordinates in the plane of motion, regularization, and the Gauss–Legendre family of higher order implicit Runge–Kutta methods.

Any Hamiltonian system of the form

$$H_{\text{2bdy}} = \frac{1}{2m_1}|\boldsymbol{p}_1|^2 + \frac{1}{2m_2}|\boldsymbol{p}_2|^2 + \phi(|\boldsymbol{q}_1 - \boldsymbol{q}_2|)$$

is integrable. For Kepler's problem, an elegant solution was known at least to Gauss and is detailed in [19]. A number of papers have considered efficient numerical methods for solving the associated nonlinear equation (see, e.g., [24, 26]). A method avoiding all transcendental functions was used in [20], based on implicit midpoint applied in the Kustaanheimo–Stiefel variables, but this scheme may not provide sufficient accuracy in some cases.

Solutions for special cases of the two-body problem with various power potentials are discussed in [8]; however, for *arbitrary* ϕ, the computation of the associated quadrature and nonlinear equations becomes complicated by potential singularities and a sign change of the integrand, and an involved procedure is then needed to perform the computations, including a special quadrature algorithm and case-dependent changes of variables.

Converting to relative coordinates, rotating the plane of motion (with normal $\boldsymbol{n} = (\boldsymbol{q}_1 - \boldsymbol{q}_2) \times (\boldsymbol{p}_1 - \boldsymbol{p}_2) = \text{constant}$) onto the xy coordinate plane, and then introducing polar coordinates gives the Hamiltonian

$$H_{\text{plane}} = \frac{1}{2\bar{m}} p_r^2 + \frac{1}{2\bar{m}} \frac{l^2}{r^2} + \phi(r),$$

where $l = p_\theta$ is the local angular momentum of the particular approaching pair. This is an integrable system. We could proceed to recover the solution by piecing together solutions of the separable scalar ODE

$$\frac{d}{dt} r = \pm \sqrt{2\bar{m} \left(E_0 - \phi(r) - \frac{l^2}{r^2} \right)},$$

where $E_0 = H_{\text{plane}}$ is the energy of the reduced problem. For general ϕ, the numerical solution of this differential equation in terms of quadratures is inefficient. Not only do we have to design an accurate numerical quadrature subject to a switching of sign, but the solution is then obtained only in implicit form, defined by a certain nonlinear

equation. Observe also that there is nothing to prevent $l = 0$; thus arbitrarily close approaches of the bodies (or even collisions) are indeed possible, causing a singularity in the integrand. Despite these problems, a viable method for the general case could undoubtedly be derived based on quadrature, but we are skeptical that it would be as efficient and robust as the alternative we describe in what follows.

The canonical equations of H_{plane} are

$$\begin{aligned}\frac{d}{dt}r &= \frac{1}{\bar{m}}p_r,\\ \frac{d}{dt}p_r &= \frac{1}{\bar{m}}\frac{l^2}{r^3} - \phi'(r),\\ \frac{d}{dt}\theta &= \frac{1}{\bar{m}}\frac{l}{r^2}.\end{aligned}$$

By introducing a new variable $\eta = r \cdot p_r$, performing a Sundman transformation $dt/d\tau = r$, and taking advantage of the constant energy, this system can be written as

$$\frac{d}{d\tau}r = \frac{1}{\bar{m}}\eta, \tag{4.1}$$

$$\frac{d}{d\tau}\eta = 2rE_0 - 2r\phi(r) - r^2\phi'(r), \tag{4.2}$$

$$\frac{d}{d\tau}t = r, \tag{4.3}$$

$$\frac{d}{d\tau}\theta = \frac{1}{\bar{m}}\frac{l}{r}. \tag{4.4}$$

These equations are no longer Hamiltonian; however, the symmetry of the original system is retained. The first three of these equations can be solved independent of the last one. And, since $\phi(r) = -\gamma/r \cdot \chi(r)$, $\chi(r)$ piecewise polynomial, all singularities have been removed from (4.1)–(4.2); thus, at least theoretically, they can be solved even through head-on collisions.

The solution of the regularized equations (4.1)–(4.4) requires an accurate and stable geometric integration method such as the efficiently implemented Gauss–Legendre method described below.

High order is needed in this step because the regularization process (absolutely essential to stable integration) introduces the energy of the two-body problem as a parameter of the differential equations, hence upsetting the iterated-map property which guarantees structural stability properties such as orbital symmetry [20]. Only in the case of a pure Kepler problem is the invariant H_{plane} quadratic, hence conserved by the Gauss methods [6]. Especially in the switching regime of our two-body potential, the energy may fluctuate significantly; thus a high-order method is likely to be needed.

A Gauss–Legendre method for the regularized reduced problem. We will here briefly describe the implementation of a fully implicit Runge–Kutta method applied to the regularized problem (4.1)–(4.4). A discussion of implementation issues for general ordinary differential equations can be found in [12, IV.8]. In this section, we concentrate on issues specific to the given problem, which in what follows will be written as

$$\frac{d^2}{d\tau^2}r = \frac{1}{\bar{m}}f(r), \quad \frac{d}{d\tau}t = r, \quad \frac{d}{d\tau}\theta = \frac{1}{\bar{m}}\alpha(r), \quad \text{and} \quad \eta = \bar{m}\frac{d}{d\tau}r. \tag{4.5}$$

These equations are solved by some (possibly) high order Gauss–Legendre methods [12, IV.5]. Using the standard Butcher notation, the coefficients in an s-stage method are given by a matrix $\boldsymbol{A} = (a_{ij})_{i,j=1}^{s}$ and vectors $\boldsymbol{b} = (b_1, b_2, \dots, b_s)^T$ and $\boldsymbol{c} = (c_1, c_2, \dots, c_s)^T$. Applied to (4.5), it results in the following system of nonlinear equations ($\hat{a}_{ij}$ being the elements of the matrix $\boldsymbol{A}^2$):

$$R_i = r_n + \frac{1}{\bar{m}} c_i \Delta\tau \eta_n + \frac{1}{\bar{m}} \Delta\tau^2 \sum_{j=1}^{s} \hat{a}_{ij} f(R_j), \quad i = 1, 2, \dots, s, \tag{4.6}$$

and the solution is updated by

$$t_{n+1} = t_n + \Delta\tau \sum_{i=1}^{s} b_i R_i, \tag{4.7}$$

$$r_{n+1} = r_n + \Delta\tau \frac{1}{\bar{m}} \eta_n + \frac{1}{\bar{m}} \Delta\tau^2 \sum_{i=1}^{s} \sum_{j=1}^{s} b_i a_{ij} f(R_j), \tag{4.8}$$

$$\eta_{n+1} = \eta_n + \Delta\tau \sum_{i=1}^{s} b_i f(R_i), \tag{4.9}$$

$$\theta_{n+1} = \theta_n + \frac{1}{\bar{m}} \Delta\tau \sum_{i=1}^{s} b_i \alpha(R_i). \tag{4.10}$$

A curious twist is that the timestep $\Delta t_n = t_{n+1} - t_n$ is known, but the corresponding fictive timestep $\Delta\tau$ must be computed. Thus (4.6) and (4.7) must be solved for $R_1, \dots, R_s$ and $\Delta\tau$ simultaneously. Newton iteration techniques normally will be adequate. The Jacobian matrix of this system of $s + 1$ nonlinear equations can be written in block form,

$$\boldsymbol{J} = \begin{pmatrix} \hat{\boldsymbol{J}} & \boldsymbol{v} \\ \boldsymbol{u}^T & w \end{pmatrix}.$$

The elements of the $(s \times s)$-matrix $\hat{\boldsymbol{J}}$ are

$$J_{ij} = \delta_{ij} - \frac{\Delta\tau^2}{\bar{m}} \hat{a}_{ij} f'(R_j),$$

where δ_{ij} is 1 if $i = j$ and otherwise zero, $\boldsymbol{v}$ has elements

$$v_i = -\frac{1}{\bar{m}} c_i \eta_n - \frac{2}{\bar{m}} \Delta\tau \sum_{i=1}^{s} \sum_{j=1}^{s} \hat{a}_{ij} f(R_j),$$

$\boldsymbol{u} = -\Delta\tau \boldsymbol{b}^T$, and $w = -\sum_{i=1}^{s} b_i R_i$. The Jacobian can be computed in the beginning of each step and then held fixed during the iterations. Having a good predictor is critical for the overall success of the method. One option is to use a high order Taylor series approximation, obtained through differentiation of the differential equation. This is especially viable because the function $f(r)$ is piecewise polynomial. Extrapolating values from a previous step (using the collocation polynomial) is inadvisable since the integration of (4.5) over one (outer) step is only a part of the overall algorithm; thus the right-hand side of the equation changes from step to step. For the same reason, the Jacobian cannot be kept fixed over several steps.

For very close encounters, the Jacobian $\boldsymbol{J}$ tends to become singular, and special care has to be taken for the solution of the nonlinear equations.

Since the energy E_0 is used as a parameter in the regularized equations, time symmetry is not really retained. However, since high order methods are used, we are willing to accept this small destruction of time symmetry. In fact, as we will show later, experiments show that this discrepancy in energy has less effect on the time symmetry than the roundoff errors introduced when solving the unregularized equations.

5. Higher-body close encounters. We next briefly consider numerical integrators for the case that a component of the local interaction graph can include three or more bodies. The equations in this case are no longer integrable, so the only possibility is to derive an appropriate numerical discretization scheme. In general, our problem is to integrate numerically a d-body problem with Hamiltonian of the form

$$\frac{1}{2}\sum_{i=1}^{d}\frac{|\boldsymbol{p}_i|^2}{m_i}+\sum_{i=1}^{d-1}\sum_{j=i+1}^{d}\phi(|\boldsymbol{q}_i-\boldsymbol{q}_j|).$$

Due to our use of splitting and time-reparametrization, we can restrict attention to the case where the number of bodies d is small.

As is the usual practice (see, e.g., [23]), we first introduce relative coordinates for the pair separations,

$$\begin{aligned}\boldsymbol{\delta}_1=\boldsymbol{q}_1-\boldsymbol{q}_2,\quad \boldsymbol{\delta}_2=\boldsymbol{q}_1-\boldsymbol{q}_3,\quad \ldots,\quad \boldsymbol{\delta}_{d-1}=\boldsymbol{q}_1-\boldsymbol{q}_d,\\ \boldsymbol{\delta}_d=\boldsymbol{q}_2-\boldsymbol{q}_3,\quad \ldots,\quad \boldsymbol{\delta}_{2d-3}=\boldsymbol{q}_2-\boldsymbol{q}_d,\\ \ldots,\quad \boldsymbol{\delta}_{\frac{1}{2}(d^2-d)}=\boldsymbol{q}_{d-1}-\boldsymbol{q}_d.\end{aligned}$$

Together with the center of mass, the $D=(d^2-d)/2$ variables (although redundant for $d>3$ since in that case $D>d$) are adequate to describe any configuration. Canonical equations can be developed in terms of the relative variables by introducing momenta $\boldsymbol{\pi}_i$ canonically conjugate to the $\boldsymbol{\delta}_i$. Neglecting the center of mass motion, the Hamiltonian becomes

$$H_{\rm rel}=\sum_{i=1}^{D}\frac{|\boldsymbol{\pi}_i|^2}{2\bar{m}_i}+\sum_{i=1}^{D}\sum_{\substack{j=1\\ j\neq i}}^{D}\sigma_{ij}\boldsymbol{\pi}_i\cdot\boldsymbol{\pi}_j+\sum_{i=1}^{D}\phi(|\boldsymbol{\delta}_i|),$$

where $\bar{m}_i=1/(1/m_\alpha+1/m_\beta)$, with m_α and m_β the masses of the bodies of the ith pair, and σ_{ij} is either zero (if the pairs on which $\boldsymbol{\delta}_i$ and $\boldsymbol{\delta}_j$ are based involve no common body) or else $\pm$ the reciprocal of the common mass of the ith and jth pairs of bodies, depending on the order of the differences.

Now there are several ways to proceed. A traditional course, described concisely in [23], is to apply the Kustaanheimo–Stiefel regularizing transformation to all the variables. The kinetic energy metric is then position dependent and rather complicated. The nonseparable nature of the Hamiltonian precludes the use of (known classes of) explicit geometric integrators, leaving us essentially with implicit methods such as the Gauss–Legendre Runge–Kutta methods or Lobatto IIIA-B partitioned Runge–Kutta pairs [11], but in this case, there are no simple reductions of the numbers of variables such as we found in the previous section. For example, a fourth order method for the three-body problem would require the solution of a nonlinear system of dimension $2\times 24=48$ at each step. This is clearly unacceptable in long-term simulations, for which many such nonlinear systems may need to be solved.

Instead, we will look for a splitting of $H_{\rm rel}$ that requires only the solution of two-body problems at each step. The numerical method of the previous section can then be used to solve the two-body problems.

One such splitting would break $H_{\rm rel}$ into two parts,

$$H_{\rm rel} = H_1 + H_2,$$

where

$$H_1 = \sum_{i=1}^{D} \frac{|\boldsymbol{\pi}_i|^2}{2\bar{m}_i} + \sum_{i=1}^{D} \phi(|\boldsymbol{\delta}_i|),$$

and

$$H_2 = \sum_{i=1}^{D} \sum_{\substack{j=1 \\ j\neq i}}^{D} \sigma_{ij} \boldsymbol{\pi}_i \cdot \boldsymbol{\pi}_j.$$

H_1 consists of D decoupled two-body problems, whereas H_2 is only $\boldsymbol{\pi}$ dependent; hence both terms are integrable.

An alternative splitting method would divide H into D parts,

$$H_{\rm rel} = G_1 + G_2 + \cdots + G_D,$$

where

$$G_i = \frac{|\boldsymbol{\pi}_i|^2}{2\bar{m}_i} + \phi(|\boldsymbol{\delta}_i|) + \sum_{\substack{j=1 \\ j\neq i}}^{D} \sigma_{ij} \boldsymbol{\pi}_i \cdot \boldsymbol{\pi}_j.$$

Each of the D terms can be seen to be integrable, since only the ith momentum vector changes during the integration of G_i. In the case of a three-body problem, this splitting has three terms. For higher order, the number would rapidly increase.

The differential equations on $H_{\rm rel}$ are given by, for $i = 1, \ldots, D$,

$$\begin{aligned} \frac{d}{dt}\boldsymbol{\delta}_i &= \frac{1}{\bar{m}_i}\boldsymbol{\pi}_i + \sum_{\substack{j=1 \\ j\neq i}}^{D} \sigma_{ij}\boldsymbol{\pi}_j, \\ \frac{d}{dt}\boldsymbol{\pi}_i &= -\frac{\phi'(|\boldsymbol{\delta}_i|)}{|\boldsymbol{\delta}_i|}\boldsymbol{\delta}_i. \end{aligned}$$

Define $\boldsymbol{L} = \sum_{i=1}^{D} \boldsymbol{\delta}_i \times \boldsymbol{\pi}_i$. Then it is easy to see that

$$\begin{aligned} \frac{d}{dt}\boldsymbol{L} &= \sum_{i=1}^{D} \left(\frac{d}{dt}\boldsymbol{\delta}_i \times \boldsymbol{\pi}_i + \boldsymbol{\delta}_i \times \frac{d}{dt}\boldsymbol{\pi}_i \right) \\ &= \sum_{i=1}^{D} \left(\left(\frac{1}{\bar{m}_i}\boldsymbol{\pi}_i + \sum_{\substack{j=1 \\ j\neq i}}^{D} \sigma_{ij}\boldsymbol{\pi}_j \right) \times \boldsymbol{\pi}_i - \boldsymbol{\delta}_i \times \left(\frac{\phi'(|\boldsymbol{\delta}_i|)}{|\boldsymbol{\delta}_i|}\boldsymbol{\delta}_i \right) \right) \\ &= -\sum_{i=1}^{D} \sum_{\substack{j=1 \\ j\neq i}}^{D} \sigma_{ij}\boldsymbol{\pi}_i \times \boldsymbol{\pi}_j \\ &= 0 \end{aligned}$$

using the antisymmetry of the cross product and the fact that $\sigma_{ij} = \sigma_{ji}$; $\boldsymbol{L}$ can be viewed as a representation of the angular momentum of the system. Moreover, both splittings are also easily seen to conserve angular momentum, since each of the terms has this feature and the discretization is constructed by concatenation of the exact flows (or conserving approximations of the exact flows) on each term.

To obtain higher order methods, we suggest use of the concatenation technique [38, 22, 30]. For the alternative decomposition, second and fourth order splitting methods are also easily derived.

6. Numerical experiments. In the experiments described below, we have used an adaptive Verlet-leapfrog method, for which one step of the method is given by

$$\boldsymbol{\Phi}_{\frac{1}{2}\Delta t_n, f_{\text{weak}}} \circ \hat{\boldsymbol{\Phi}}_{\Delta t_n, \mathscr{H}_{\text{loc}}} \circ \boldsymbol{\Phi}_{\frac{1}{2}\Delta t_n, f_{\text{weak}}}.$$

The stepsize is updated between the steps, using (2.2) with $g = g(\boldsymbol{q}^n, \boldsymbol{p}^n)$, g given by (2.3) or its smooth equivalent.[2] For close encounters, $\hat{\boldsymbol{\Phi}}_{\Delta t_n, \mathscr{H}_{\text{loc}}}$ is solved by an eighth order Gauss method, as described.

Example 1: *The Kepler problem.* Our first example, the two-dimensional Kepler problem, has been chosen to demonstrate the reliability of the smooth switch for solving two-body close encounters. The Hamiltonian is

$$H = \frac{1}{2}(p_1^2 + p_2^2) - \frac{1}{\sqrt{q_1^2 + q_2^2}}.$$

As initial values we choose

$$q_1(0) = 1 + e, \quad q_2(0) = 0, \quad p_1(0) = 0, \quad p_2(0) = \sqrt{(1-e)/(1+e)};$$

thus the solution (q_1, q_2) forms an ellipse with eccentricity e, $0 \le e < 1$, and with period 2π. The smallest distance from the origin is $r_{\min} = 1 - e$. The system with $e = 0.9999$ was integrated over 32 periods.

In the experiment, we compare

(a) the variable stepsize leapfrog method without a switch

with the same method using

(b) a hard switch, meaning that the switch is turned on, that is $\chi = 1$, whenever $r < \bar{r}$ in the beginning of the step; otherwise the switch is turned off;

(c) an improved hard switch, in which linear extrapolation is used to decide whether r is less than $\bar{r}$ over the whole step, in which case the switch is turned on;

(d) the smooth switch (2.1) with $k = 8$.

For the variable stepsize method, the fictive stepsize was $\Delta t = 0.04$ and the average stepsize about 0.01. For the smooth switch we have used $r_+ = 0.02$, $r_m = r_+/8$ and for the hard switches $\bar{r} = (r_+ + r_-)/2$.

The superiority of the smooth switch is clearly demonstrated by Figures 6.1 and 6.2. The adaptive leapfrog without switch suffers from a precession caused by the high energy error in the close domain. The hard switches partly eliminate that precession, but a severe drift in the energy error is introduced. The combination of leapfrog and the smooth switch eliminates both.

Figure 6.3 illustrates how the energy varies over one period. To better visualize the behavior of the methods within the close domain, the energy error is shown as

[2] After the first half step, this is equivalent to the symmetric adaptive Verlet method [14].

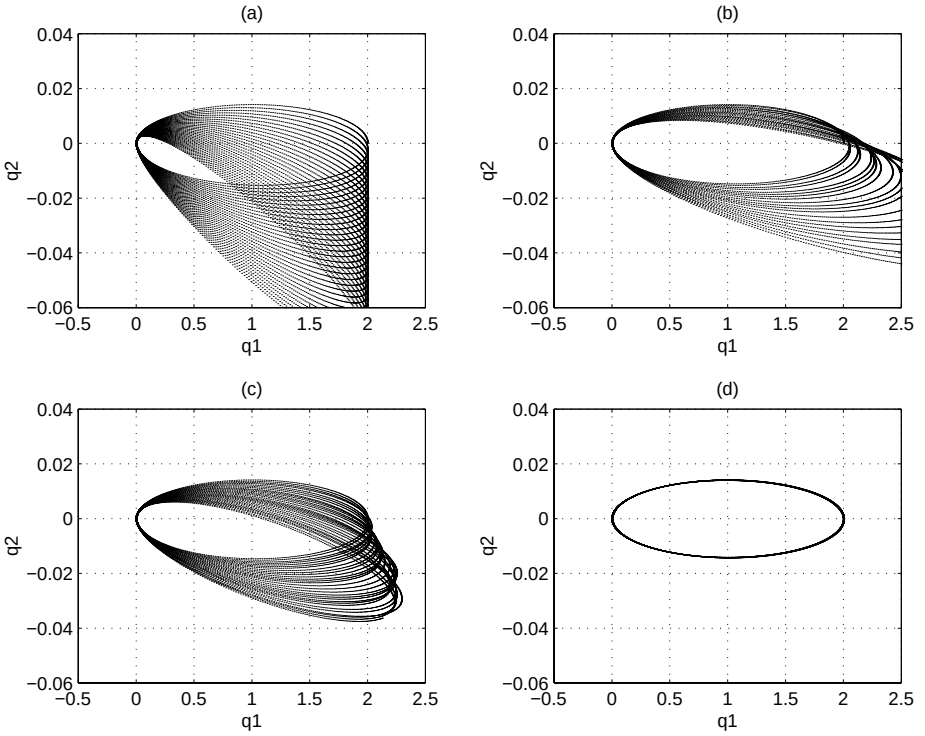


FIG. 6.1. *The orbit of the Kepler problem* ($e = 0.9999$) *solved by the variable stepsize leapfrog method* (a) *without switch, with* (b) *hard switch,* (c) *improved hard switch, and* (d) *smooth switch.*

functions of the number of steps. For comparision, the energy fluctuations for two eccentricities, $e = 0.9999$ and $e = 0.999999$, are shown. We can clearly see how the switches "cut the peak" off the energy error of the leapfrog method. The hard switches are, however, unable to regain the energy level when out of the close domain, a problem that gets worse for higher eccentricities. It is also interesting to notice that the energy error for the smooth switch is almost independent of the eccentricity. However, the smooth switch creates fluctuations in the switching regime. Experiments show that these fluctuations become worse when r_+ and r_- are close to each other.

It is also of interest to see how the switches affect the time symmetry. To this end, the Kepler problem is integrated forward one period and then backward using the same stepsize sequence. Figure 6.4 shows the norm of the displacement of $\boldsymbol{q}$ caused by this process for several choices of $r_{\min} = 1 - e$. Thus, for $r_{\min}$ less than r_+ (or $\bar{r}$ for the hard switches), the orbit is always outside the switching regime. However, when in effect, the simple hard switch ($\star$) destroys the time symmetry completely. It is also interesting to note that the method without switches ($\times$), which is known to retain time symmetry, at least in the theory, loses this property in practice.

This is caused by large rounding errors introduced in solving the unregularized equations close to the singularity. The smooth switch ($\circ$) as well as the improved hard switch ($+$) retain the time symmetry fairly well. Thus, it seems that the benefits of solving the regularized equations outweigh the disadvantage of the energy error introduced in the switching regime.

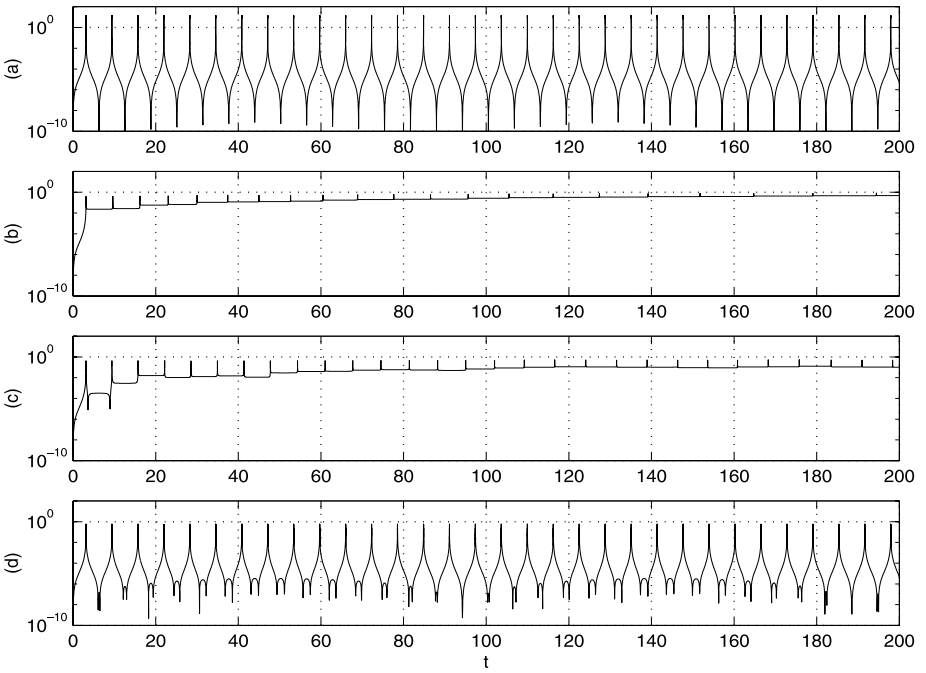


FIG. 6.2. *The relative energy error in simulation of the Kepler problem* ($e = 0.9999$). (a) *Without switch, with* (b) *hard switch,* (c) *improved hard switch, and* (d) *smooth switch.*

Even if time symmetry is maintained fairly well for the hard switch, we have already seen that the numerical solution produced by this method shows a quite irregular behavior. (This may be partly explained by the existence of an approximating continuous time-reversible flow—a truncated form of the "modified equations"—for the smoothly switched method.)

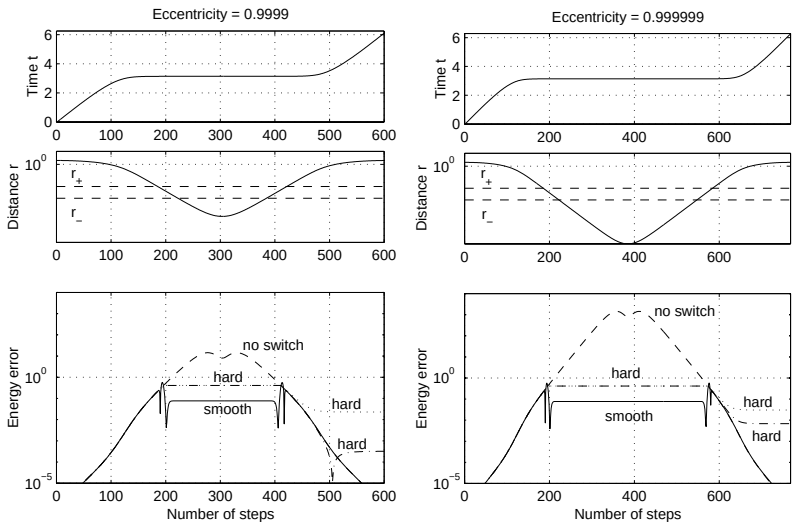


FIG. 6.3. *The distance r and the energy error over one period for the variable stepsize method without switch* (- -)*, with hard switch* ($\cdots$)*, improved hard switch* ($\cdot$—)*, and smooth switch* (–).

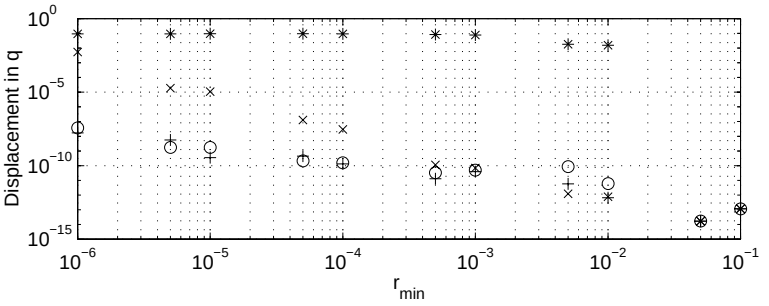


FIG. 6.4. *The displacement of* $\boldsymbol{q}$ *after integrating forward one period, and then backward again, without switch* ($\times$), *with hard switch* ($\star$), *improved hard switch* ($+$), *and smooth switch* ($\circ$).

Finally, Figure 6.5 shows the development of the global error in $\boldsymbol{q}$. As expected, both the method without the switch and the method with a smooth switch show linear error growth, although the error of the latter is only about 20% of the first one. The hard switches give no such regular behavior of the growth of the global error, and further, the magnitude of the global error is quite alarming.

Example 2: *Two fixed point problem.* We next seek to test the long-term stability behavior of the integrator. For this purpose, we employ the two fixed point (TFP) problem, which describes a particle of unit mass moving in the gravitational field of two fixed centers. The Hamiltonian is given by

$$H = \frac{1}{2}(p_1^2 + p_2^2) - \frac{m_1}{\sqrt{q_1^2 + q_2^2}} - \frac{m_2}{\sqrt{(q_1 + q_p)^2 + q_2^2}}.$$

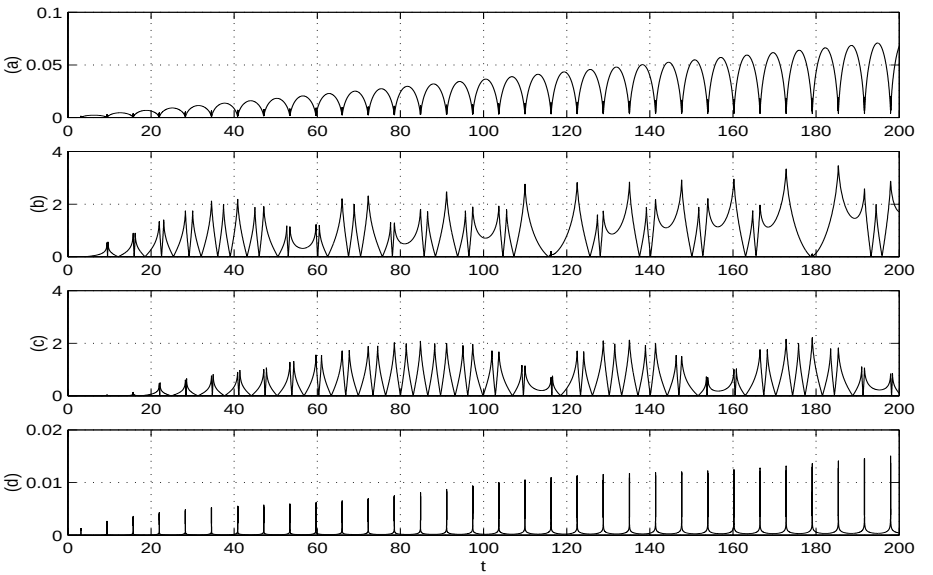


FIG. 6.5. *The global error in simulation of the Kepler problem* ($e = 0.9999$). (a) *Without switch, with* (b) *hard switch,* (c) *improved hard switch, and* (d) *smooth switch.*

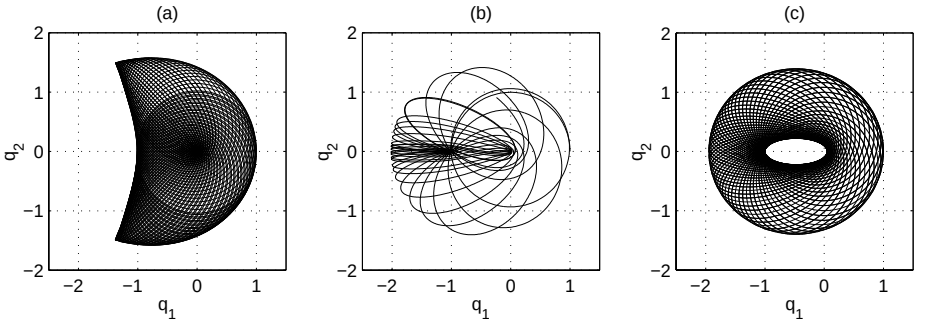


FIG. 6.6. *The orbits of the TFP problem with* (a) $q_p = 1.5$, (b) $q_p = 1.02$, *and* (c) $q_p = 0.95$.

The masses are $m_1 = 1$ and $m_2 = 0.01$, and the initial values are $q_1(0) = 1$, $q_2(0) = 0$, $p_1(0) = 0$, and $p_2(0) = 1$. The distance q_p between the two fixed centers controls the behavior of the systems. We have used three different values of q_p; exactly the same problem has been used by Rauch and Holman [28] to compare long-term stability behavior of several integrators for planetary orbits. In each case the relative energy error is monitored. The results are given in Figures 6.6 and 6.7.

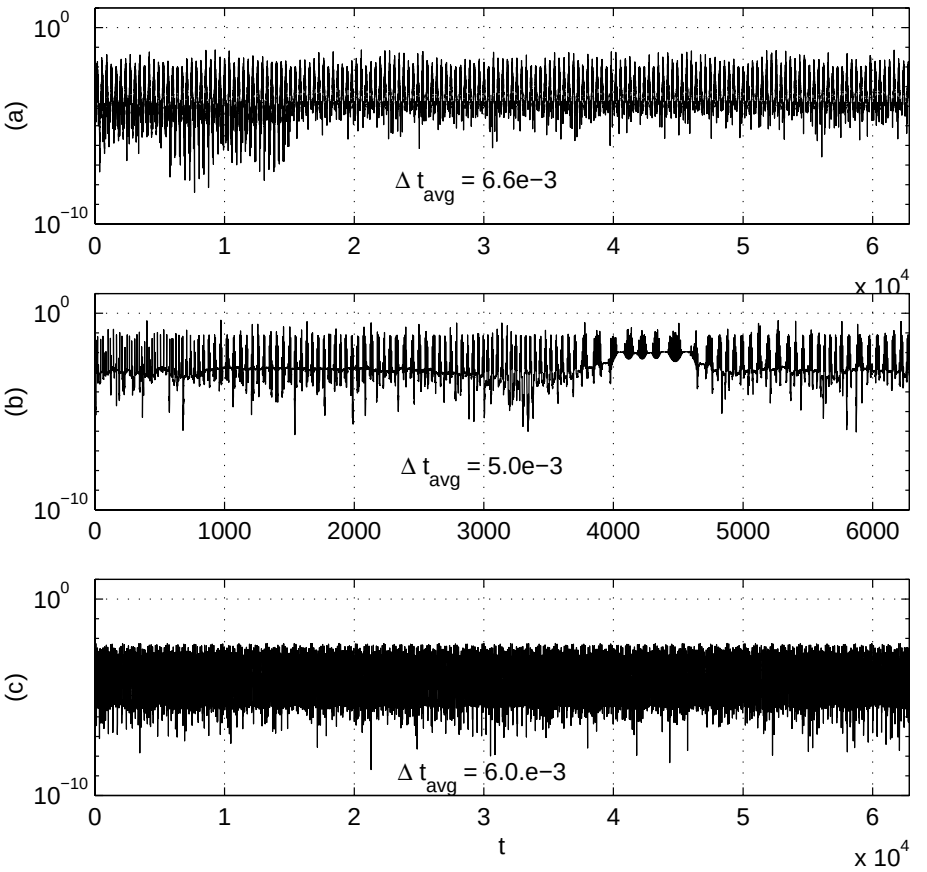


FIG. 6.7. *The relative energy error of the TFP problem with* (a) $q_p = 1.5$, (b) $q_p = 1.02$, *and* (c) $q_p = 0.95$.

(a) $q_p = 1.5$. There are no close encounters with the small mass m_2, but periodically the orbit becomes highly eccentric. No drift in the energy error can be observed.
(b) $q_p = 1.02$. In this case there are frequent close encounters with both masses, and there is no limit to how close they can be. Again, we observe that the integrator is reasonably stable. However, jumps in the energy error have been detected in some experiments. This happens when the moving body comes too close to m_2, typically if the distance becomes less than $\sim 10^{-8}$. The integrator handles very close encounters with m_1 very well.
(c) $q_p = 0.95$. The moving particle is never sufficiently close to any of the two fixed centers to activate the switch. Still, the variable stepsize algorithm is sufficient to solve the problem without any drift in the energy error.

It is natural to compare our results with the ones reported by Rauch and Holman [28]. They consider three different integrators: The Wisdom–Holman mapping [37], a potential splitting method by Duncan, Levison, and Lee [7], and a method based on Stark splitting. All of the methods are tested with and without use of the time-regularization of Mikkola [25]. Precise comparisons with our method are difficult since there are so many factors that determine the timestep cost for each method (e.g., size of the switching regime, order and type of method used for close approaches). For example, the Stark splitting method is an implicit scheme and formally, at least, requires the iteration of what is essentially a Kepler solver, although it is not clear that the authors do perform this iteration (or how they determine that it is not needed). We will summarize a few of our observations based on many experiments using our method and our reading of the article of Rauch and Holman.

In general, it appears that our method behaves at least as stably as the others in the long term, in the sense that the average energy error does not increase over time. In the first example, the energy error is about the same magnitude as for the

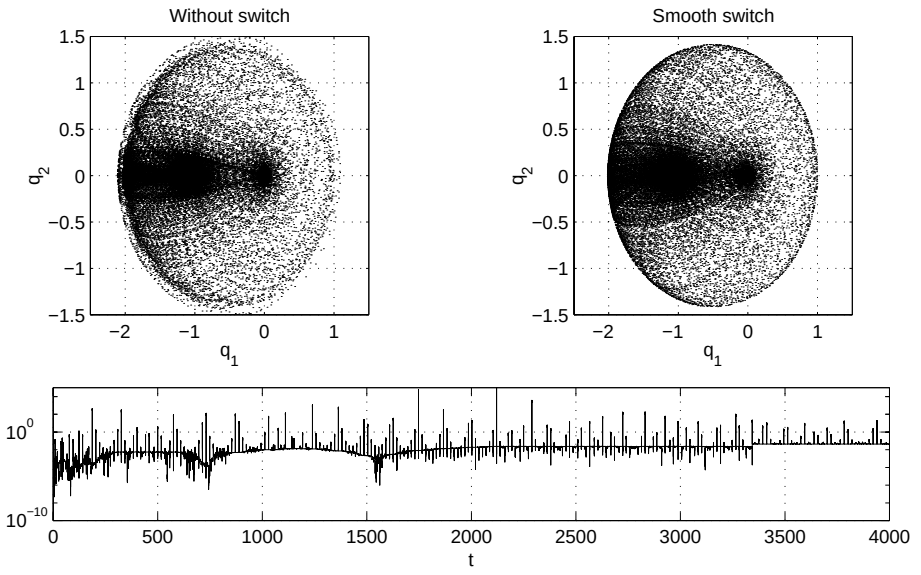


FIG. 6.8. *The upper panel gives the numerical solution points when integrating with and without the smooth switch. The lower panel shows the relative energy error for the method without switch.*

unregularized methods of Rauch and Holman; however, the error of their regularized methods are substantially smaller.

In the second example, where the body can approach both masses, our method outperforms all the methods by Rauch and Holman, with the exception of the regularized potential splitting method, which uses a relatively costly transcendental function for switching.

In the last example, our method exhibits a larger energy error than the methods of Rauch and Holman for similar average timestep. On the other hand, in these cases the moving body never enters the switching regime, so per timestep cost of our scheme is very low compared with the other methods.

Finally, in Figure 6.8 we show the numerical solution of the TFP problem with $q_p = 1.02$ when integrated with and without the smooth switch. We can clearly see how the drift in the energy error gradually destroys the regular appearance of the orbit. The unswitched method was able to solve the problem only for $t < 4400$.

7. Discussion. In this paper, we have presented a gravitational N-body integration algorithm. Smooth switching functions have been used to split the potential terms into local and weak parts. Close approaches between bodies are solved through a regularized system of equations, using a high order Gauss method. The cost of this can still be justified since close encounters are generally isolated events occurring relatively rarely along trajectories. Numerical experiments clearly demonstrate the excellent qualitative behavior of the proposed algorithm and show that the method is at least competitive with alternatives presented in the astronomical literature; particularly in cases with arbitrary close approaches, it appears to be superior to existing schemes. The number of steps within the switching regime can be reduced by reducing the switching regime; thus a correct choice of r_+ and r_- is critical for the overall performance of the method. Several other implementation issues, such as how to solve the nonlinear equations arising from applying the Gauss method to the regularized equations, still need to be addressed. Also, the proposed algorithms for solving higher-body close encounters must be studied more thoroughly and tested numerically. These and other issues are under consideration by the authors and will be addressed in a forthcoming article. However, the preliminary results given in this article are very promising and more than justify further work on the topic.

Acknowledgments. The authors would like to thank Marvin McNett (KU) for many fruitful discussions during the work, and Martin Lo (NASA JPL) for his encouragement at the start of the project.

The authors also wish to thank the referees for their helpful suggestions regarding the recent astronomical literature.

REFERENCES

[1] S. J. Aarseth, *Direct methods for N-body simulation*, in Multiple Time Scales, J. Brackbill and B. Cohen, eds., Academic Press, NY, 1985.

[2] S. Bond and B. Leimkuhler, *Time-transformations for reversible variable stepsize integration*, Numerical Algorithms, 19 (1998), pp. 55–71.

[3] C. J. Budd and A. Iserles, *Geometric integration: Numerical solution of differential equations on manifolds*, R. Soc. London Philos. Trans. Ser. A Math. Phys. Engrg. Sci., 357 (1999), pp. 945–956.

[4] M. P. Calvo and J. M. Sanz-Serna, *The development of variable-step symplectic integrators, with applications to the two-body problem*, SIAM J. Sci. Comput., 14 (1993), pp. 936–952.

[5] J. E. Chambers, *A hybrid symplectic integrator that permits close encounters between massive bodies*, Mon. Not. R. Astron. Soc., 304 (1999), pp. 793–799.

[6] G. J. Cooper, *Stability of Runge-Kutta methods for trajectory problems*, IMA J. Numer. Anal., 7 (1987), pp. 1–13.
[7] M. J. Duncan, H. F. Levison, and M. H. Lee, *A multiple time step symplectic algorithm for integrating close encounters*, Astronomical J., 116 (1998), pp. 2067–2077.
[8] H. Goldstein, *Classical Mechanics*, 2nd ed., Addison–Wesley, New York, 1980.
[9] H. Grobmuller, H. Heller, A. Windemuth, and K. Schulten, *Generalized Verlet algorithm for efficient molecular dynamics simulations with long-range interactions*, Molecular Simulation, 6 (1991), pp. 121–142.
[10] Y. Gu and J.-M. Yuan, *Chaotic scattering of electrons with He^+*, Phys. Rev. A, 47 (1993), pp. R2442–R2445.
[11] E. Hairer, S. P. Nørsett, and G. Wanner, *Solving Ordinary Differential Equations* I, *Non-stiff Problems*, 2nd ed., Springer-Verlag, Berlin, New York, 1994.
[12] E. Hairer and G. Wanner, *Solving Ordinary Differential Equations* II, *Stiff and Differential-Algebraic Problems*, 2nd ed., Springer-Verlag, Berlin, New York, 1996.
[13] D. C. Heggie, *The N-body problem in stellar dynamics*, in Long-Term Dynamical Behavior of Natural and Artificial N-Body Systems, A. E. Roy, ed., NATO ASI Series, Kluwer, Dordrecht, the Netherlands, 1988.
[14] T. Holder, B. Leimkuhler, and S. Reich, *Explicit Variable Stepsize and Time-Reversible Integration*, preprint, 1998.
[15] W. Huang and B. Leimkuhler, *The adaptive Verlet method*, SIAM J. Sci. Comput., 18 (1997), pp. 239–256.
[16] P. Hut, Y. Funato, E. Kokubo, J. Makino, and S. McMillan, *Time symmetrization meta-algorithms*, in Computational Astrophysics, D. A. Clarke and M. J. West, eds., ASP Conf. Series, Proceedings of the 12th Kingston Meeting on Theoretical Astrophysics, Vol. 123, 1999, pp. 177–188.
[17] P. Hut, J. Makino, and S. McMillan, *Building a better leapfrog*, Astrophysical J., 443 (1995), pp. L93–L96.
[18] M. Klein and A. Knauf, *Classical Planar Scattering by Coulombic Potentials*, Lecture Notes in Phys. 13, Springer-Verlag, Berlin, 1992.
[19] L. D. Landau and E. M. Lifshitz, *Mechanics*, Course in Theoretical Physics, Vol. 1, 3rd ed., Pergamon Press, Oxford, 1976.
[20] B. Leimkuhler, *Reversible adaptive regularization: Perturbed Kepler motion and classical atomic trajectories*, R. Soc. London Philos. Trans. Ser. A Math. Phys. Engrg. Sci., 357 (1999), pp. 1101–1133.
[21] J. Marsden, *Lectures on Mechanics*, Cambridge University Press, Cambridge, UK, 1992.
[22] R. I. McLachlan, *On the numerical integration of ordinary differential equations by symmetric composition methods*, SIAM J. Sci. Comput., 16 (1995), pp. 151–168.
[23] S. Mikkola, *A practical and regular formulation of the N-body equations*, Mon. Not. R. Astr. Soc., 215 (1985), pp. 171–177.
[24] S. Mikkola, *A cubic approximation for Kepler's equation*, Celestial Mechanics and Dynamical Astronomy, 40 (1987), pp. 303–312.
[25] S. Mikkola, *Practical symplectic methods with time transformation for the few-body problem*, Celestial Mechanics and Dynamical Astronomy, 67 (1997), pp. 145–165.
[26] A. Nijerhuis, *Solving Kepler's equation with high efficiency and accuracy*, Celestial Mechanics and Dynamical Astronomy, 51 (1991), pp. 319–330.
[27] S. J. Peale, *Some unsolved problems in evolutionary dynamics*, Celestial Mechanics and Dynamical Astronomy, 46 (1989), pp. 253–275.
[28] P. R. Rauch and N. Holman, *Dynamical chaos in the Wisdom-Holman integrator: Origins and solutions*, Astronomical J., 117 (1999), pp. 1087–1102.
[29] K. Rzazewski, M. Lewenstein, and P. Salieres, *Multielectron stabilization of atoms in a laser field: Classical perspective*, Phys. Rev., A 49 (1994), pp. 1196–1201.
[30] J. M. Sanz-Serna and M. P. Calvo, *Numerical Hamiltonian Problems*, Chapman and Hall, London, 1994.
[31] R. D. Skeel and J. Biesiadecki, *Symplectic integration with variable stepsize*, Ann. Numer. Math., 1 (1994), pp. 191–198.
[32] J. Steckel and C. Jaffe, *The bifurcations of the Langmuir orbit in the two-electron atom*, in Hamiltonian Systems with Three or More Degrees of Freedom, NATO ASI Series, Kluwer, Dordrecht, the Netherlands, 1998.
[33] D. Stoffer, *Variable steps for reversible integration methods*, Computing, 55 (1995), pp. 1–22.
[34] M. Tuckerman, B. J. Berne, and G. J. Martyna, *Reversible multiple time scale molecular dynamics*, J. Chem. Phys., 97 (1992), pp. 1990–2001.
[35] L. Verlet, *Computer experiments on classical fluids* I. *Thermodynamical properties of*

lennard-jones molecules, Phys. Rev., 159 (1967), pp. 98–103.

[36] J. Waldvogel, *A new regularization of the planar problem of three bodies*, Celestial Mechanics, 6 (1972), pp. 221–231.

[37] J. Wisdom and M. Holman, *Symplectic maps for the N-body problem*, Astronomical J., 102 (1991), pp. 1528–1538.

[38] H. Yoshida, *Construction of higher order symplectic integrators*, Phys. Lett., A 150 (1990), pp. 262–268.

[39] K. Zare and V. Szebehely, *Time transformations for the extended phase space*, Celestial Mechanics, 11 (1975), pp. 219–227.

SIAM J. SCI. COMPUT.
Vol. 22, No. 3, pp. 1036–1052
© 2000 Society for Industrial and Applied Mathematics

NEW FAST ALGORITHMS FOR ERROR RATE-BASED STEPWISE VARIABLE SELECTION IN DISCRIMINANT ANALYSIS*

S. AEBERHARD†, O. Y. DE VEL‡, AND D. H. COOMANS§

Abstract. Variable selection is an important technique for reducing the dimensionality in multivariate predictive discriminant analysis and classification. In the past, direct evaluation of the subsets by means of a classifier has been computationally too expensive, rendering necessary the use of heuristic measures of class separation, such as Wilk's Λ or the Mahalanobis distance between class means. We present new fast algorithms for stepwise variable selection based on quadratic and linear classifiers with time complexities which, to within a constant, are the same as those applying measures of class separation. Comparing the new algorithms to previous implementations of classifier-based variable selection, we show that dramatic speed-ups are achieved.

Key words. dimensionality reduction, classification, classifier-based variable selection

AMS subject classifications. 68W40, 65Y20

PII. S1064827596300784

1. Introduction.

1.1. Discriminant analysis and variable selection. Consider a classification problem with a total of N_p observation vectors $\mathbf{x} = (x_1, x_2, \ldots, x_d)$, sampled from K groups or classes. The aim of *predictive* discriminant analysis is to construct a rule for the allocation of future observations to one or several of the K classes. Linear and quadratic discriminant analysis is the name given to two rules constructed under the assumption that the data are multivariate normally distributed. The former assumes equal, the latter unequal, covariance matrices. *Descriptive* discriminant analysis investigates how the groups differ, which can also be indirectly assessed with the predictive approach. This paper focuses on stepwise variable selection applied in conjunction with predictive discriminant analysis, also called *classification*.

The benefits of variable subset selection differ depending on the ultimate aim of the underlying investigation. If the data at hand are from a preliminary study, a major consideration may be the reduction of the cost in sampling future data by a decreased number of variables. If the aim of the investigation is to analyze class differences, variables selection may lead to a more parsimonious description. If the goal is that of allocating future unknown observations to one or several of the K classes, variable subset selection may increase the discriminant power of the classification method [16]. As the potential benefits increase with increased numbers of variables, variable selection is particularly useful in high-dimensional settings. For a more general overview of discriminant analysis refer to [3] or [10].

*Received by the editors March 18, 1996; accepted for publication (in revised form) September 7, 1997; published electronically September 27, 2000. This research was supported by Australian Research Council grants ARC145033 and ARC3903.

http://www.siam.org/journals/sisc/22-3/30078.html

†Department of Computer Science, James Cook University, Townsville, QLD 4811, Australia (stefan@cs.jcu.edu.au).

‡Information Technology Division, Defence Science and Technology Organisation, P.O. Box 1500, Salisbury SA 5108, Australia (olivier.devel@dsto.defence.gov.au).

§Department of Mathematics and Statistics, James Cook University, Townsville, QLD 4811, Australia (danny.coomans@jcu.edu.au).

1.2. Variable selection. Variable selection is essentially a search problem over all combinations of variables, consisting of two main components:

- a search procedure used to determine the variable subsets that are evaluated;
- a criterion $J(\mathcal{V}^{(l)})$ used to evaluate which of the subsets is "best," where $\mathcal{V}^{(l)}$ denotes a variable subset of size l, $1 \leq l \leq d$. Examples used in this paper are the error rate of linear and quadratic discriminant rules and class separability measures.

1.2.1. Search heuristics. There are $2^d - 1$ distinct subsets of d variables, rendering an exhaustive search computationally infeasible for all problems except those with low dimensionality. However, it is still possible to efficiently find the optimal subset using the branch-and-bound algorithm if the criterion $J(\mathcal{V}^{(l)})$ is monotonic [11]. If $J(\mathcal{V}^{(l)})$ is not monotonic, such as the error rate of classifiers, heuristic search strategies have to be used. Among these, the stepwise procedures are popular; variables are either selected in a forward manner, sequentially deleted in the backward elimination approach, or selected by a combination of the two [11].

1.2.2. Selection criteria $J(\mathcal{V}^{(l)})$. In predictive discriminant analysis, assuming equal cost of misclassification, one possible optimal criterion $J(\mathcal{V}^{(l)})$ to assess the subsets is the error rate. However, even the relatively simple stepwise procedures consider too many subsets to be computationally feasible for medium- to high-dimensional problems, as obtaining the error rates is an expensive procedure in itself. Consequently, there have been relatively few reports in this area; [5] and [7] support the use of error rates; however, their experiments had to be limited to data with relatively few dimensions. For these cost reasons, error rates are usually replaced with measures of class separation, which can be computed much more efficiently. Two commonly used measures are Wilk's Λ and the Mahalanobis distance between the two nearest class means (see [8], [3] for more details). Another useful measure is the Lawley–Hotelling trace sometimes used in the multivariate analysis of variance [9]. However, the computational advantage of these criteria are traded off by several disadvantages [13]:

- Variable selection is terminated based on the application of a test statistic to the values of $J(\mathcal{V}^{(l)})$ at adjacent levels. The number of variables that are included depends on the critical value of the statistic, and no general rule for their determination is known.
- Because the variable that maximizes the increase in $J(\mathcal{V}^{(l)})$ (forward selection) is selected at each step, the actual significance levels are unknown.
- Maximizing class separability does not necessarily minimize error rates. This is especially true for problems with more than two classes.

Simpler approaches to variable selection choose the variables based on canonical variate analysis or based on ranking of the variables under the assumption of their mutual independence. However, these methods are less powerful and have been criticized in the comprehensive review by McKay and Campbell [13].

1.2.3. Classifier performance evaluation. Variable selection based on classifier performance requires the evaluation of the classifier. Several sample-based evaluation techniques are in common use, but only the resubstitution and cross validation methods are computationally suitable in this context [1]. Further, because variable selection is often performed with ill- or poorly posed problems, the optimistic bias of the resubstitution method precludes its use. An improvement to the bias can be made using Snapinn and Knoke's smoothed resubstitution method for two classes [17]

and the multiclass extension proposed by Hirst [9]. Among the cross-validation techniques, the leave-one-out method [12] has the advantage of being nearly unbiased and computationally efficient through the use of fast downdating formulas (see [2], [4]). One drawback of the leave-one-out method is its large variance, which, in combination with the many subsets investigated by the search procedure, unfortunately leads to optimistically biased performance estimates. However, similar bias also occurs when the selection is based on measures of class separation rather than error rates [15].

A straightforward implementation of stepwise variable selection based on the performance of linear discriminant analysis (LDA) or quadratic discriminant analysis (QDA) has $O(N_p d^4)$ time complexity,[1] where N_p is the total number of observations and d is the dimensionality, i.e., the total number of variables. If downdating formulas are used, a single leave-one-out evaluation of QDA or LDA has $O(N_p d^2)$ time complexity. At each iteration of stepwise variable selection, the number of variables considered is of order $O(d)$. As the number of iterations is also of order $O(d)$, the overall time complexity is $O(N_p d^4)$, if implemented in a straightforward manner. However, it is possible to reduce that complexity by a combined application of two sets of downdating formulas. The resulting new algorithms have a time complexity of $O(N_p d^2)$, which is optimal and the same as that for a single discriminant analysis.

In this paper we bring some known results together with some new results for stepwise variable selection into a set of streamlined and efficient algorithms. We derive the algorithm for forward selection in section 2, followed by a summary of the corresponding formulas for backward elimination. The fourth and last section reports on run-time comparisons of the new algorithms with previous implementations and with two approaches that use measures of class separation.

2. New $O(N_p d^2)$ algorithms for forward selection based on the performance of QDA and LDA. With the forward selection approach, variables are sequentially included into $\mathcal{V}^{(l)}$ until further inclusion reduces the estimate of the classification accuracy, or until $\mathcal{V}^{(l)}$ reaches a specified size. This stopping criterion was chosen for its simplicity and for its easy implementation. However, the user may choose a more complex stopping criterion as this will not affect the variable selection algorithm. In order to achieve the optimal time complexity $O(N_p d^2)$, the combination of two sets of downdating formulas is required. The first set allows the efficient computation of the inverse and determinant of the covariance matrix in the case where one variable is added to, or deleted from, the current set of variables $\mathcal{V}^{(l)}$. The second set of formulas are applied in the context of the leave-one-out estimation. Before deriving the algorithms, we present these two sets of formulas.

2.1. Adding or deleting one variable from $\mathcal{V}^{(l)}$. The formulas below are used to efficiently calculate the inverse and determinant of the sample covariance matrix when one variable is added to, or deleted from, $\mathcal{V}^{(l)}$.

Consider the case where variable j is added to $\mathcal{V}^{(l)}$, and assume without loss of generality that the sample covariance matrix $\mathbf{S}_k^{(l+1)}$ of class ω_k at level $l+1$ is constructed from the sample covariance matrix $\mathbf{S}_k^{(l)}$ at level l as follows:

$$\mathbf{S}_k^{(l+1)} = \begin{bmatrix} \mathbf{S}_k^{(l)} & \mathbf{d}_{k,j} \\ \mathbf{d}_{k,j}^T & s_{k,j,j} \end{bmatrix}, \tag{2.1}$$

[1] A function $f(n)$ is said to be $O(g(n))$ if there are positive constants K and n_0 such that $\forall n \geq n_0$, $f(n) \leq Kg(n)$.

where $[\mathbf{d}_{k,j}^T\ ,\ s_{k,j,j}]$ is the row corresponding to the newly included variable j. Then $\overset{(l+1)}{\mathbf{S}_k^{-1}}$ is given in terms of $\overset{(l)}{\mathbf{S}_k^{-1}}$ by (see [6])

$$\overset{(l+1)}{\mathbf{S}_k^{-1}} = \begin{bmatrix} \overset{(l)}{\mathbf{S}_k^{-1}} + \frac{1}{a_{k,j}}\overset{(l)}{\mathbf{S}_k^{-1}}\mathbf{d}_{k,j}\mathbf{d}_{k,j}^T\overset{(l)}{\mathbf{S}_k^{-1}} & -\frac{1}{a_{k,j}}\overset{(l)}{\mathbf{S}_k^{-1}}\mathbf{d}_{k,j} \\ -\frac{1}{a_{k,j}}\mathbf{d}_{k,j}^T\overset{(l)}{\mathbf{S}_k^{-1}} & \frac{1}{a_{k,j}} \end{bmatrix}, \tag{2.2}$$

where

$$a_{k,j} = s_{k,j,j} - \mathbf{d}_{k,j}^T\overset{(l)}{\mathbf{S}_k^{-1}}\mathbf{d}_{k,j}. \tag{2.3}$$

For the case where variable j is deleted from $\mathcal{V}^{(l)}$, assume without loss of generality that variable j corresponds to the last row and column in $\overset{(l)}{\mathbf{S}_k^{-1}}$, which we then write as

$$\overset{(l)}{\mathbf{S}_k^{-1}} = \begin{bmatrix} \mathbf{A} & \mathbf{c}_j \\ \mathbf{c}_j^T & b_j \end{bmatrix}. \tag{2.4}$$

$\overset{(l-1)}{\mathbf{S}_k^{-1}}$ can then efficiently be computed from $\overset{(l)}{\mathbf{S}_k^{-1}}$ by (see [6])

$$\overset{(l-1)}{\mathbf{S}_k^{-1}} = \mathbf{A} - \frac{1}{b_j}\mathbf{c}_j\mathbf{c}_j^T. \tag{2.5}$$

The ratio of the determinants at adjacent levels is the diagonal element of $\overset{(l)}{\mathbf{S}_k^{-1}}$ corresponding to variable j,

$$\left|\overset{(l)}{\mathbf{S}_k^{-1}}\right| = b_j \left|\overset{(l-1)}{\mathbf{S}_k^{-1}}\right|. \tag{2.6}$$

Then, using (2.4) and (2.5), the following equation for a quadratic form is easily derived (see [1]):

$$\mathbf{x}_1^{(l-1)T}\overset{(l-1)}{\mathbf{S}_k^{-1}}\mathbf{x}_2^{(l-1)} = \mathbf{x}_1^{(l)T}\overset{(l)}{\mathbf{S}_k^{-1}}\mathbf{x}_2^{(l)} - \frac{1}{b_j}\left(\mathbf{x}_1^{(l-1)T}\mathbf{c}_j + b_j\mathbf{x}_{1,j}\right)\left(\mathbf{c}_j^T\mathbf{x}_2^{(l-1)} + b_j\mathbf{x}_{2,j}\right). \tag{2.7}$$

Here, $\mathbf{x}_1^{(l)}$ and $\mathbf{x}_2^{(l)}$ are two column vectors of length l, $\mathbf{x}_{1,j}$ and $\mathbf{x}_{2,j}$ are their jth element, and $\overset{(l)}{\mathbf{S}_k^{-1}}$, $\mathbf{c}_j$, and b_j are defined by (2.4). Equation (2.7) is used at a later stage for the derivation of the algorithms; it allows the efficient computation of a quadratic form given its value with one extra variable included or excluded.

2.2. Leaving one observation out. The following, second set of formulas is used to downdate the class sample mean vector $\mathbf{x}_{\bullet k}^{(l)}$, $\overset{(l)}{\mathbf{S}_k^{-1}}$, and $|\overset{(l)}{\mathbf{S}_k^{-1}}|$ in the case where observation $\mathbf{x}_i$ is omitted from the training set. Let ω_λ be the class to which $\mathbf{x}_i$ belongs, and let N_λ be the number of training observations in ω_λ. The subscript "$\backslash i$" indicates that the corresponding quantity is computed with observation $\mathbf{x}_i$ left out.

The formulas for QDA are given by (see [2])

$$\mathbf{x}_{\bullet\,\lambda\setminus i}^{(l)} = \mathbf{x}_{\bullet\,\lambda}^{(l)} - \frac{1}{N_\lambda - 1}(\mathbf{x}_i^{(l)} - \mathbf{x}_{\bullet\,\lambda}^{(l)}), \tag{2.8}$$

$$\mathbf{S}_{\lambda\setminus i}^{-1\,(l)} = \frac{N_\lambda - 2}{N_\lambda - 1}\left[\mathbf{S}_\lambda^{-1\,(l)} + \frac{N_\lambda \mathbf{S}_\lambda^{-1\,(l)}(\mathbf{x}_i^{(l)} - \mathbf{x}_{\bullet\,\lambda}^{(l)})(\mathbf{x}_i^{(l)} - \mathbf{x}_{\bullet\,\lambda}^{(l)})^T \mathbf{S}_\lambda^{-1\,(l)}}{(N_\lambda - 1)^2 - N_\lambda(\mathbf{x}_i^{(l)} - \mathbf{x}_{\bullet\,\lambda}^{(l)})^T \mathbf{S}_\lambda^{-1\,(l)}(\mathbf{x}_i^{(l)} - \mathbf{x}_{\bullet\,\lambda}^{(l)})}\right], \tag{2.9}$$

$$\left|\mathbf{S}_{\lambda\setminus i}^{(l)}\right| = \left(\frac{N_\lambda - 1}{N_\lambda - 2}\right)^l \left|\mathbf{S}_\lambda^{(l)}\right| \left[1 - \frac{N_\lambda}{(N_\lambda - 1)^2}(\mathbf{x}_i^{(l)} - \mathbf{x}_{\bullet\,\lambda}^{(l)})^T \mathbf{S}_\lambda^{-1\,(l)}(\mathbf{x}_i^{(l)} - \mathbf{x}_{\bullet\,\lambda}^{(l)})\right]. \tag{2.10}$$

In the case of LDA, the downdating formula for the mean vector is the same. Furthermore, as LDA uses the *pooled* covariance matrix $\mathbf{S}_p^{(l)}$, the determinant $\left|\mathbf{S}_p^{(l)}\right|$ is the same for all classes and is hence omitted. The formula for downdating the inverse $\mathbf{S}_p^{-1\,(l)}$ of the pooled covariance matrix is (see [1])

$$\begin{aligned}\mathbf{S}_{p\setminus i}^{-1\,(l)} &= \frac{N_p - K - 1}{N_p - K} \\ &\times \left[\mathbf{S}_p^{-1\,(l)} + \frac{N_\lambda \mathbf{S}_p^{-1\,(l)}(\mathbf{x}_i^{(l)} - \mathbf{x}_{\bullet\,\lambda}^{(l)})(\mathbf{x}_i^{(l)} - \mathbf{x}_{\bullet\,\lambda}^{(l)})^T \mathbf{S}_p^{-1\,(l)}}{(N_p - K)(N_\lambda - 1) - N_\lambda(\mathbf{x}_i^{(l)} - \mathbf{x}_{\bullet\,\lambda}^{(l)})^T \mathbf{S}_p^{-1\,(l)}(\mathbf{x}_i^{(l)} - \mathbf{x}_{\bullet\,\lambda}^{(l)})}\right],\end{aligned} \tag{2.11}$$

where N_p is the total number of observations.

2.3. Forward selection based on QDA. In the forward selection step, given l variables already included, one seeks to find the variable that, when added to $\mathcal{V}^{(l)}$, maximizes the classification performance of QDA. This is done by tentatively including each prospective variable and estimating the resulting error rate. For each of these estimates, we need to compute the K class discriminant scores for all N_p observations. The algorithm halts when further addition of variables increases the error rate estimate. The discriminant or classification score is the outcome of the discriminant function for a particular data vector.

2.3.1. Discriminant score for class of object $\mathbf{x}_i$ left out. Let the object left out be $\mathbf{x}_i$, and let j be the variable considered for inclusion. We want to compute the score

$$f_{\lambda,\setminus i,j}^{(l+1)} = (\mathbf{x}_i^{(l+1)} - \mathbf{x}_{\bullet\,\lambda\setminus i}^{(l+1)})^T \mathbf{S}_{\lambda\setminus i}^{-1\,(l+1)}(\mathbf{x}_i^{(l+1)} - \mathbf{x}_{\bullet\,\lambda\setminus i}^{(l+1)}) + \ln\left|\mathbf{S}_{\lambda\setminus i}^{(l+1)}\right| - 2\ln\left(P(\omega_\lambda)\right), \quad j \notin \mathcal{V}^{(l)}, \tag{2.12}$$

for observation $\mathbf{x}_i$, for its class ω_λ, with variable j added to $\mathcal{V}^{(l)}$ (i.e., $\mathcal{V}^{(l+1)} = \mathcal{V}^{(l)} \cup j$). Writing $\mathbf{S}_{\lambda\setminus i}^{-1\,(l+1)}$ as

$$\mathbf{S}_{\lambda\setminus i}^{-1\,(l+1)} = \begin{bmatrix} \mathbf{A} & \mathbf{c}_j \\ \mathbf{c}_j^T & b_j \end{bmatrix} \tag{2.13}$$

and applying (2.7), we find that the first term in (2.12) becomes

$$\begin{aligned}(\mathbf{x}_i^{(l+1)} - \mathbf{x}_{\bullet\,\lambda\setminus i}^{(l+1)})^T \mathbf{S}_{\lambda\setminus i}^{-1\,(l+1)}(\mathbf{x}_i^{(l+1)} - \mathbf{x}_{\bullet\,\lambda\setminus i}^{(l+1)}) &= (\mathbf{x}_i^{(l)} - \mathbf{x}_{\bullet\,\lambda\setminus i}^{(l)})^T \mathbf{S}_{\lambda\setminus i}^{-1\,(l)}(\mathbf{x}_i^{(l)} - \mathbf{x}_{\bullet\,\lambda\setminus i}^{(l)}) \\ &+ \frac{1}{b_j}\left\{[\mathbf{c}_j^T\ ,\ b_j](\mathbf{x}_i^{(l+1)} - \mathbf{x}_{\bullet\,\lambda\setminus i}^{(l+1)})\right\}^2.\end{aligned} \tag{2.14}$$

Note that the variable j left out corresponds to the last row of $\mathbf{S}_{\lambda\setminus i}^{-1\,(l+1)}$ for notational convenience only. From (2.8) and (2.9), the first term on the right-hand side of (2.14) becomes

$$(\mathbf{x}_i^{(l)} - \mathbf{x}_{\bullet\lambda\setminus i}^{(l)})^T \mathbf{S}_{\lambda\setminus i}^{-1\,(l)} (\mathbf{x}_i^{(l)} - \mathbf{x}_{\bullet\lambda\setminus i}^{(l)}) = \frac{N_\lambda^2 (N_\lambda - 2)}{(N_\lambda - 1)^3} \left[(\mathbf{x}_i^{(l)} - \mathbf{x}_{\bullet\lambda}^{(l)})^T \mathbf{S}_\lambda^{-1\,(l)} (\mathbf{x}_i^{(l)} - \mathbf{x}_{\bullet\lambda}^{(l)}) \right.$$

$$\left. + \frac{N_\lambda (\mathbf{x}_i^{(l)} - \mathbf{x}_{\bullet\lambda}^{(l)})^T \mathbf{S}_\lambda^{-1\,(l)} (\mathbf{x}_i^{(l)} - \mathbf{x}_{\bullet\lambda}^{(l)}) (\mathbf{x}_i^{(l)} - \mathbf{x}_{\bullet\lambda}^{(l)})^T \mathbf{S}_\lambda^{-1\,(l)} (\mathbf{x}_i^{(l)} - \mathbf{x}_{\bullet\lambda}^{(l)})}{(N_\lambda - 1)^2 - N_\lambda (\mathbf{x}_i^{(l)} - \mathbf{x}_{\bullet\lambda}^{(l)})^T \mathbf{S}_\lambda^{-1\,(l)} (\mathbf{x}_i^{(l)} - \mathbf{x}_{\bullet\lambda}^{(l)})} \right]. \tag{2.15}$$

For computational simplicity, the same ordering of the rows of $\mathbf{S}_\lambda^{-1\,(l)}$ as for the variables themselves is kept. Because the jth row of $\mathbf{S}_\lambda^{-1\,(l)}$ will not generally correspond to variable j, we index the row of $\mathbf{S}_\lambda^{-1\,(l)}$ corresponding to the variable j as j^*. Focusing on the second term in (2.14), we see that $[\mathbf{c}_j^T \ , \ b_j]$ is the row of $\mathbf{S}_{\lambda\setminus i}^{-1\,(l+1)}$ corresponding to the newly included variable. Using (2.8) and (2.9),

$$[\mathbf{c}_j^T, b_j](\mathbf{x}_i^{(l+1)} - \mathbf{x}_{\bullet\lambda\setminus i}^{(l+1)}) = \frac{N_\lambda}{N_\lambda - 1} \left[\mathbf{S}_{\lambda\setminus i}^{-1\,(l+1)} \right]_{j^*} (\mathbf{x}_i^{(l+1)} - \mathbf{x}_{\bullet\lambda}^{(l+1)})$$

$$= \frac{N_\lambda (N_\lambda - 2)}{(N_\lambda - 1)^2} \left[\left[\mathbf{S}_\lambda^{-1\,(l+1)} \right]_{j^*} (\mathbf{x}_i^{(l+1)} - \mathbf{x}_{\bullet\lambda}^{(l+1)}) \right.$$

$$\left. + \frac{N_\lambda \left[\mathbf{S}_\lambda^{-1\,(l+1)} \right]_{j^*} (\mathbf{x}_i^{(l+1)} - \mathbf{x}_{\bullet\lambda}^{(l+1)}) (\mathbf{x}_i^{(l+1)} - \mathbf{x}_{\bullet\lambda}^{(l+1)})^T \mathbf{S}_\lambda^{-1\,(l+1)} (\mathbf{x}_i^{(l+1)} - \mathbf{x}_{\bullet\lambda}^{(l+1)})}{(N_\lambda - 1)^2 - N_\lambda (\mathbf{x}_i^{(l+1)} - \mathbf{x}_{\bullet\lambda}^{(l+1)})^T \mathbf{S}_\lambda^{-1\,(l+1)} (\mathbf{x}_i^{(l+1)} - \mathbf{x}_{\bullet\lambda}^{(l+1)})} \right]. \tag{2.16}$$

Similarly, b_j in (2.14) becomes

$$b_j = \left[\mathbf{S}_{\lambda\setminus i}^{-1\,(l+1)} \right]_{j^*,j^*}$$

$$= \frac{N_\lambda - 2}{N_\lambda - 1} \left[\left[\mathbf{S}_\lambda^{-1\,(l+1)} \right]_{j^*,j^*} + \frac{N_\lambda \left[\mathbf{S}_\lambda^{-1\,(l+1)} \right]_{j^*} (\mathbf{x}_i^{(l+1)} - \mathbf{x}_{\bullet\lambda}^{(l+1)}) (\mathbf{x}_i^{(l+1)} - \mathbf{x}_{\bullet\lambda}^{(l+1)})^T \left[\mathbf{S}_\lambda^{-1\,(l+1)} \right]_{j^*}^T}{(N_\lambda - 1)^2 - N_\lambda (\mathbf{x}_i^{(l+1)} - \mathbf{x}_{\bullet\lambda}^{(l+1)})^T \mathbf{S}_\lambda^{-1\,(l+1)} (\mathbf{x}_i^{(l+1)} - \mathbf{x}_{\bullet\lambda}^{(l+1)})} \right]. \tag{2.17}$$

We also need to downdate the determinant term in (2.12). Using (2.6) and (2.10), we find that

$$\left| \mathbf{S}_{\lambda\setminus i}^{(l+1)} \right| = \left(\frac{N_\lambda - 1}{N_\lambda - 2} \right)^{l+1} a_{\lambda,j} \left| \mathbf{S}_\lambda^{(l)} \right|$$

$$\times \left(1 - \frac{N_\lambda}{(N_\lambda - 1)^2} (\mathbf{x}_i^{(l+1)} - \mathbf{x}_{\bullet\lambda}^{(l+1)})^T \mathbf{S}_\lambda^{-1\,(l+1)} (\mathbf{x}_i^{(l+1)} - \mathbf{x}_{\bullet\lambda}^{(l+1)}) \right), \tag{2.18}$$

where $a_{\lambda,j}$ is defined by (2.2) and (2.3).

In order to simplify the above expressions, we define

$$m^{(l)}_{k,t,i:j} = (\mathbf{x}_i^{(l+1)} - \mathbf{x}_{\bullet k}^{(l+1)})^T \overset{(l+1)}{\mathbf{S}_k^{-1}} (\mathbf{x}_i^{(l+1)} - \mathbf{x}_{\bullet t}^{(l+1)}), \qquad j \notin \mathcal{V}^{(l)}, \;\; k,t = 1,\ldots,K. \tag{2.19}$$

If $k = t$, as is always the case for formulas associated with QDA, $m^{(l)}_{k,k,i:j}$ is the Mahalanobis distance between $\mathbf{x}_i$ and $\mathbf{x}_{\bullet k}^{(l+1)}$ at level $l+1$ if the variable j is added to $\mathcal{V}^{(l)}$. Also, let

$$p^{(l)}_{k,i,j} = \left[\overset{(l+1)}{\mathbf{S}_k^{-1}}\right]_{j^*} (\mathbf{x}_i^{(l+1)} - \mathbf{x}_{\bullet k}^{(l+1)}), \qquad j \notin \mathcal{V}^{(l)}, \;\; k = 1,\ldots,K; \tag{2.20}$$

i.e., $p^{(l)}_{k,i,j}$ is the inner product of the row $\left[\overset{(l+1)}{\mathbf{S}_k^{-1}}\right]_{j^*}$ corresponding to the variable j added to $\mathcal{V}^{(l)}$, and $(\mathbf{x}_i^{(l+1)} - \mathbf{x}_{\bullet k}^{(l+1)})$. Note that the index l in $m^{(l)}_{k,k,i:j}$ and $p^{(l)}_{k,i,j}$ denotes the level at which these quantities are computed, and not the length of the vectors involved, which is $l+1$.

We further define

$$\eta^{(l)}_{k,t,i} \equiv m^{(l-1)}_{k,t,i:q} = (\mathbf{x}_i^{(l)} - \mathbf{x}_{\bullet k}^{(l)})^T \overset{(l)}{\mathbf{S}_k^{-1}} (\mathbf{x}_i^{(l)} - \mathbf{x}_{\bullet t}^{(l)}), \tag{2.21}$$

where variable q was included at level $l-1$. Again, $k = t$ for QDA, and the $\eta^{(l)}_{k,k,i}$ are the $K \times N_p$ Mahalanobis distances between the N_p objects and the K mean vectors at the present level l.

Combining (2.14)–(2.21), and substituting them into (2.12), we see that the leave-one-out classification score for the class of $\mathbf{x}_i$ becomes

$$\begin{aligned} f^{(l+1)}_{\lambda,\setminus i,j} &= \frac{N_\lambda^2(N_\lambda-2)}{(N_\lambda-1)^3}\left[\eta^{(l)}_{\lambda,\lambda,i} + \frac{N_\lambda\left(\eta^{(l)}_{\lambda,\lambda,i}\right)^2}{(N_\lambda-1)^2 - N_\lambda \eta^{(l)}_{\lambda,\lambda,i}}\right] \\ &+ \frac{N_\lambda^2(N_\lambda-2)}{(N_\lambda-1)^3}\left(\frac{1}{a_{\lambda,j}} + \frac{N_\lambda\left(p^{(l)}_{\lambda,i,j}\right)^2}{(N_\lambda-1)^2 - N_\lambda m^{(l)}_{\lambda,\lambda,i:j}}\right)^{-1}\left(p^{(l)}_{\lambda,i,j} + \frac{N_\lambda p^{(l)}_{\lambda,i,j} m^{(l)}_{\lambda,\lambda,i:j}}{(N_\lambda-1)^2 - N_\lambda m^{(l)}_{\lambda,\lambda,i:j}}\right)^2 \\ &+ \ln\left[a_{\lambda,j}\left(\frac{N_\lambda-1}{N_\lambda-2}\right)^{l+1}\left|\mathbf{S}^{(l)}_\lambda\right|\left(1 - \frac{N_\lambda m^{(l)}_{\lambda,\lambda,i:j}}{(N_\lambda-1)^2}\right)\right] - 2\ln\left(P(\omega_\lambda)\right). \end{aligned} \tag{2.22}$$

2.3.2. Score for classes other than those of object $\mathbf{x}_i$. Equation (2.22) gives the score for object $\mathbf{x}_i$ left out for the class ω_λ of $\mathbf{x}_i$. Similar to (2.12), the leave-one-out score of $\mathbf{x}_i$ for classes other than its own is written as

$$f^{(l+1)}_{k,\setminus i,j} = (\mathbf{x}_i^{(l+1)} - \mathbf{x}_{\bullet k}^{(l+1)})^T \overset{(l+1)}{\mathbf{S}_k^{-1}} (\mathbf{x}_i^{(l+1)} - \mathbf{x}_{\bullet k}^{(l+1)}) + \ln\left|\mathbf{S}^{(l+1)}_k\right| - 2\ln\left(P(\omega_k)\right) \;. \tag{2.23}$$

Note that $\overset{(l+1)}{\mathbf{S}_k^{-1}} = \overset{(l+1)}{\mathbf{S}_{k\setminus i}^{-1}}$ since $\mathbf{x}_i$ belongs to class ω_k, $k \neq \lambda$.

We assume again that j is the newly included variable and thus write $\mathbf{S}^{(l+1)}_k$ as follows:

$$\overset{(l+1)}{\mathbf{S}_k^{-1}} = \begin{bmatrix} \mathbf{A} & \mathbf{c}_j \\ \mathbf{c}_j^T & b_j \end{bmatrix}. \tag{2.24}$$

Then, using (2.6) and (2.7), (2.23) becomes

$$f_{k,\setminus i,j}^{(l+1)} = (\mathbf{x}_i^{(l)} - \mathbf{x}_{\bullet k}^{(l)})^T \mathbf{S}_k^{-1(l)} (\mathbf{x}_i^{(l)} - \mathbf{x}_{\bullet k}^{(l)}) + b_j^{-1} \left\{ [\mathbf{c}_j^T, b_j] (\mathbf{x}_i^{(l+1)} - \mathbf{x}_{\bullet k}^{(l+1)}) \right\}^2$$

$$+ \ln \left[b_j^{-1} \left| \mathbf{S}_k^{(l)} \right| \right] - 2 \ln \left(P(\omega_k) \right). \tag{2.25}$$

Applying (2.20) and (2.21) to (2.25), and using $b_j^{-1} = a_{k,j}$, we have that

$$f_{k,\setminus i,j}^{(l+1)} = \eta_{k,k,i}^{(l)} + a_{k,j} (p_{k,i,j}^{(l)})^2 + \ln \left(a_{k,j} \left| \mathbf{S}_k^{(l)} \right| \right) - 2 \ln \left(P(\omega_k) \right). \tag{2.26}$$

The above result is the score for classes other than the one to which $\mathbf{x}_i$ belongs.

In order for the algorithm to be efficient, we need to be able to efficiently precompute $p_{k,i,j}^{(l)}$ and $m_{k,k,i:j}^{(l)}$ at each level. In the next section, we derive formulas that allow these quantities to be computed in constant time from their previous values at the level below.

2.3.3. The computational scheme. Let

$$\theta_{k,i,j}^{(l)} = \mathbf{d}_{k,j}^T \mathbf{S}_k^{-1(l)} (\mathbf{x}_i^{(l)} - \mathbf{x}_{\bullet k}^{(l)}), \qquad j \notin \mathcal{V}^{(l)}, \tag{2.27}$$

$$\Delta_{k,i,j} = (\mathbf{x}_{i,j} - \mathbf{x}_{\bullet k,j}), \qquad j \notin \mathcal{V}^{(l)}, \tag{2.28}$$

where $\mathbf{d}_{k,j}$ is given by (2.1). Considering $p_{k,i,j}^{(l)}$, from (2.2) and (2.20),

$$\begin{aligned} p_{k,i,j}^{(l)} &= \left[-\frac{1}{a_{k,j}} \mathbf{d}_{k,j}^T \mathbf{S}_k^{-1(l)}, \frac{1}{a_{k,j}} \right] \left(\left[\mathbf{x}_i^{(l)T}, \mathbf{x}_{i,j} \right]^T - \left[\mathbf{x}_{\bullet k}^{(l)T}, \mathbf{x}_{\bullet k,j} \right]^T \right) \\ &= -\frac{1}{a_{k,j}} \mathbf{d}_{k,j}^T \mathbf{S}_k^{-1(l)} (\mathbf{x}_i^{(l)} - \mathbf{x}_{\bullet k}^{(l)}) + \frac{1}{a_{k,j}} \left(\mathbf{x}_{i,j} - \mathbf{x}_{\bullet k,j} \right) \\ &= -\frac{1}{a_{k,j}} \left(\theta_{k,i,j}^{(l)} - \Delta_{k,i,j} \right). \end{aligned} \tag{2.29}$$

Working with $m_{\lambda,\lambda,i:j}^{(l)}$, from (2.2) and (2.19), we have that

$$\begin{aligned} m_{k,k,i:j}^{(l)} &= \left(\left[\mathbf{x}_i^{(l)T}, \mathbf{x}_{i,j} \right]^T - \left[\mathbf{x}_{\bullet k}^{(l)T}, \mathbf{x}_{\bullet k,j} \right]^T \right)^T \\ &\quad \times \begin{bmatrix} \mathbf{S}_k^{-1(l)} + \frac{1}{a_{k,j}} \mathbf{S}_k^{-1(l)} \mathbf{d}_{k,j} \mathbf{d}_{k,j}^T \mathbf{S}_k^{-1(l)} & -\frac{1}{a_{k,j}} \mathbf{S}_k^{-1(l)} \mathbf{d}_{k,j} \\ -\frac{1}{a_{k,j}} \mathbf{d}_{k,j}^T \mathbf{S}_k^{-1(l)} & \frac{1}{a_{k,j}} \end{bmatrix} \\ &\quad \times \left(\left[\mathbf{x}_i^{(l)T}, \mathbf{x}_{i,j} \right]^T - \left[\mathbf{x}_{\bullet k}^{(l)T}, \mathbf{x}_{\bullet k,j} \right]^T \right) \\ &= (\mathbf{x}_i^{(l)} - \mathbf{x}_{\bullet k}^{(l)})^T \mathbf{S}_k^{-1(l)} (\mathbf{x}_i^{(l)} - \mathbf{x}_{\bullet k}^{(l)}) \\ &\qquad + \frac{1}{a_{k,j}} (\mathbf{x}_i^{(l)} - \mathbf{x}_{\bullet k}^{(l)})^T \mathbf{S}_k^{-1(l)} \mathbf{d}_{k,j} \mathbf{d}_{k,j}^T \mathbf{S}_k^{-1(l)} (\mathbf{x}_i^{(l)} - \mathbf{x}_{\bullet k}^{(l)}) \\ &\qquad - \frac{2}{a_{k,j}} (\mathbf{x}_{i,j} - \mathbf{x}_{\bullet k,j}) (\mathbf{x}_i^{(l)} - \mathbf{x}_{\bullet k}^{(l)})^T \mathbf{S}_k^{-1(l)} \mathbf{d}_{k,j} + \frac{1}{a_{k,j}} (\mathbf{x}_{i,j} - \mathbf{x}_{\bullet k,j})^2 \\ &= \eta_{k,k,i}^{(l)} + \frac{1}{a_{k,j}} \left({\theta_{k,i,j}^{(l)}}^2 - 2 \Delta_{k,i,j} \theta_{k,i,j}^{(l)} + \Delta_{k,i,j}^2 \right). \end{aligned} \tag{2.30}$$

Hence we can efficiently compute the $m^{(l+1)}_{k,k,i:j}$ and $p^{(l+1)}_{k,i,j}$ if the $\theta^{(l+1)}_{k,i,j}$ are precomputed. This is done using fast updating formulas derived below.

Let

$$r^{(l)}_{k,j,h} = \mathbf{d}^T_{k,j} \overset{(l)}{\mathbf{S}}{}^{-1}_k \mathbf{d}_h, \qquad j \notin \mathcal{V}^{(l)}. \tag{2.31}$$

In order to consider $\theta^{(l+1)}_{k,i,j}$, we assume that the variable q was included at level l, and that $\mathbf{d}^{(l)}_{k,j}$ from the previous level corresponds to the l first elements in $\mathbf{d}^{(l+1)}_{k,j}$. Hence

$$\theta^{(l+1)}_{k,i,j} = \left[\mathbf{d}^{(l)}_{k,j}, s_{k,j,q}\right]^T \overset{(l+1)}{\mathbf{S}}{}^{-1}_k (\mathbf{x}^{(l+1)}_i - \mathbf{x}^{(l+1)}_{\bullet k}), \tag{2.32}$$

where $s_{k,j,q}$ is the covariance of the jth and qth variables. Then

$$\begin{aligned}
\theta^{(l+1)}_{k,i,j} &= \left[\mathbf{d}^{(l)\,T}_{k,j}, s_{k,j,q}\right] \begin{bmatrix} \overset{(l)}{\mathbf{S}}{}^{-1}_k + \frac{1}{a_{k,q}} \overset{(l)}{\mathbf{S}}{}^{-1}_k \mathbf{d}_{k,q} \mathbf{d}^T_{k,q} \overset{(l)}{\mathbf{S}}{}^{-1}_k & -\frac{1}{a_{k,q}} \overset{(l)}{\mathbf{S}}{}^{-1}_k \mathbf{d}_{k,q} \\ -\frac{1}{a_{k,q}} \mathbf{d}^T_{k,q} \overset{(l)}{\mathbf{S}}{}^{-1}_k & \frac{1}{a_{k,q}} \end{bmatrix} \\
&\qquad \times \left(\left[\mathbf{x}^{(l)\,T}_i, \mathbf{x}_{i,q}\right]^T - \left[\mathbf{x}^{(l)\,T}_{\bullet k}, \mathbf{x}_{\bullet k,q}\right]^T \right) \\
&= \mathbf{d}^{(l)\,T}_{k,j} \overset{(l)}{\mathbf{S}}{}^{-1}_k (\mathbf{x}^{(l)}_i - \mathbf{x}^{(l)}_{\bullet k}) + \frac{1}{a_{k,q}} \mathbf{d}^{(l)\,T}_{k,j} \overset{(l)}{\mathbf{S}}{}^{-1}_k \mathbf{d}_{k,q} \mathbf{d}^T_{k,q} \overset{(l)}{\mathbf{S}}{}^{-1}_k (\mathbf{x}^{(l)}_i - \mathbf{x}^{(l)}_{\bullet k}) \\
&\qquad - \frac{1}{a_{k,q}} (\mathbf{x}_{i,q} - \mathbf{x}_{\bullet k,q}) \mathbf{d}^{(l)\,T}_{k,j} \overset{(l)}{\mathbf{S}}{}^{-1}_k \mathbf{d}_{k,q} - \frac{1}{a_{k,q}} s_{k,j,q} \mathbf{d}^T_{k,q} \overset{(l)}{\mathbf{S}}{}^{-1}_k (\mathbf{x}^{(l)}_i - \mathbf{x}^{(l)}_{\bullet k}) \\
&\qquad + \frac{1}{a_{k,q}} s_{k,j,q} (\mathbf{x}_{i,q} - \mathbf{x}_{\bullet k,q}) \\
&= \theta^{(l)}_{k,i,j} + \frac{1}{a_{k,q}} \left(r^{(l)}_{k,j,q} - s_{k,j,q} \right) \left(\theta^{(l)}_{k,i,q} - \Delta_{k,i,q} \right).
\end{aligned} \tag{2.33}$$

The updating formulas for $r^{(l)}_{k,j,h}$ are found in a similar way:

$$\begin{aligned}
r^{(l+1)}_{k,j,h} &= \mathbf{d}^{(l+1)\,T}_{k,j} \overset{(l+1)}{\mathbf{S}}{}^{-1}_k \mathbf{d}^{(l+1)}_{k,h} \\
&= \left[\mathbf{d}^{(l)\,T}_{k,j}, s_{k,j,q}\right] \\
&\qquad \times \begin{bmatrix} \overset{(l)}{\mathbf{S}}{}^{-1}_k + \frac{1}{a_{k,q}} \overset{(l)}{\mathbf{S}}{}^{-1}_k \mathbf{d}_{k,q} \mathbf{d}^T_{k,q} \overset{(l)}{\mathbf{S}}{}^{-1}_k & -\frac{1}{a_{k,q}} \overset{(l)}{\mathbf{S}}{}^{-1}_k \mathbf{d}_{k,q} \\ -\frac{1}{a_{k,q}} \mathbf{d}^T_{k,q} \overset{(l)}{\mathbf{S}}{}^{-1}_k & \frac{1}{a_{k,q}} \end{bmatrix} \left[\mathbf{d}^{(l)\,T}_{k,h}, s_{k,h,q}\right]^T \\
&= \mathbf{d}^{(l)\,T}_{k,j} \overset{(l)}{\mathbf{S}}{}^{-1}_k \mathbf{d}^{(l)}_{k,h} + \frac{1}{a_{k,q}} \mathbf{d}^{(l)\,T}_{k,j} \overset{(l)}{\mathbf{S}}{}^{-1}_k \mathbf{d}_{k,q} \mathbf{d}^T_{k,q} \overset{(l)}{\mathbf{S}}{}^{-1}_k \mathbf{d}^{(l)}_{k,h} \\
&\qquad + \frac{1}{a_{k,q}} s_{k,h,q} \mathbf{d}^{(l)\,T}_{k,j} \overset{(l)}{\mathbf{S}}{}^{-1}_k \mathbf{d}_{k,q} - \frac{1}{a_{k,q}} s_{k,j,q} \mathbf{d}^T_{k,q} \overset{(l)}{\mathbf{S}}{}^{-1}_k \mathbf{d}^{(l)}_{k,h} + \frac{1}{a_{k,q}} s_{k,j,q} s_{k,h,q} \\
&= r^{(l)}_{k,j,h} + \frac{1}{a_{k,q}} \left(r^{(l)}_{k,j,q} - s_{k,j,q} \right) \left(r^{(l)}_{k,h,q} - s_{k,h,q} \right).
\end{aligned} \tag{2.34}$$

Equations (2.33) and (2.34) allow the computation of $\theta^{(l+1)}_{k,i,j}$ and $r^{(l+1)}_{k,j,h}$ from their values at the previous level in constant O(1) time.

2.3.4. Summary of the relevant formulas for QDA. The following is a summary of the formulas constituting the algorithm for QDA-based forward variable selection.

The classification score for the class of $\mathbf{x}_i$:

$$
\begin{aligned}
f_{\lambda,\setminus i,j}^{(l+1)} &= \frac{N_\lambda^2(N_\lambda-2)}{(N_\lambda-1)^3}\left[\eta_{\lambda,\lambda,i}^{(l)} + \frac{N_\lambda\left(\eta_{\lambda,\lambda,i}^{(l)}\right)^2}{(N_\lambda-1)^2 - N_\lambda\eta_{\lambda,\lambda,i}^{(l)}}\right] \\
&\quad + \frac{N_\lambda^2(N_\lambda-2)}{(N_\lambda-1)^3}\left(\frac{1}{a_{\lambda,j}} + \frac{N_\lambda\left(p_{\lambda,i,j}^{(l)}\right)^2}{(N_\lambda-1)^2 - N_\lambda m_{\lambda,\lambda,i:j}^{(l)}}\right)^{-1} \\
&\quad \times \left(p_{\lambda,i,j}^{(l)} + \frac{N_\lambda p_{\lambda,i,j}^{(l)} m_{\lambda,\lambda,i:j}^{(l)}}{(N_\lambda-1)^2 - N_\lambda m_{\lambda,\lambda,i:j}^{(l)}}\right)^2 \\
&\quad + \ln\left[a_{\lambda,j}\left(\frac{N_\lambda-1}{N_\lambda-2}\right)^{l+1}\left|\mathbf{S}_\lambda^{(l)}\right|\left(1-\frac{N_\lambda m_{\lambda,\lambda,i:j}^{(l)}}{(N_\lambda-1)^2}\right)\right] - 2\ln(P(\omega_k)).
\end{aligned} \tag{2.35}
$$

The score for all other classes is

$$
f_{k,\setminus i,j}^{(l+1)} = \eta_{k,k,i}^{(l)} + a_{k,j}\left(p_{k,i,j}^{(l)}\right)^2 + \ln\left(a_{k,j}\left|\mathbf{S}_k^{(l)}\right|\right) - 2\ln\left(P(\omega_k)\right). \tag{2.36}
$$

Where

$$
a_{k,j} = s_{k,j,j} - r_{k,j,j}^{(l)}, \tag{2.37}
$$

$$
\Delta_{k,i,j} = (\mathbf{x}_{i,j} - \mathbf{x}_{\bullet k,j}), \tag{2.38}
$$

$$
p_{k,i,j}^{(l)} = -\frac{1}{a_{k,j}}\left(\theta_{k,i,j}^{(l)} - \Delta_{k,i,j}\right), \tag{2.39}
$$

$$
m_{k,k,i:j}^{(l)} = \eta_{k,k,i}^{(l)} + \frac{1}{a_{k,j}}\left(\theta_{k,i,j}^{(l)} - \Delta_{k,i,j}\right)^2. \tag{2.40}
$$

$s_{k,j,j}$ is the (j,j) element in the covariance matrix $\mathbf{S}_k^{(d)}$, which is computed using all variables.

Initialization is undertaken at level $l=0$:

$$
\theta_{k,i,j}^{(0)} = r_{k,j,h}^{(0)} = 0, \qquad \left|\mathbf{S}_k^{(0)}\right| = 1. \tag{2.41}
$$

Let q be the variable added at level l. The following formulas allow the quantities necessary at level $l+1$ to be recursively calculated from their values at level l:

$$
r_{k,j,h}^{(l+1)} = r_{k,j,h}^{(l)} + \frac{1}{a_{k,q}}\left(r_{k,j,q}^{(l)} - s_{k,j,q}\right)\left(r_{k,h,q}^{(l)} - s_{k,h,q}\right), \qquad j,h \notin \mathcal{V}^{(l+1)}, \tag{2.42}
$$

$$
\theta_{k,i,j}^{(l+1)} = \theta_{k,i,j}^{(l)} + \frac{1}{a_{k,q}}\left(r_{k,j,q}^{(l)} - s_{k,j,q}\right)\left(\theta_{k,i,q}^{(l)} - \Delta_{k,i,q}\right), \qquad j \notin \mathcal{V}^{(l+1)}, \tag{2.43}
$$

$$
\eta_{k,k,i}^{(l+1)} = \eta_{k,k,i}^{(l)} + \frac{1}{a_{k,q}}\left(\theta_{k,i,q}^{(l)} - \Delta_{k,i,q}\right)^2, \tag{2.44}
$$

$$
\left|\mathbf{S}_k^{(l+1)}\right| = a_{k,q}\left|\mathbf{S}_k^{(l)}\right|. \tag{2.45}
$$

Equations (2.35)–(2.41) can be evaluated in constant time. Hence, since there are N_p scores for $d-l$ variables, the next variable to add in forward selection is found in optimal $\mathrm{O}(N_p d)$ time. Since up to d variables can be selected, the overall complexity of the algorithm is $\mathrm{O}(N_p d^2)$.

2.4. Forward selection based on LDA. The derivation of the algorithm is very similar to that of QDA and is therefore omitted. However, there is one main difference. Because LDA uses the pooled covariance matrix, the leave-one-out scores of $\mathbf{x}_i$ for classes $\omega_k, k \neq \lambda$ are essentially derived in the same way as that for class ω_λ. The only difference is that only the mean $\mathbf{x}_{\bullet\lambda}$ of class ω_λ is affected by omitting $\mathbf{x}_i$, resulting in an extra factor of $N_\lambda^2/(N_\lambda-1)^2$ for the score of ω_λ. Other than that, the derivation for the scores of all classes (2.48) and (2.49) is the same as that for the score of ω_λ for the algorithm of QDA (2.12)–(2.22), with (2.11) replacing (2.9).

Similarly, the derivation of the computational scheme is the same as that for QDA, except that the $\theta^{(l)}_{k,i,j}$ of (2.27) are replaced with the two equations

$$v^{(l)}_{i,j} = \mathbf{d}^T_{k,j} \overset{(l)}{\mathbf{S}_p^{-1}} \mathbf{x}^{(l)}_i, \tag{2.46}$$

$$w^{(l)}_{k,j} = \mathbf{d}^T_{k,j} \overset{(l)}{\mathbf{S}_p^{-1}} \mathbf{x}^{(l)}_{\bullet k}. \tag{2.47}$$

The recursive formulas for both $v^{(l)}_{i,j}$ and $w^{(l)}_{k,j}$ ((2.56) and (2.57)) are derived in the same way as those for $\theta^{(l)}_{k,i,j}$ (2.32)–(2.33). Full details are given in [1].

2.4.1. Summary of the relevant formulas for LDA. The following is a summary of the formulas constituting the algorithm for LDA-based forward variable selection.

The classification score for the class of $\mathbf{x}_i$ with variable j added to the present subset $\mathcal{V}^{(l)}$ is

$$\begin{aligned} f^{(l+1)}_{\lambda,\setminus i,j} = {} & \frac{N_\lambda^2(N_p-K-1)}{(N_\lambda-1)^2(N_p-K)} \left[\eta^{(l)}_{\lambda,\lambda,i} + \frac{N_\lambda \left(\eta^{(l)}_{\lambda,\lambda,i}\right)^2}{(N_\lambda-1)(N_p-K) - N_\lambda \eta^{(l)}_{\lambda,\lambda,i}} \right] \\ & + \frac{N_\lambda^2(N_p-K-1)}{(N_\lambda-1)^2(N_p-K)} \left(\frac{1}{a_{p,j}} + \frac{N_\lambda \left(p^{(l)}_{\lambda,i,j}\right)^2}{(N_\lambda-1)(N_p-K) - N_\lambda m^{(l)}_{\lambda,\lambda,i:j}} \right)^{-1} \\ & \times \left(p^{(l)}_{\lambda,i,j} + \frac{N_\lambda p^{(l)}_{\lambda,i,j} m^{(l)}_{\lambda,\lambda,i:j}}{(N_\lambda-1)(N_p-K) - N_\lambda m^{(l)}_{\lambda,\lambda,i:j}} \right)^2 - 2\ln\left(P(\omega_k)\right). \end{aligned} \tag{2.48}$$

The score for all other classes is

$$\begin{aligned} f^{(l+1)}_{k\setminus i,j} = {} & \frac{N_p-K-1}{N_p-K} \left[\eta^{(l)}_{k,k,i} + \frac{N_\lambda \left(\eta^{(l)}_{\lambda,k,i}\right)^2}{(N_\lambda-1)(N_p-K) - N_\lambda \eta^{(l)}_{\lambda,\lambda,i}} \right] \\ & + \frac{N_p-K-1}{N_p-K} \left(\frac{1}{a_{p,j}} + \frac{N_\lambda \left(p^{(l)}_{\lambda,i,j}\right)^2}{(N_\lambda-1)(N_p-K) - N_\lambda m^{(l)}_{\lambda,\lambda,i:j}} \right)^{-1} \end{aligned}$$

$$\times \left(p_{k,i,j}^{(l)} + \frac{N_\lambda p_{\lambda,i,j}^{(l)} m_{\lambda,k,i:j}^{(l)}}{(N_\lambda - 1)(N_p - K) - N_\lambda m_{\lambda,\lambda,i:j}^{(l)}} \right)^2 - 2\ln\left(P(\omega_k)\right), \tag{2.49}$$

where

$$a_{p,j} = s_{p,j,j} - r_{p,j,j}^{(l)}, \tag{2.50}$$

$$\Delta_{k,i,j} = (\mathbf{x}_{i,j} - \mathbf{x}_{\bullet k,j}), \tag{2.51}$$

$$p_{k,i,j}^{(l)} = -\frac{1}{a_{p,j}} \left(v_{i,j}^{(l)} - w_{k,j}^{(l)} - \Delta_{k,i,j} \right), \tag{2.52}$$

$$m_{k,t,i:j}^{(l)} = \eta_{k,t,i}^{(l)} + \frac{1}{a_{p,j}} \left(v_{i,j}^{(l)} - w_{k,j}^{(l)} - \Delta_{k,i,j} \right) \left(v_{i,j}^{(l)} - w_{t,j}^{(l)} - \Delta_{t,i,j} \right). \tag{2.53}$$

$s_{p,j,j}$ is the (j,j) element in the pooled covariance matrix $\mathbf{S}_p^{(d)}$, formed using all variables.

Initialization is performed at level $l = 0$:

$$v_{i,j}^{(0)} = w_{k,j}^{(0)} = r_{k,j,h}^{(0)} = 0. \tag{2.54}$$

Let q be the variable added at level l. The following formulas allow the quantities necessary at level $l+1$ to be recursively calculated from those at level l:

$$r_{p,j,h}^{(l+1)} = r_{p,j,h}^{(l)} + \frac{1}{a_{p,q}} \left(r_{p,j,q}^{(l)} - s_{p,j,q} \right) \left(r_{p,h,q}^{(l)} - s_{p,h,q} \right), \qquad j \notin \mathcal{V}^{(l+1)}, \tag{2.55}$$

$$v_{i,j}^{(l+1)} = v_{i,j}^{(l)} + \frac{1}{a_{p,q}} (r_{k,j,q}^{(l)} - s_{p,j,q})(v_{i,q}^{(l)} - \mathbf{x}_{i,q}), \qquad j \notin \mathcal{V}^{(l+1)}, \tag{2.56}$$

$$w_{k,j}^{(l+1)} = w_{k,j}^{(l)} + \frac{1}{a_{p,q}} (r_{p,j,q}^{(l)} - s_{p,j,q})(w_{k,q}^{(l)} - \mathbf{x}_{\bullet k,q}), \qquad j \notin \mathcal{V}^{(l+1)}, \tag{2.57}$$

$$\eta_{k,t,i}^{(l+1)} = \eta_{k,t,i}^{(l)} + \frac{1}{a_{p,q}} \left(v_{i,q}^{(l)} - w_{k,q}^{(l)} - \Delta_{k,i,q} \right) \left(v_{i,q}^{(l)} - w_{t,q}^{(l)} - \Delta_{t,i,q} \right). \tag{2.58}$$

Again, (2.48)–(2.54) can be evaluated in constant time; hence the next variable to be selected is found in $O(N_p d)$ time, which is optimal.

3. New $O(N_p d^2)$ algorithms for backward elimination based on the performance of QDA and LDA. With backward elimination, the aim is to delete the variable from the present subset $\mathcal{V}^{(l)}$ such that the performance of a classifier is maximized with the remaining variables. This procedure is repeated until a further reduction in variables increases the error rate estimate or until a specified number of variables remains.

For brevity we omit the derivation of the formulas but present a summary of the equations constituting the algorithm. Full details, as well as formulas for combined forward and backward elimination steps, are given in [1]. With one exception, all quantities in the equations below have been defined in the previous sections. The exception is $\rho_{k,i,j}^{(l)}$, which is defined as follows:

$$\rho_{k,i,j}^{(l)} = \left[\mathbf{S}_k^{(l)-1} \right]_{j^*} (\mathbf{x}_i^{(l)} - \mathbf{x}_{\bullet k}^{(l)}), \qquad j \in \mathcal{V}^{(l)}. \tag{3.1}$$

3.1. Summary of the formulas for backward elimination based on QDA. The following is a summary of the formulas constituting the algorithm for QDA-based backward elimination of variables.

The classification score for the class of $\mathbf{x}_i$ with variable j deleted from $\mathcal{V}^{(l)}$ follows:

$$\begin{aligned} f^{(l-1)}_{\lambda\setminus i,j} = {} & \frac{N_\lambda^2(N_\lambda-2)}{(N_\lambda-1)^3}\left(\eta^{(l)}_{\lambda,\lambda,i}+\frac{N_\lambda\left(\eta^{(l)}_{\lambda,\lambda,i}\right)^2}{(N_\lambda-1)^2-N_\lambda\eta^{(l)}_{\lambda,\lambda,i}}\right) \\ & -\frac{N_\lambda^2(N_\lambda-2)}{(N_\lambda-1)^3}\left(\left[\mathbf{S}_\lambda^{(l)-1}\right]_{j^*,j^*}+\frac{N_\lambda\left(\rho^{(l)}_{\lambda,i,j}\right)^2}{(N_\lambda-1)^2-N_\lambda\eta^{(l)}_{\lambda,\lambda,i}}\right)^{-1} \\ & \times\left(\rho^{(l)}_{\lambda,i,j}+\frac{N_\lambda\rho^{(l)}_{\lambda,i,j}\eta^{(l)}_{\lambda,\lambda,i}}{(N_\lambda-1)^2-N_\lambda\eta^{(l)}_{\lambda,\lambda,i}}\right)^2+(l-1)\ln\left(\frac{N_\lambda-1}{N_\lambda-2}\right) \\ & \times\ln\left[\left|\mathbf{S}^{(l)}_\lambda\right|\left(1-\frac{N_\lambda\eta^{(l)}_{\lambda,\lambda,i}}{(N_\lambda-1)^2}\right)\left(\left[\mathbf{S}_\lambda^{(l)-1}\right]_{j^*,j^*}+\frac{N_\lambda\left(\rho^{(l)}_{\lambda,i,j}\right)^2}{(N_\lambda-1)^2-N_\lambda\eta^{(l)}_{\lambda,\lambda,i}}\right)\right] \\ & -2\ln(P(\omega_k)). \end{aligned} \tag{3.2}$$

The score for all other classes is

$$\begin{aligned} f^{(l-1)}_{k\setminus i,j} = {} & \eta^{(l)}_{k,k,i}-\frac{{\rho^{(l)}_{k,i,j}}^2}{\left[\mathbf{S}_k^{(l)-1}\right]_{j^*,j^*}} \\ & +\ln\left(\left[\mathbf{S}_k^{(l)-1}\right]_{j^*,j^*}\left|\mathbf{S}^{(l)}_k\right|\right)-2\ln(P(\omega_k)), \qquad k\neq\lambda. \end{aligned} \tag{3.3}$$

Initialization at level d follows:

$$\mathbf{S}^{(d)}_k=\frac{1}{N_k-1}\sum_{i=1}^{N_k}(\mathbf{x}^{(d)}_i-\mathbf{x}^{(d)}_{\bullet k})(\mathbf{x}^{(d)}_i-\mathbf{x}^{(d)}_{\bullet k})^T, \tag{3.4}$$

$$\eta^{(d)}_{k,k,i}=(\mathbf{x}^{(d)}_i-\mathbf{x}^{(d)}_{\bullet k})^T\mathbf{S}_k^{(d)-1}(\mathbf{x}^{(d)}_i-\mathbf{x}^{(d)}_{\bullet k}), \tag{3.5}$$

$$\rho^{(d)}_{k,i,j}=\left[\mathbf{S}_k^{(d)-1}\right]_{j^*}(\mathbf{x}^{(d)}_i-\mathbf{x}^{(d)}_{\bullet k}). \tag{3.6}$$

Having found variable q to be deleted at level l, we see that the quantities needed at level $l-1$ are computed as follows.

Let

$$\mathbf{S}_k^{(l)-1}=\begin{bmatrix}\mathbf{A} & \mathbf{c}_q\\ \mathbf{c}_q^T & b_q\end{bmatrix}. \tag{3.7}$$

From (2.5),

$$\mathbf{S}_k^{(l-1)^{-1}} = \mathbf{A} - \frac{1}{b_q}\mathbf{c}_q\mathbf{c}_q^T, \tag{3.8}$$

$$\eta_{k,k,i}^{(l-1)} = \eta_{k,k,i}^{(l)} - \frac{1}{b_q}\left\{[\mathbf{c}_q, b_q](\mathbf{x}_i^{(l)} - \mathbf{x}_{\bullet k}^{(l)})\right\}^2, \tag{3.9}$$

$$\left|\mathbf{S}_k^{(l-1)}\right| = b_q \left|\mathbf{S}_k^{(l)}\right|, \tag{3.10}$$

$$\rho_{k,i,j}^{(l-1)} = \rho_{\lambda,i,j}^{(l)} - \mathbf{c}_{q,j^*}(\mathbf{x}_{i,q} - \mathbf{x}_{\bullet k,q}) - \frac{\mathbf{c}_{q,j^*}}{b_q}\mathbf{c}_q^T(\mathbf{x}_i^{(l-1)} - \mathbf{x}_{\bullet k}^{(l-1)}). \tag{3.11}$$

3.2. Summary of the formulas for backward elimination based on LDA. The following is a summary of the formulas constituting the algorithm for LDA-based backward elimination of variables.

The classification score for class ω_λ of $\mathbf{x}_i$ with variable j deleted from $\mathcal{V}^{(l)}$ follows:

$$\begin{aligned} f_{\lambda\setminus i,j}^{(l-1)} = {} & \frac{N_\lambda^2(N_p - K - 1)}{(N_\lambda - 1)^2(N_p - K)}\left(\eta_{\lambda,\lambda,i}^{(l)} + \frac{N_\lambda\left(\eta_{\lambda,\lambda,i}^{(l)}\right)^2}{(N_\lambda - 1)(N_p - K) - N_\lambda\eta_{\lambda,\lambda,i}^{(l)}}\right) \\ & - \frac{N_\lambda^2(N_p - K - 1)}{(N_\lambda - 1)^2(N_p - K)}\left(\left[\mathbf{S}_p^{(l)^{-1}}\right]_{j^*,j^*} + \frac{N_\lambda\left(\rho_{\lambda,i,j}^{(l)}\right)^2}{(N_\lambda - 1)(N_p - K) - N_\lambda\eta_{\lambda,\lambda,i}^{(l)}}\right)^{-1} \\ & \times \left(\rho_{\lambda,i,j}^{(l)} + \frac{N_\lambda\rho_{\lambda,i,j}^{(l)}\eta_{\lambda,\lambda,i}^{(l)}}{(N_\lambda - 1)(N_p - K) - N_\lambda\eta_{\lambda,\lambda,i}^{(l)}}\right)^2 - 2\ln(P(\omega_k)). \end{aligned} \tag{3.12}$$

The score of all other classes is

$$\begin{aligned} f_{k,\setminus i,j}^{(l-1)} = {} & \frac{N_p - K - 1}{N_p - K}\left[\eta_{k,k,i}^{(l)} + \frac{N_\lambda\left(\eta_{\lambda,k,i}^{(l)}\right)^2}{(N_\lambda - 1)(N_p - K) - N_\lambda\eta_{\lambda,\lambda,i}^{(l)}}\right] \\ & - \frac{N_p - K - 1}{N_p - K}\left(\left[\mathbf{S}_p^{(l)^{-1}}\right]_{j^*,j^*} + \frac{N_\lambda\left(\rho_{\lambda,i,j}^{(l)}\right)^2}{(N_\lambda - 1)(N_p - K) - N_\lambda\eta_{\lambda,\lambda,i}^{(l)}}\right)^{-1} \\ & \times \left(\rho_{k,i,j}^{(l)} + \frac{N_\lambda\rho_{\lambda,i,j}^{(l)}\eta_{\lambda,k,i}^{(l)}}{(N_\lambda - 1)(N_p - K) - N_\lambda\eta_{\lambda,\lambda,i}^{(l)}}\right)^2 - 2\ln\left(P(\omega_k)\right). \end{aligned} \tag{3.13}$$

Initialization at level d (total number of variables):

$$\mathbf{S}_p^{(d)} = \frac{1}{N_p - K}\sum_{k=1}^{K}(N_k - 1)\mathbf{S}_k^{(d)}, \tag{3.14}$$

$$\eta^{(d)}_{k,t,i} = (\mathbf{x}^{(d)}_i - \mathbf{x}^{(d)}_{\bullet k})^T \mathbf{S}^{(d)\,-1}_p (\mathbf{x}^{(d)}_i - \mathbf{x}^{(d)}_{\bullet t}), \tag{3.15}$$

$$\rho^{(d)}_{k,i,j} = \left[\mathbf{S}^{(d)\,-1}_p\right]_{j^*} (\mathbf{x}^{(d)}_i - \mathbf{x}^{(d)}_{\bullet k}). \tag{3.16}$$

Having found variable q to be deleted at level l, the quantities needed at level $l-1$ are computed as follows.

Let

$$\mathbf{S}^{(l)\,-1}_p = \begin{bmatrix} \mathbf{A} & \mathbf{c}_q \\ \mathbf{c}_q^T & b_q \end{bmatrix}. \tag{3.17}$$

From (2.5),

$$\mathbf{S}^{(l-1)\,-1}_p = \mathbf{A} - \frac{1}{b_q}\mathbf{c}_q\mathbf{c}_q^T, \tag{3.18}$$

$$\begin{aligned} \eta^{(l-1)}_{k,t,i} = \eta^{(l)}_{k,t,i} - \frac{1}{b_q} &\left((\mathbf{x}^{(l-1)}_i - \mathbf{x}^{(l-1)}_{\bullet k})^T \mathbf{c}_q + b_q(\mathbf{x}_{i,q} - \mathbf{x}_{\bullet k,q})\right), \\ &\times \left(\mathbf{c}_q^T(\mathbf{x}^{(l-1)}_i - \mathbf{x}^{(l-1)}_{\bullet t}) + b_q(\mathbf{x}_{i,q} - \mathbf{x}_{\bullet t,q})\right), \end{aligned} \tag{3.19}$$

$$\rho^{(l-1)}_{k,i,j} = \rho^{(l)}_{\lambda,i,j} - \mathbf{c}_{q,j^*}(\mathbf{x}_{i,q} - \mathbf{x}_{\bullet k,q}) - \frac{\mathbf{c}_{q,j^*}}{b_q}\mathbf{c}_q^T(\mathbf{x}^{(l-1)}_i - \mathbf{x}^{(l-1)}_{\bullet k}), \qquad j \in \mathcal{V}^{(l-1)}. \tag{3.20}$$

4. Run-time comparisons on two data sets. To illustrate the relative computational speed of the various alternatives, the new optimal algorithms were compared with their previous implementations and with two measures of class separation. Note that for the implementations of the previous QDA- and LDA-based variable selection, fast downdating formulas (2.8)–(2.11) were made use of for the evaluation of the resulting classifier when adding or deleting one variable. The two measures used as the class separation criteria $\mathcal{V}^{(l)}$ are Wilk's Λ and the Mahalanobis distance between the two nearest class means (see [8]). The implementations of these also employ fast down- and updating formulas (2.2) and (2.5), reducing their time complexities from $O(d^4)$ to $O(d^3)$.

Two real data sets were used. The first, the wine data set [14], is 13-dimensional with three classes and 59, 71, and 48 objects per class. The classes correspond to three different cultivars, and the 13 variables measure the different constituents of the three resulting wines. The second data set, the sonar data [14], is 60-dimensional with two classes of size 111 and 97, respectively. The two classes correspond to sonar signals bounced off a metal cylinder and reflected off a roughly cylindrical rock.

In order to compare the measures of class separation and the classifier performance-based algorithms, and also between forward selection and backward elimination, the time measured was the CPU time taken for half the total number of variables available for selection to be added (forward selection) or deleted (backward elimination). All algorithms were implemented in the language C++ on a DEC-Alpha (133 MHz) computer.

TABLE 4.1

CPU times for forward selection and backward elimination for the new "fast" algorithms for QDA and LDA, their previous "slow" implementations, and two F-test-based heuristics. The time recorded is the CPU time taken to add or eliminate half of the total number of variables available for selection.

	Wine		*Sonar*	
	Forward	*Backward*	*Forward*	*Backward*
Fast QDA	5.3 sec	5.3 sec	87.3 sec	70.9 sec
Slow QDA	25.5 sec	62.4 sec	3066 sec	15840 sec
Fast LDA	5.2 sec	8.4 sec	77.0 sec	100.3 sec
Slow LDA	27.5 sec	66.1 sec	3138 sec	15696 sec
Wilk's Λ	1.2 sec	1.1 sec	17.4 sec	16.7 sec
Mahal. dist.	1.2 sec	1.0 sec	17.8 sec	11.7 sec

Table 4.1 summarizes the results and several interesting points are noted. The fastest algorithms are the F-test-based measures, by about a factor of five. An important point to note is that this factor did not change from the medium-dimensional wine data to the high-dimensional sonar data, reflecting the similarity of the time complexities of the two types of algorithms. In contrast, the traditional implementations of QDA and LDA require very large CPU times in the case of the sonar data. For LDA in backward elimination, for example, the speed-up provided by the new algorithm increased from a factor of about 8 to about 160. Hence the speed-up factor itself increased by a factor of about 20, while the dimensionality increased by a factor of less than 5. This reflects that the difference in the time complexity of the algorithms is $O(d^2)$; i.e., the speed-up increases quadratically with the dimensionality. Also note that the F-test and classifier performance-based algorithms are about as fast for the forward and backward approaches, while the previous algorithms for QDA- and LDA-based variable selection are significantly slower for backward elimination.

5. Discussion and conclusions. In stepwise variable selection, the aim is to find the variable to add or delete from the present set of variables such that, in the context of classification, the performance of a classifier is maximized. Due to the excessive computational cost of selecting the variables based on the evaluation of a classifier, measures of the class separation had to be used in the past. We have presented algorithms for variable selection based on the evaluation of a linear or quadratic classifier that, to within a constant factor, are as fast as those used with the measures of class separation. Therefore, because of this dramatic speed-up over the old algorithms, it is now possible to perform classifier performance-based variable selection for data that before would have required the use of a class separability measure.

Acknowledgment. The authors wish to thank the Australian Research Council for supporting this work.

REFERENCES

[1] S. AEBERHARD, O. DE VEL, AND D. COOMANS, *New Fast Algorithms for Variable Selection Based on Classifier Performance*, Tech. report, Department of Computer Science and Department of Mathematics and Statistics, James Cook University, Townsville, Australia, 1993.

[2] K. Fukunaga, *Introduction to Statistical Pattern Recognition*, Academic Press, San Diego, CA, 1990.
[3] G. McLaghlan, *Discriminant Analysis and Statistical Pattern Recognition*, Wiley, New York, 1992.
[4] J. Friedman, *Regularized discriminant analysis*, J. Amer. Statist. Assoc., 84 (1989), pp. 165–175.
[5] S. Ganeshanandam and W.J. Krzanowski, *On selecting variables and assessing their performance in linear discriminant analysis*, Austral. J. Statist., 31 (1989), pp. 433–447.
[6] G.H. Golub and C.F. van Loan, *Matrix Computations*, The John Hopkins University Press, Baltimore, MD, 1989.
[7] J.D.F. Habbema and J. Hermans, *Selection of variables in discriminant analysis by F-statistic and error rate*, Technometrics, 19 (1977), pp. 487–493.
[8] D.J. Hand, *Discrimination and Classification*, Wiley, Chichester, UK, 1981.
[9] D. Hist, *Error-rate estimation in multiple-group linear discriminant analysis*, Technometrics, 38 (1996), pp. 389–399.
[10] C.J. Huberty, *Applied Discriminant Analysis*, Wiley, New York, 1994.
[11] J. Kittler, *Feature set search algorithms*, in Pattern Recognition and Signal Processing, C.H. Chen, ed., Sijthoff & Nordhoff, Groningen, The Netherlands, 1978, pp. 41–60.
[12] P.A. Lachenbruch and M.R. Mickey, *Estimation of error rates in discriminant analysis*, Technometrics, 10 (1968), pp. 1–11.
[13] R.J. McKay and N.A. Campbell, *Variable selection techniques in discriminant analysis:* I. *Description, and* II. *Allocation*, British J. Math. Statist. Psych., 35 (1982), pp. 1–41.
[14] P.M. Murphy and D.W. Aha, *UCI repository of machine learning databases* [Machine-readable data repository], University of California, Department of Information and Computer Science, Irvine, CA, 1991.
[15] G.D. Murray, *A cautionary note on selection of variables in discriminant analysis*, Appl. Statist., 26 (1977), pp. 246–250.
[16] S.J. Raudys and A.K. Jain, *Small sample size effects in statistical pattern recognition: Recommendations for practitioners*, IEEE Trans. Pattern Anal. Machine Intelligence, 13 (1991), pp. 253–264.
[17] S.M. Snapinn and J.D. Knoke, *An evaluation of smoothed classification error-rate estimators*, Technometrics, 27 (1985), pp. 199–206.

SIAM J. SCI. COMPUT.
Vol. 22, No. 3, pp. 1053–1086
© 2000 Society for Industrial and Applied Mathematics

PARALLEL ADAPTIVE SOLUTION OF A POISSON EQUATION WITH MULTIWAVELETS*

A. AVERBUCH†, E. BRAVERMAN‡, AND M. ISRAELI‡

Abstract. We present an adaptive algorithm for the solution of the Poisson equation. The domain is divided into subdomains. The resolution of each subdomain depends on the smoothness of the right-hand side of the Poisson equation. This determines the adaptivity of the algorithm. In each subdomain a particular solution is found. These solutions are patched by introducing double/single layers at the interfaces of the subdomains. The influence of these layers is effectively computed using multiwavelets. In the wavelet bases kernels of integrals which represent double layers are sparse. When the number of grid points increases as N, the number of essential wavelet coefficients, which represent a vector, increases as $\log N$. Hence, using this sparsity reduces the number of operations from $O(N^2)$ to $O(N \log N)$. The algorithm was implemented on parallel computers of SP2 and SGI types while each processor was assigned to each box. The efficiency of the algorithm was demonstrated.

Key words. adaptive algorithms, multiwavelet bases, double and single layers, sparse data structures

AMS subject classifications. 65N35, 65D30, 42C15, 45L10

PII. S106482759833694X

1. Introduction.

Problems in elasticity, fluid dynamics, material science, semiconductor device simulations lead to large-scale initial-boundary value problems for nonlinear parabolic partial differential equations (PDEs). This paper proposes a numerical solution for evolution equations with parabolic terms like the incompressible Navier–Stokes equations.

Semi-implicit discretization in time removes the stringent diffusive stability limit but requires the solution of many elliptic problems. The solution of such equations, at the high resolution necessary in typical applications, requires considerable computation resources. This can be achieved on parallel computers if efficient parallelization is possible. The numerical solution of PDEs has the potential for efficient parallelization on massively parallel multiprocessors because partial derivatives represent in principle local behavior. Of course, it is well known that local changes reach all parts of the computational domain and this dictates global communication. Global communication can degrade the performance of the achieved parallel speedup. There are cases (see [10, 12]) where the effect of local changes usually decays quickly with distance in the time stepping scheme.

In this paper the parallelization of the serial algorithm is achieved by decomposition of the computational domain into smaller domains, mainly of rectangular (cubic) shape. The solution in each rectangular domain is fast and accurate, and it is based on the algorithm that was developed in [3, 4] for the 2-D case and in [8] for the 3-D case. The particular solutions, which were obtained by this approach, have discontinuities on the domain interfaces. The discontinuities can be removed by adding

*Received by the editors April 7, 1998; accepted for publication (in revised form) February 24, 2000; published electronically September 27, 2000.
http://www.siam.org/journals/sisc/22-3/33694.html
†School of Mathematical Sciences, Tel Aviv University, Tel Aviv 69978, Israel (amir@math.tau.ac.il).
‡Technion—Israel Institute of Technology, Computer Science Department, Haifa 32000, Israel (maelena@cs.technion.ac.il, israeli@cs.technion.ac.il).

singularity layers. The effects of the layers on all the other interfaces are computed. The number of computations can essentially be reduced if the corresponding operators are efficiently represented.

An algorithm for a fast solution of the Poisson equation by decomposition of the domain into square domains and the subsequent matching of these solutions by the fast multipole method was developed in [9]. We note the following advantages of the proposed algorithm.

1. The fast implementation of the algorithm in [9] was stipulated by the division of the global domain into squares with a small number of points in each subdomain. This is done since $O(N^3)$ algorithm is applied for the solution of the Poisson equation in the subdomains (here N is a number of points in each direction). Our algorithm has complexity of $O(N^2 \log N)$ in each subdomain. The complexity is independent of the size of the subdomain. The number of subdomains is determined only by the number of processors in the multiprocessor environment.
2. The algorithm has an inherent parallel structure. Therefore, we do not have to modify the serial algorithm when it is ported onto a parallel computer. When the algorithm runs on a parallel computer each subdomain is assigned to a distinct processor. Therefore, each neighboring subdomain (processor) has to transfer to each other the information (jumps of the function or its first derivative represented in wavelet bases) on the common interfaces. Since we transfer between the processors only the information on the interfaces, it costs $O(N)$. The representation by multiwavelets reduces the data to be transmitted to $O(\log N)$, even to the nearest neighbors of the processor.
3. We use a very efficient method to compute the solution of Poisson/Laplace equations inside each domain. The construction of the general solution via matching of the independent solutions in each subdomain can degrade the gain of the solutions inside the domains. Since the computations of the matching among the subdomains are based on double/single layer influences which are not more expensive than the algorithms inside the domain, we are successful in having an efficient solution for the matching step as well. The efficiency of the matching step is due to the usage of multiwavelets discretization.

A basic problem in the numerical solution of differential and integral equations is to find an efficient discrete representation of the underlying continuous operators. The problem with classical methods is that they lead to a dense representation for most operators. That means that matrices are full whereas sparse representation has the advantage of minimizing the operation count during the application of the operator. Dealing with sparse matrices also leads to a decrease of the computational time. Therefore, an important step in the numerical solution consists of building algorithms which lead to a better representation of the usual operators. A good representation means "few coefficients for the same accuracy."

The method that was described in [5, 6] is based on the wavelet transform which provides sparse representations of operator kernels. This transform consists of expanding a given function or an operator over a set of wavelet basis functions obtained by dilations and translations of an elementary function localized in both physical and Fourier spaces. The wavelet transform leads to less computations than those obtained with the regular Fourier transform. The reason to have an efficient solution in [5, 6] is that there is a description for an efficient discretization and adaptive solution of PDEs which are forced by the right-hand side (RHS) with regions of smooth (nonoscilla-

tory) behavior and possibly localized regions with nonsmooth structures. In addition, the wavelet basis allows automatic adaptation (using thresholding) in the sense that only a few coefficients are needed to describe smooth sections of the RHS while more coefficients are needed for sharp transitions and singular points. This saving is due to the vanishing moments property of high-order wavelets. The algorithms are adaptive, i.e., the number of operations performed is proportional to the number of significant coefficients in the wavelet expansion of the "inputs" of a given differential equation problem. The basic tool in this approach is the preconditioned conjugate gradient iteration in a "constrained" form. In the wavelet basis diagonal preconditioners are available which render the condition number of elliptic operators to O(1). This means that a constant number of iterations is required for solution to a prescribed accuracy. In [5, 6] there is a description of a fast 1-D–3-D adaptive method for solving certain elliptic equations with periodic boundary conditions using tensor wavelets with sparse structures.

In this paper we employ a multiwavelet representation of the integral operators which represent single and double layers. The layers are introduced in order to match the particular solutions which were obtained separately in the subdomains. A double layer is introduced on each interface in order to correct the discontinuities of the solution. We assume that we have Dirichlet conditions on the original boundary and matching Neumann conditions on the interfaces. Therefore, we assume that the mixed problems are solved for the nonhomogeneous equation in each subdomain. For faster communication it is preferable to use double layers on the inner boundaries of the subdomains interfaces since they decay faster with distance. A basis that was built in [2] was chosen due to its local character which does not lead to the Gibbs phenomenon. The obtained sparse representation reduces both the computation and the communication time required for parallel processing.

The paper is organized as follows. Section 2 describes the problem, the plan for its solution, and presents a simple 1-D example which illustrates the idea of the solution. Section 3 is focused on the matching step of the algorithm. It contains relevant formulas, numerical examples with the associated operation count. In section 4 we treat the singularities arising from the computation of the influence of the layer at a line which intersects the layer. Section 5 describes the matching step associated with the adaptive algorithm. The adaptation is needed because the resolution in each subdomain is different. Section 6 outlines the parallelization details and illustrates how the amount of communication is reduced if the multiwavelets coefficients are transferred among domains instead of the pointwise values. It also contains timing results for the performance of the parallel algorithm. The appendix contains a brief description of multiwavelets basis developed in [2].

2. The problem. We solve the Poisson equation

$$\Delta u = f(x, y) \quad \text{in} \quad \Omega \tag{2.1}$$

in the rectangular domain $\Omega = [0, L] \times [0, 1]$ with either Dirichlet,

$$u = \Phi(x, y) \quad \text{on} \quad \partial\Omega, \tag{2.2}$$

or Neumann,

$$\frac{\partial u}{\partial n} = \Phi(x, y) \quad \text{on} \quad \partial\Omega, \tag{2.3}$$

boundary conditions where n is the internal normal to $\partial\Omega$. A mixed Dirichlet/Neumann-type boundary condition can also be considered.

To have a parallel solution the domain Ω is divided into l rectangular subdomains. The solution process includes the following steps:

1. In each subdomain a particular solution $u_1^{(s)}$ of the nonhomogeneous equation with arbitrary Neumann (Dirichlet) boundary conditions is found.
2. The collection of particular solutions $u_1^{(s)}$, $s = 1, \ldots, l$, usually have discontinuities (or discontinuities in the derivatives) on the boundaries of the subdomains. We introduce double (single) layers on the boundaries to match the solutions from different domains to have continuous global solution. The effect of these layers on other boundaries is calculated.
3. With the boundary conditions that were computed in the previous step, the solutions $u_1^{(s)}$ are patched by adding the solutions $u_2^{(s)}$, $s = 1, \ldots, l$, of the Laplace equation.
4. An additional solution of the Laplace equation is added to satisfy the boundary conditions on $\partial\Omega$. Namely, for the Dirichlet case the solution u_3 of the homogeneous equation on the boundary $\partial\Omega$ is derived by

$$u_3(x, y) = \Phi(x, y) - u_1(x, y) - u_2(x, y). \tag{2.4}$$

(The case with Neumann boundary conditions is treated similarly.) Thus $u = u_1 + u_2 + u_3$ is the solution of the nonhomogeneous equation with the initial nonhomogeneous boundary conditions.

The following 1-D example illustrates the constructive idea of the algorithm.

Example. We consider the problem

$$u''(x) = 2, \quad x \in [-1, 2], \quad u(-1) = 1, \quad u(2) = 4. \tag{2.5}$$

Step 1. The domain $[-1, 2]$ is divided into three equal subsegments,

$$[-1, 2] = [-1, 0] \cup [0, 1] \cup [1, 2],$$

with arbitrary boundary conditions such that the following three nonhomogeneous boundary value problems are being solved:

$$u_1''(x) = 2, \quad x \in [-1, 0], \quad u_1(-1) = 0, \quad u_1'(0) = 0,$$

$$u_1''(x) = 2, \quad x \in [0, 1], \quad u_1'(0) = 0, \quad u_1(1) = 1,$$

$$u_1''(x) = 2, \quad x \in [1, 2], \quad u_1(1) = 1, \quad u_1(2) = 0.$$

The obtained solution is

$$u_1(x) = \begin{cases} x^2 - 1, & x \in [-1, 0), \\ x^2, & x \in (0, 1), \\ x^2 - 4x + 4, & x \in (1, 2]. \end{cases} \tag{2.6}$$

The global solution is unmatched on 0 as we see in Figure 1.

Step 2. However, the obtained solution has discontinuity in the function on $x = 0$ and discontinuity in the derivative on $x = 1$. We can fix both discontinuities by imposing double and single layers potentials, respectively. In one dimension

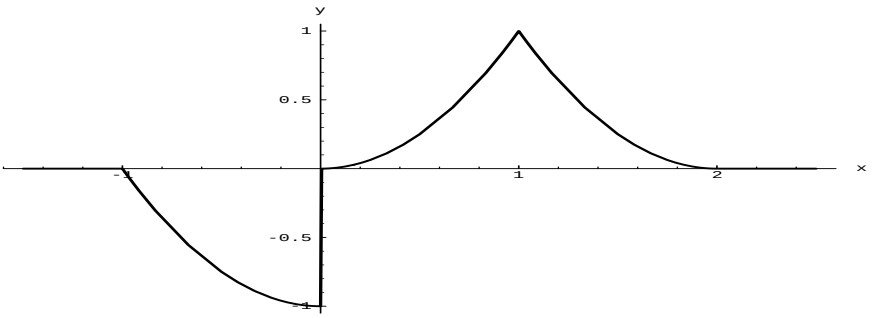


FIG. 1. *Different parts of the solution in each subdomain are unmatched.*

TABLE 1
The influence of double and single layers potential on the boundaries -1,0,1,2.

x	-1	0	1	2
Double layer influence	0.5	0.5/-0.5	-0.5	-0.5
Single layer influence	4	2	0	2
Total influence	4.5	2.5/1.5	-0.5	1.5
Initial value	0	-1/0	1	0
Sum	4.5	1.5	0.5	1.5

the double layer influence is equivalent to adding plus/minus half of the jump to the left/right of the "layer point," respectively. Here, the jump is 1, so 0.5 has to be added to the values of the function for $x < 0$ and -0.5 for $x > 0$. In addition, the single layer in $x = 1$ adds the linear function $a(x-1)/2$ $(-a(x-1)/2$) to each $x < 1$ $(x > 1)$, where $a = u_1'(1+) - u_1'(1-) = -4$. We compute the influence only on the boundaries at $x = -1, 0, 1, 2$.

Step 3. In each domain the homogeneous equation

$$u_2'' = 0$$

is solved with the following boundary conditions (using Figure 1):

$$u_2(-1) = 4.5, \quad u_2(0) = 2.5, \quad x \in (-1, 0),$$

$$u_2(0) = 1.5, \quad u_2(1) = -0.5, \quad x \in (0, 1),$$

$$u_2(1) = -0.5, \quad u_2(2) = 1.5, \quad x \in (1, 2),$$

which corresponds to the row *total influence* in Table 1.
Evidently,

$$u_2(x) = \begin{cases} -2x + 2.5, & x \in [-1, 0), \\ -2x + 1.5, & x \in (0, 1), \\ 2x - 2.5, & x \in (1, 2]. \end{cases} \tag{2.7}$$

The resulting $u_1 + u_2 = x^2 - 2x + 1.5$ is a continuous function which does not satisfy the boundary conditions of (2.5) (see Figure 2), but

$$u_1(-1) + u_2(-1) = 4.5, \quad u_1(2) + u_2(2) = 1.5.$$

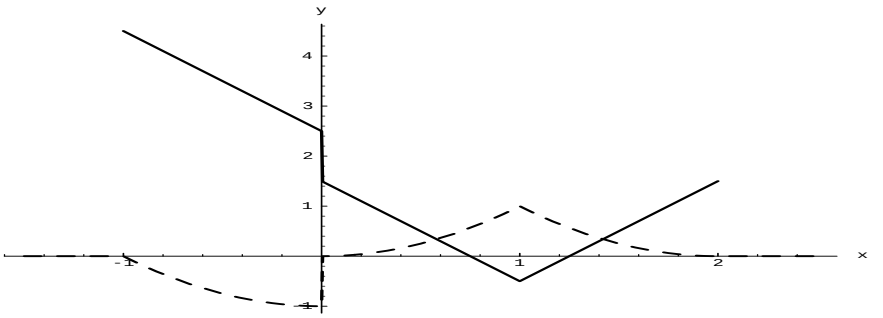


FIG. 2. *Matching of the global solution. Still it does not satisfy the boundary conditions.*

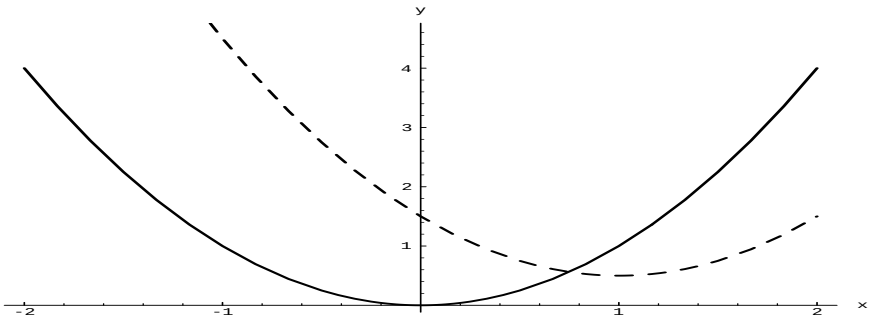


FIG. 3. *Global solution which satisfies the boundary conditions.*

Step 4. We find and add the solution of the homogeneous equation

$$u_3''(x) = 0, \quad u_3(-1) = -3.5, \quad u_3(2) = 2.5.$$

Thus, we obtain the solution for the initial problem. Namely, $u_3 = 2x - 1.5$. Finally, $u(x) = u_1(x) + u_2(x) + u_3(x) = x^2$ (see Figure 3).

The main focus of this paper is how to implement and compute efficiently the influence of the double (single) layer on the other interfaces and boundaries. This corresponds to Step 2 of the above algorithm. All the other steps of the algorithm are discussed in [3, 10, 12].

A double layer is introduced on each interface in order to correct the discontinuities of the solution. From now on we assume that the mixed problems were solved for nonhomogeneous equation in each subdomain. Mixed problem means that we assume that we have Dirichlet conditions on the original boundary and matching Neumann conditions on the interfaces. We use double layers on the inner boundaries of the subdomains interfaces since they decay faster with distance.

3. Computation of the influence of double layer potential. Let us assume the geometry of Figure 4.

3.1. The form of the double layer in the current geometry. The potential of a unit strength dipole which is located at x_0 and has orientation in the $e \in \mathcal{R}^2$ direction is defined by

$$\phi_{x_0,e}(x) = \frac{\partial}{\partial t}(\phi_{x_0}(x + te))\Big|_{t=0} = \frac{e(x - x_0)}{\pi \|x - x_0\|^2}.$$

The potential of the double layer which is introduced on the line $x = \gamma(t),\ t \in [0, L]$ with a dipole density $\tau(t)$ is

$$P_\tau(x) = \int_0^L \phi_{\gamma(t),N(t)}(x)\tau(t)\ dt.$$

We consider the case where the domain $[0, L]\times[0, 1]$ is decomposed into L squares. Half of the jump of the solution on the kth boundary, which is denoted by τ_k, is

$$\tau_k(y) = \frac{u_k(k+0, y) - u_{k-1}(k-0, y)}{2}, \quad k = 1, \ldots, L-1. \tag{3.1}$$

The influence of the dipole layer introduced on the segment $[0, 1]$ of the Y axis with dipole density $\tau(y)$ is evaluated by the following equation on the parallel line $x = d$ and the perpendicular line $y = 0$, respectively:

$$P_d^{paral}(y) = \frac{1}{\pi}\int_0^1 \frac{d}{d^2 + (x-y)^2}\tau(x)dx, \quad 0 \le y \le 1, \tag{3.2}$$

$$P_0^{cross}(x) = \frac{1}{\pi}\int_0^1 \frac{x}{x^2 + y^2}\tau(y)dy, \qquad 0 < x \le L. \tag{3.3}$$

Similarly, the influence of double layers on the other interfaces is computed.

The evaluation of (3.3) and (3.2) is a time consuming step. The representation of the integral operators is done through matrix-vector multiplication [7, 2]. The matrix-vector multiplication in this case can be implemented efficiently using *wavelet* or *multiwavelet* methods.

The representation of the kernels of the integral operators in the multiwavelet bases is precomputed. These kernels depend only on the known geometry of the domain and the subdomains. The coefficients of the multiwavelet representations of τ_k are to be transmitted from processor to processor where the integral of double layer potential is calculated. However, this vector is expected to be sparse for τ_k which is smooth. The sparsity of the coefficient vector also reduces the number of computations needed for the integration.

We describe now the algorithm that computes the integral

$$\mathcal{T}(x) = \int_0^1 T(x, y)\tau_k(y)dx, \quad y \in [0, 1], \tag{3.4}$$

where T is the potential for (x, y) in the mth subdomain, $m \ge k$,

$$T(x, y) = \frac{m - k + x}{(m - k + x)^2 + y^2}, \quad x \in [0, 1],\ y \in [0, 1] \tag{3.5}$$

(other integrals are similarly treated). T is smooth everywhere except at the intersection point of the layer line and the line where the potential is calculated $x = 0,\ y = 0$ if $m = k$.

The integral operator (3.4) is discretized using a simple equispaced quadrature. Thus we define n equispaced points

$$x_i = \frac{i-1}{n-1}, \tag{3.6}$$

and the elements T_{ij} of the $n \times n$ matrix become

$$T_{ij} = \begin{cases} \frac{1}{n-1} T(y_i, x_j), & m-k \neq 0 \text{ or } i \neq 1 \text{ or } j \neq 1, \\ 0, & \text{otherwise.} \end{cases} \tag{3.7}$$

This corresponds to a trapezoid-like quadrature discretization of the operator $\mathcal{T}$.

We decompose the matrix $\{T_{ij}\}$ in a multiwavelet basis of $L^2(\mathcal{R})$ that was developed in [1, 2] which is useful here due the following reasons: 1. Its locality. 2. The multiwavelets computation in each domain does not need any information from neighboring domains. 3. Automatic adaptation to smooth and nonsmooth areas via thresholding. It means that if a function has a mix of singularities and smooth areas, then the number of significant multiwavelet coefficients in its multiscale decomposition is determined by the number of singularities and the number of multiwavelet coefficients that are needed to describe these singularities [5, 6].

We employ here a generalization of this basis which was designed in [2]. The description of the basis is given in the appendix.

3.2. Numerical examples.

3.2.1. Evaluation of the sparsity of the matrices that represent the influence of the layer.

Example 1. We solve the nonhomogeneous Laplace equation

$$\Delta u = 4$$

in the rectangle $0 \leq x \leq 3,\ 0 \leq y \leq 1$ in each of three squares

$$0 \leq x \leq 1,\ 0 \leq y \leq 1,$$

$$1 \leq x \leq 2,\ 0 \leq y \leq 1,$$

$$2 \leq x \leq 3,\ 0 \leq y \leq 1,$$

with Neumann conditions on the boundaries between them:

$$\frac{\partial u}{\partial x}(1-, y) = \frac{\partial u}{\partial x}(1+, y) = 0, \qquad \frac{\partial u}{\partial x}(2-, y) = \frac{\partial u}{\partial x}(2+, y) = 4.$$

Suppose that we solved the three problems and the obtained solution (see Figure 5)

$$u(x, y) = \begin{cases} 2y^2 - 2y + 2, & 0 \leq x < 1, \\ 2x^2 - 4x + 4, & 1 < x < 2, \\ x^2 + y^2 - y, & 2 < x < 3, \end{cases}$$

is discontinuous on the boundaries $y = 1, 2$:

$$\begin{aligned} u(1+, y) - u(1-, y) &= 2 - 2y^2 + 2y - 2 = 2y(1-y), u(2+, y) - u(2-, y) \\ &= 4 + y^2 - y - 8 + 8 - 4 = y^2 - y = y(y-1). \end{aligned}$$

(One can easily check that the derivative in x is continuous for $x = 1, 2$.)

To remove these discontinuities we introduce the double layers with $\tau_1(y) = y(1-y)$ for $x = 1$ and $\tau_2(y) = y(y-1)/2$ for $x = 2$. Evidently, $\tau_1(y) = -2\tau_2(y)$, thus for

TABLE 2

The number of points on the interface is given by the row called "number of points." This number is equal to the number of points on the parallel line at distance 1 *where the influence of the double layer is computed. "Nonzero coefficients" refers to the number of elements above the threshold in the multiwavelet representation of the matrix* $1/(1+(x-y)^2)$. *This matrix is the kernel of the dipole layer potential on the parallel line. The average number of nonzero coefficients per row is also computed. The multiwavelet basis is chosen with* 4 *vanishing moments. The accuracy is given by the maximal absolute error.*

Threshold	Number of points	64	128	256	512	1024	2048
1e-3	nonzero coef.	7	4	9	26	0	0
	per row	0.11	0.03	0.04	0.05	0	0
	accuracy	1.1e-3	1.8e-3	1.8e-3	1.8e-3	-	-
1e-4	nonzero coef.	17	20	29	49	76	41
	per row	0.27	0.16	0.11	0.10	0.07	0.02
	accuracy	4.5e-5	6.5e-5	1.3e-4	1.8e-4	3.3e-4	2.6e-4
1e-5	nonzero coef.	29	45	34	81	107	181
	per row	0.45	0.35	0.13	0.16	0.10	0.09
	accuracy	1.1e-5	8.7e-6	1.0e-5	1.4e-5	1.4e-5	1.7e-5
1e-6	nonzero coef.	65	65	119	144	229	284
	per row	1.01	0.51	0.46	0.28	0.22	0.14
	accuracy	5.5e-6	1.7e-6	7.9e-7	8.7e-7	1.7e-6	1.8e-6
1e-7	nonzero coef.	137	155	204	340	610	836
	per row	2.14	1.21	0.80	0.66	0.60	0.41
	accuracy	5.2e-6	1.4e-6	4.0e-7	1.6e-7	1.1e-7	1.5e-7
1e-8	nonzero coef.	316	348	310	402	904	1218
	per row	4.93	2.72	1.21	0.79	0.88	0.59
	accuracy	5.2e-6	1.3e-6	3.4e-7	9.0e-8	3.4e-8	4.2e-8

precision analysis it is sufficient to estimate the influence f_1 (or f_2) of the layer with the dipole density $\tau_1(y)$. It is antisymmetric for the segments $y = 1, 0 < x < 1$ and $1 < x < 2$. Besides, $\tau_1(y)$ is symmetric, thus for $y = 0,\ 0 < y < 1$ this influence is antisymmetric to the influence on $1 < x < 2$ for $y = 1$. Furthermore, the influence on the parallel lines $x = 2$ and $x = 0$ is the same with the opposite sign. Thus, for the precision analysis we consider the effect of the dipole layer on the crossing segment $y = 1, 1 < x < 3$ and on two parallel segments $x = 2$ (see Table 2) and $x = 3$ (see Table 3) for $0 < y < 1$.

It is natural to assume that the number of essential coefficients decays with the increase of the distance between two parallel segments: one segment is where the double layer is introduced, and the second segment is where the influence of this layer is computed. Table 3 gives the number of essential coefficients in the matrix, and the achieved accuracy when other coefficients for the parallel line at the distance equal to 2 are ignored.

It is to be emphasized that when the matrix in the multiwavelet basis is decomposed all the intermediate results below the threshold are ignored.

We conclude this example with an estimate of how the number of significant wavelet coefficients decays as the distance from the single layer increases in Table 4. We assume the geometry of Figure 4 and compute the representation of the kernel of the *single layer* potential at a parallel line in a multiwavelet basis with four vanishing moments. The distance from the single layer is from 1–16. Obviously for a double layer the decay of the number of significant coefficients is quicker (compare Tables 2, 3).

Example 2. In the previous example we were concerned with a very smooth function for the density of the double layer. Now we consider the case when the density of the double layer is $\exp\{-5(y-0.2)^2\} + \exp\{-50(y-0.7)^2\}$.

TABLE 3

The same computation as in Table 2, but here the influence of the double layer is computed on the parallel line at distance 2. This corresponds to the kernel $2/(4+(x-y)^2)$.

Threshold	Number of points	64	128	256	512	1024	2048
1e-3	nonzero coef.	2	2	2	0	0	0
	per row	0.03	0.02	0.01	0	0	0
	accuracy	5.3e-4	5.4e-4	4.6e-4	-	-	-
1e-4	nonzero coef.	4	8	11	23	15	18
	per row	0.06	0.03	0.04	0.04	0.01	0.009
	accuracy	8.4e-5	1.3e-5	2.6e-4	2.5e-4	2.5e-4	2.4e-4
1e-5	nonzero coef.	14	12	18	37	41	110
	per row	0.22	0.09	0.07	0.07	0.04	0.05
	accuracy	3.7e-6	4.7e-6	1.3e-5	1.3e-5	2.0e-5	4.8e-5
1e-6	nonzero coef.	18	26	31	71	77	272
	per row	0.28	0.20	0.12	0.14	0.08	0.13
	accuracy	3.3e-6	8.6e-7	1.0e-6	6.9e-7	1.3e-6	2.0e-6
1e-7	nonzero coef.	38	46	97	177	290	622
	per row	0.59	0.36	0.38	0.35	0.28	0.30
	accuracy	3.1e-6	8.1e-7	2.4e-7	9.7e-8	1.1e-7	1.7e-7
1e-8	nonzero coef.	95	79	96	216	327	809
	per row	1.48	0.62	0.38	0.42	0.32	0.40
	accuracy	3.0e-6	7.6e-7	2.0e-7	5.5e-8	3.7e-8	4.0e-8

TABLE 4

The number of multiwavelet coefficients above 10^{-7} *for the matrix that describes a single layer potential at a parallel line at distance* d.

Number of coef. above 10^{-7}	Distance d				
	1	2	4	8	16
128×128	234	112	25	15	10
256×256	410	393	62	26	12

It is obvious that the number of essential elements in the matrix, which represents the kernel in the multiwavelet basis, is the same as in the previous example. Therefore, we present here only the accuracy results.

It is important to emphasize that the kernel for a double/single layer depends only on the geometry of the problem. The sparse representation and the structure of this kernel is computed at the preprocessing step. It is common for all the problems with the same geometry and domain decomposition.

3.2.2. Evaluation of the sparsity of the wavelet representation of the density of layers. The complexity of the matching step is determined by the sparsity of the matrix and the vector that are involved in the matrix-vector multiplication. The matrix represents the influence of the layer in the wavelet basis, while the vector represents the density of the layer in the same basis. The number of operations depends not only on the sparsity of the matrix but also on the sparsity of the vector. For instance, in Example 2 the vector was a quadratic polynomial, so its multiwavelet representation had only three nonzero coefficients at the coarsest level if the number of vanishing moments is not less than three. In the case when the vector is a steep Gaussian function the accuracy results are given in Table 5 and the sparsity results are presented in Table 6.

At the matching step first each dipole layer density is represented in the multiwavelet basis. In the parallel implementation, then, the multiwavelet coefficients are transmitted to each processor in order to compute the influence of all double layers on

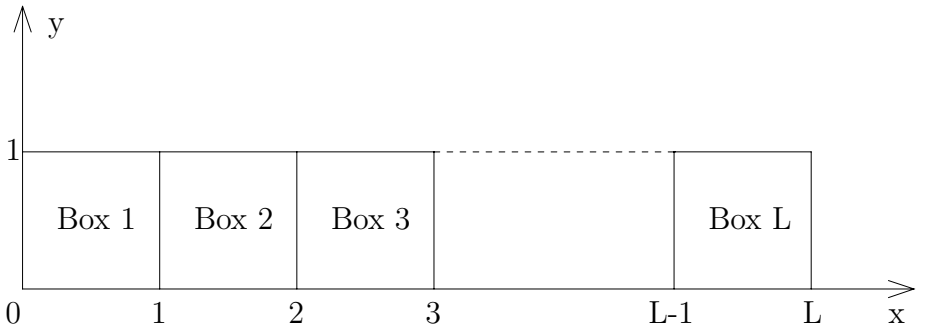


FIG. 4. *The domain is decomposed into L subdomains. Double layers are introduced on the* $L-1$ *interfaces.*

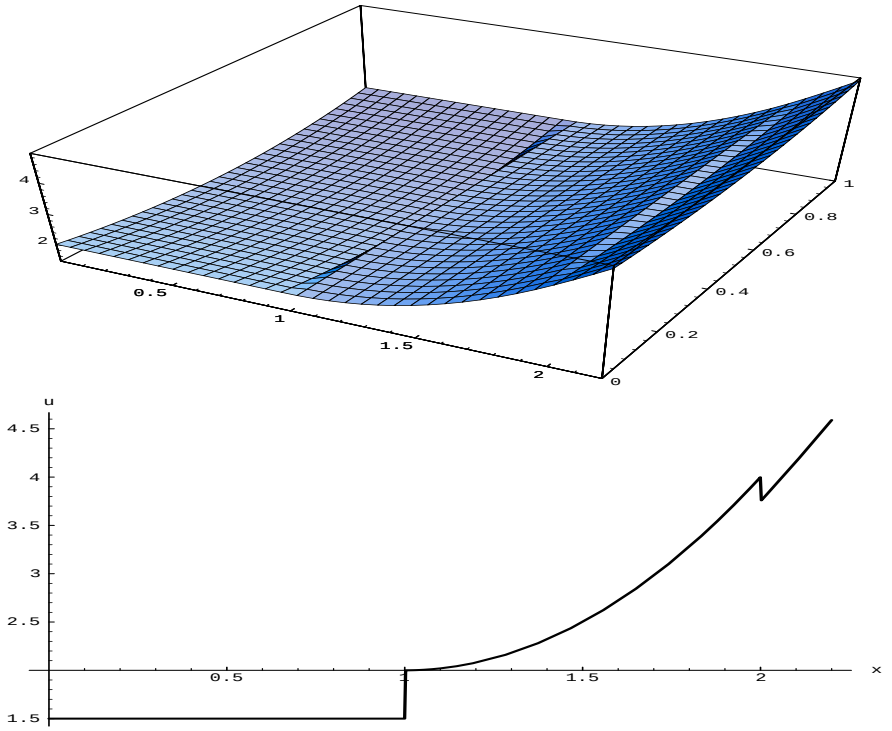


FIG. 5. *A surface plot and graph of* $u(0.5, y)$.

its boundaries. Only the coefficients above the threshold are transmitted. Therefore, the sparsity of the multiwavelet coefficients vector reduces the volume of the communication. If the density is constant then only one coefficient has to be transmitted. Consider the case where the density is in the form of a Gaussian bell

$$\tau(y) = e^{-\alpha(y-y_0)^2}.$$

Table 6 describes the comparison between the number of essential coefficients in the original function and in the multiwavelet representation. The number of vanishing moments is $k = 4$ and $y_0 = 0.5$.

TABLE 5

Accuracy of the computation in the multiwavelet basis with 4 *vanishing moments in the case when the density of the double layer is* $\exp\{-5(y-0.2)^2\} + \exp\{-50(y-0.7)^2\}$. *The line where the effect of the double layer is computed is parallel at distance* 1.

Threshold/number of points	64	128	256	512	1024	2048
1e-3	2.1e-3	4.2e-3	6.1e-3	1.9e-2	-	-
1e-4	1.1e-4	2.0e-4	4.0e-4	9.1e-4	1.0e-3	5.8e-4
1e-5	2.3e-5	2.5e-5	2.8e-5	4.1e-5	6.4e-5	3.5e-5
1e-6	9.5e-6	3.7e-6	3.5e-6	5.2e-6	7.3e-6	8.4e-6
1e-7	8.1e-6	2.2e-6	7.9e-7	6.3e-7	9.4e-7	8.0e-7
1e-8	7.8e-6	2.0e-6	5.3e-7	1.7e-7	1.0e-7	2.2e-7
1e-9	7.6e-6	2.0e-6	4.8e-7	1.3e-7	4.1e-8	2.0e-8

TABLE 6

Number of essential multiwavelet coefficients for the representation of Gaussian functions $\tau(y) = \exp\{-\alpha(y-0.5)^2\}$; *number of vanishing moments is* $k = 4$.

α	Threshold	512	1024	2048	4096	8192
400	1.e-3	35	41	43	43	47
	1.e-4	57	59	63	67	73
	1.e-5	77	81	91	93	103
	1.e-6	107	129	135	141	151
	1.e-7	135	171	199	225	235
1600	1.e-3	43	43	49	51	51
	1.e-4	57	65	67	71	75
	1.e-5	77	85	92	99	101
	1.e-6	91	115	137	143	149
	1.e-7	103	143	179	207	233
4000	1.e-3	43	49	53	55	57
	1.e-4	61	67	67	69	71
	1.e-5	63	81	95	99	105
	1.e-6	75	101	127	137	151
	1.e-7	81	119	161	195	213

The influence of the dipole layer decays as $1/r$, where r is the distance from the point to the dipole line (in our case, the distance between the parallel lines is an integer; see Figure 4). This is important to the parallel communication when the essential elements are transmitted from a subdomain to a subdomain. Small elements are not transmitted to large distances. It reduces the information stream to remote processors.

3.2.3. Implementation of the algorithm. The full algorithm was implemented on the configuration depicted in Figure 4. First, the Dirichlet problem for the Poisson equation was solved in each of the three boxes. The boundary conditions were chosen to provide the continuity of the solutions. The single layers were introduced to match the first derivative in the adjacent boxes. The sum of influences of these layers was computed at all the interfaces. Then, the Laplace equation was solved with the boundary conditions equal to the sum of influences. Finally, the global Dirichlet problem was solved in the domain $0 \le x \le L$, $0 \le y \le 1$ to satisfy the original boundary conditions. All the algorithms were chosen to provide $O(h^4)$ convergence. The algorithm employed for the solution of the Poisson and the Laplace equations was developed in [3].

Assume that u is the exact solution and u' is the computed solution. In the

TABLE 7

MAX, MSQ, and $\mathcal{L}^2$ errors for the Poisson equation with the exact solution $u(x,y,z) = \cos x \cos y$ for three boxes.

$N_x \times N_y$ in each box	ε_{MAX}	ε_{MSQ}	$\varepsilon_{\mathcal{L}^2}$
8×8	8.2e-5	2.4e-5	4.0e-5
16×16	5.4e-6	1.6e-6	2.7e-6
32×32	4.1e-7	1.2e-7	2.1e-7
64×64	2.9e-8	8.2e-9	1.4e-8
128×128	2.0e-9	5.4e-10	9.2e-10
256×256	1.3e-10	3.5e-11	5.9e-11
512×512	8.5e-12	2.2e-12	3.8e-12

TABLE 8

MAX, MSQ, and $\mathcal{L}^2$ errors for the Poisson equation where the exact solution is $u(x,y,z) = \cos x \cos y$ for three boxes, number of points in each box is 128×128, and the number of extension points N_ϵ for the solution of the Poisson equation varies.

N_ϵ	ε_{MAX}	ε_{MSQ}	$\varepsilon_{\mathcal{L}^2}$
8	6.0e-9	1.8e-9	3.1e-9
11	1.9e-9	5.3e-10	9.0e-10
16	2.0e-9	5.4e-10	9.2e-10
32	2.0e-9	5.4e-10	9.2e-10
64	2.0e-9	5.4e-10	9.2e-10
128	2.0e-9	5.4e-10	9.2e-10

examples we will use the following measures to estimate the errors:

$$\begin{aligned} \varepsilon_{MAX} &= \max \|u_i' - u_i\|, \\ \varepsilon_{MSQ} &= \sqrt{\frac{\sum_{i=1}^{N} (u_i' - u_i)^2}{n}}, \\ \varepsilon_{\mathcal{L}^2} &= \sqrt{\frac{\sum_{i=1}^{N} (u_i' - u_i)^2}{\sum_{i=1}^{N} u_i^2}}. \end{aligned}$$

Example 3. We solve the Poisson equation $\Delta u = -2 \cos x \cos y$ with the boundary conditions corresponding to the exact solution $u(x,y,z) = \cos x \cos y$ in the domain $[0,3] \times [0,1]$ divided into three equal boxes (see Table 7).

We recall that for the solution of the Poisson equation the RHS is extended into a wider domain such that it is periodic in the area including $N + 2N_\epsilon$ grid points in each direction. First, the Poisson equation is solved in the extended domain; then the solution in the original domain is considered as a partial solution. Table 8 describes the dependency between the accuracy and the length of the extension N_ϵ when the number of points in each subdomain is equal to 128×128 and the number of subdomains is three. We can see that when the extension exceeds 11 grid points the accuracy does not change.

Example 4. Consider the Poisson equation with the same solution as in Example 3 in the domain $[0,8] \times [0,1]$ divided into eight equal boxes (see Table 9).

Table 10 presents the accuracy of the numerical solution when the exact solution is $u(x,y,z) = \cos x \cos y$ and the number of boxes is varied.

We can see that the error is nearly independent of the number of processors.

Example 5. We solve the Poisson equation with the boundary conditions corresponding to the exact solution

$$u(x,y) = \exp\left\{\alpha\left((x-x_0)^2 + (y-y_0)^2\right)\right\},$$

TABLE 9

MAX, MSQ, and $\mathcal{L}^2$ errors for the Poisson equation with the exact solution $u(x,y,z) = \cos x \cos y$ for the configuration with eight equal boxes.

$N_x \times N_y$ in each box	ε_{MAX}	ε_{MSQ}	$\varepsilon_{\mathcal{L}^2}$
16×16	9.6e-6	2.8e-6	4.8e-6
32×32	7.4e-7	2.1e-7	3.6e-7
64×64	5.3e-8	1.5e-9	2.5e-8
128×128	3.6e-9	9.8e-10	1.6e-9

TABLE 10

MAX, MSQ, and $\mathcal{L}^2$ errors for the Poisson equation with the exact solution $u(x,y,z) = \cos x \cos y$. In each box there are 64 points. The number of boxes reflects the number of available processors.

Number L of boxes	ε_{MAX}	ε_{MSQ}	$\varepsilon_{\mathcal{L}^2}$
3	2.9e-8	8.2e-9	1.4e-8
4	5.3e-8	1.3e-8	2.1e-8
6	5.3e-8	1.5e-8	2.2e-8
8	5.3e-8	1.5e-8	2.5e-8
12	5.3e-8	1.5e-9	2.5e-8
16	5.3e-8	1.5e-9	2.5e-8

with $x_0 = 1.5,\ y_0 = 0.5,\ \alpha = 2$ in the domain $[0,3] \times [0,1]$ divided into three equal boxes (see Table 11).

In Figure 6 the solution before the matching step is presented. It has discontinuities in the x derivative at the lines $x = 1$ and $x = 2$.

After the matching step the solution becomes smooth (see Figure 7).

Table 12 presents the accuracy of the algorithm for a Gaussian function as the exact solution when the steepness $\alpha = 0.5,\ 2,\ 8$ varies.

Example 6. We solve the Poisson equation with an exact solution in $[0,4] \times [0,1]$ being a sum of random Gaussian functions $u(x,y) = \exp\left\{\alpha_i\left((x-x_i)^2 + (y-y_i)^2\right)\right\}$, centered at the points (0.002,0.4), (1.7,0.8), (1.95,2.5), (2.3,1.1), (1.4,2.3), (3.8,2.9), (1.5,3.6), (2.05,1.8), (0.4,3.3), (2.8,1.6), (3.5,3.1), (0.6,3.8), with $\alpha_i = 3,\ 7,\ 1,\ 0.5,\ 4,\ 0.7,\ 2.5,\ 0.2,\ 5.5{,}1.5,\ 3.2,\ 0.8$, respectively. Some of the centers are close to the boundaries and interfaces. The domain was divided into four equal squares. Table 13 presents the accuracies obtained.

3.3. Operation count. Consider the case when the domain is divided into L square subdomains as in Figure 4. We estimate the number of operations while taking into consideration the influences of the double layers which were introduced on the interfaces.

Preprocessing. The preprocessing contains a representation of the kernels of the integrals (3.2) and (3.3) in the multiwavelet basis. We have to treat L kernels which correspond to the influence of double layers on the boundaries (interfaces) parallel to the y axis with distance of $1, 2, \ldots, L$ from the double layer. The same number of kernels describes the influences of double layers on perpendicular boundaries. (It is obvious that the influences on $y = 0$ and $y = 1$ lines lead to the same matrices.) Representation of each kernel requires $O(Nk^2)$ operations [2], where k is the number of vanishing moments and N is the number of discretization points. Therefore, this requires $2LO(Nk^2)$ operations. This is the most time consuming step in the whole algorithm. However, this procedure is implemented as a preprocessing step since it depends only on the geometry of the subdomains. Once this is completed, various

TABLE 11

MAX, MSQ, and $\mathcal{L}^2$ errors for the Poisson equation with the exact solution $u(x,y) = \exp\{2(x-1.5)^2 + 2(y-0.5)^2\}$.

$N_x \times N_y$ in each box	ε_{MAX}	ε_{MSQ}	$\varepsilon_{\mathcal{L}^2}$
8×8	2.3e-3	7.1e-4	1.6e-3
16×16	1.3e-4	3.9e-5	8.5e-5
32×32	6.8e-6	2.0e-6	4.4e-6
64×64	3.9e-7	1.1e-7	2.4e-7
128×128	2.3e-8	6.6e-9	1.4e-8
256×256	1.4e-9	4.0e-10	8.5e-10
512×512	8.7e-11	2.4e-11	5.2e-11
1024×1024	6.3e-12	1.9e-12	4.0e-12

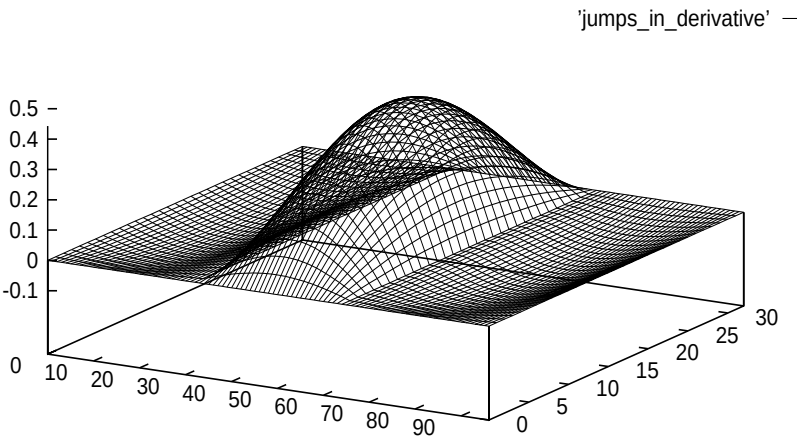


FIG. 6. *The global solution as a collection of nonmatched solutions with discontinuities in the first derivative.*

problems with different RHSs and boundary conditions can be treated. For example, if we solve the Navier–Stokes equation in a certain domain the integral operators of the type (3.2) and (3.3) are represented in the multiwavelet basis only once.

The resulting matrices have $O(N)$ essential elements above the threshold. The matrices which describe the influence of the layer on distant boundaries and interfaces have only $O(1)$ elements with absolute value above a certain threshold.

The actual computation. The algorithm incorporates the following steps:

1. We compute the difference between the adjacent solutions which costs $O(N)$ operations. Double layers for matching solutions are introduced. The densities of double layers τ_k, $k = 1, \ldots, L-1$ are decomposed in multiwavelet bases. This requires $O(Nk)$ operations for each density [2]. It is worth mentioning that we used the standard form for the matrix representation in a multiwavelet base [7]. This leads to a sparse representation of τ_k for a smooth dipole density τ_k.

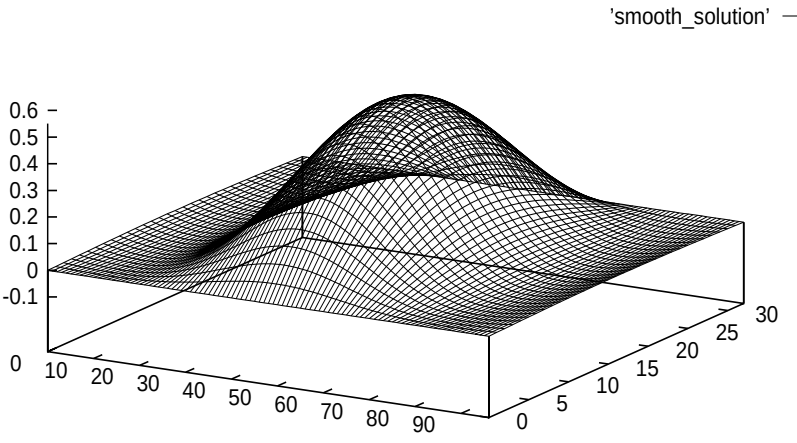


FIG. 7. *The global solution after the matching step.*

TABLE 12
MAX error for the Poisson equation with the exact solution $u(x,y) = \exp\{\alpha((x-1.5)^2 + 2(y-0.5)^2)\}$ *for* $\alpha = 0.5, 2, 8$.

$N_x \times N_y$	$\alpha = 0.5$	$\alpha = 2$	$\alpha = 8$
24×8	1.5e-4	2.3e-3	9.7e-3
48×16	1.1e-5	1.3e-4	1.4e-3
96×32	8.4e-7	6.8e-6	5.9e-5
192×64	5.6e-8	3.9e-7	2.1e-6
376×128	3.7e-9	2.3e-8	8.4e-8
768×256	2.3e-10	1.4e-9	4.0e-9
1536×512	1.5e-11	8.7e-11	2.2e-10
3072×1024	7.5e-13	6.3e-12	1.5e-11

2. Matrix-vector multiplication for a matrix with not more than $O(N)$ essential elements and a sparse vector with $O(1)$ essential elements can be implemented in $O(N)$ operations using sparse data structures. Tables 2–6 demonstrate the obtained sparsity of matrices and vectors, even in the cases when the density is a steep function. In matrices which represent kernels of integral operators the number of essential elements per row decreases with the increase of N.
3. The reconstruction of the resulting vector from the multiwavelet coefficients back to the physical space requires $O(Nk)$ operations.

Therefore, the total number of operations is $O(Nk^2)$ for the preprocessing step and $O(Nk)$ for the actual implementation, where N is the number of discretization points on the long side of the rectangle and k is the number of vanishing moments.

4. Treatment of singularities on a line intersecting the double layer. Consider (3.3) for the computation of the influence of the double layer on the inter-

TABLE 13
MAX, MSQ, and $\mathcal{L}^2$ errors for the Poisson equation with the exact solution being a sum of random Gaussians.

$N_x \times N_y$ in each box	ε_{MAX}	ε_{MSQ}	$\varepsilon_{\mathcal{L}^2}$
8×8	2.4e-2	4.2e-3	1.8e-3
16×16	2.0e-3	3.2e-4	1.3e-4
32×32	1.0e-4	1.3e-5	5.6e-6
64×64	5.1e-6	5.4e-7	2.3e-7
128×128	2.8e-7	2.6e-8	1.1e-8
256×256	1.6e-8	1.5e-9	6.2e-10
512×512	9.7e-10	8.9e-11	3.8e-11

TABLE 14
The accuracy for computing the influence of the double layer with density $\tau(y) = y(1-y)$ for parallel lines with distance equal to 1. The $\sin^4$ transformation was used.

Number of points	4	8	16	32
Accuracy (maximal error)	5.4e-3	1.1e-5	8.6e-11	6.8e-14

secting line. It contains an improper integral with the kernel

$$K(y,x) = \frac{x}{x^2+y^2},$$

which is unbounded at the origin $(0,0)$. Such an integral cannot be efficiently evaluated by the equispaced trapeze method. High accuracy can be achieved if we make the substitution $y = \varphi(t)$ in the integral

$$\int_0^1 K(x,y)\tau(y)\ dy = \int_0^1 K(x,\varphi(t))\tau(\varphi(t))\varphi'(t)\ dt.$$

Here $\varphi(t)$ is an increasing function of t which defines a one-to-one mapping of [0,1] and has m vanishing derivatives at $t = 0$ and $t = 1$. After the substitution the error of the trapezoidal rule in the variable t becomes $O(n^{-2m-1})$, where n is a number of the discretization points [11]. This method corresponds to a nonequispaced discretization where the grid points are concentrated near the edges. We employed the trigonometric $\sin^m$ transformation of [11] with $m = 4$ vanishing derivatives at the end points

$$\varphi(t) = \frac{1}{12\pi}(12\pi t - 8\sin 2\pi t + \sin 4\pi t).$$

The theoretical error in this special case is $O(n^{-2m-2}) = O(n^{-10})$, i.e., for each doubling of the number of points the error is reduced 1000 times. Table 14 shows the influence of the double layer with density $\tau(y) = y(1-y)$ at the parallel line at distance 1 using the $\varphi(t)$ substitution.

Table 15 shows the influence of the double layer with density $\tau(y) = y^3(1-y)$ on the perpendicular line using the $\sin^4$ transformation.

This method gives accurate results for a smooth and nonoscillating density function. However, it fails for both oscillating (for example, $\tau(y) = \cos 6y$ was checked) and δ-type functions (such as steep Gaussians). Moreover, it requires nonequispaced discretization. The densities of the double layers are established as the jumps between adjacent particular solutions which are found in the equispaced points. These are the reasons why the above method was not used by us.

TABLE 15

The accuracy for computing the influence of the double layer with density $\tau(y) = y^3(1-y)$ on the perpendicular line. The $\sin^4$ *transformation was used.*

Number of points	8	16	32
Accuracy (maximal error)	1.4e-5	9.5e-9	7e-12

TABLE 16

Accuracy for the computation of the influence of the double layer on the intersecting line. The kernel of the integral representing the influence is equal to $xy^3/(x^2+y^2)$; the density of the double layer is $\tau(y) = 1-y$. Middle point integration method was applied.

Number of points	8	16	32	64	128	256
Maximal error	9.3e-5	2.5e-5	6.4e-6	1.62e-6	4.04e-7	1.01e-7
Left point error	6.0e-5	7.8e-6	1e-6	1.3e-7	1.6e-8	2e-9

We use the subtraction technique which allows to keep both accuracy and equispaced discretization. Due to the Taylor series in $y = 0$ each density function $\tau(y)$ can be written as

$$\tau(y) = A + By + Cy^2 + y^3\delta(y),$$

where $\delta(y)$ is a continuous function in $y = 0$. Therefore, one easily finds that

$$\int_0^1 K(x,y)\tau(y)dy = \left(-A\arctan\frac{x}{y} + B\frac{x}{2}\ln(x^2+y^2) + Cxy + Cx^2\arctan\frac{x}{y}\right)\Bigg|_0^1 + \int_0^1 K_1(y,x)\delta(y)dy,$$

where the kernel $K_1(x,y) = xy^3/(x^2+y^2)$ is nonsingular. The accuracy results from the computation of the last integral is presented in Table 16. The following errors were computed: the error at the closest point to the point of intersection (left point error), and the maximal absolute value of all the errors at the boundary (maximal, or l_∞, error).

In fact, subtraction of two terms is sufficient in order to reach the same accuracy as in the case of the parallel lines.

5. Adaptive algorithm. In this section we describe an adaptive implementation of the algorithm for the solution of the Poisson equation. The main strategy is similar to that developed in [9]. The domain is divided into subdomains where the grid is chosen according to the smoothness of the RHS. This allows avoidance of massive computation in the domains where the solution is smooth, and thus, even a small number of points is enough to reach a high accuracy. In section 5.1 we illustrate this idea with some examples which demonstrate high accuracy and quick convergence for an adaptive grid. In these examples the RHS is such that the exact solution is a steep Gaussian centered in the middle of the domain. The domain is divided into three sections, where a denser grid is applied in the central section. Similarly, a hierarchy of grids can be chosen according to the behavior of the RHS. In section 5.2 the matching procedure for subdomains with different grids is discussed. The number of operations in matching procedure is evaluated when the corresponding matrix-vector multiplications are accomplished in multiwavelet basis. The efficiency of the algorithm depends on the sparsity of the matrix obtained.

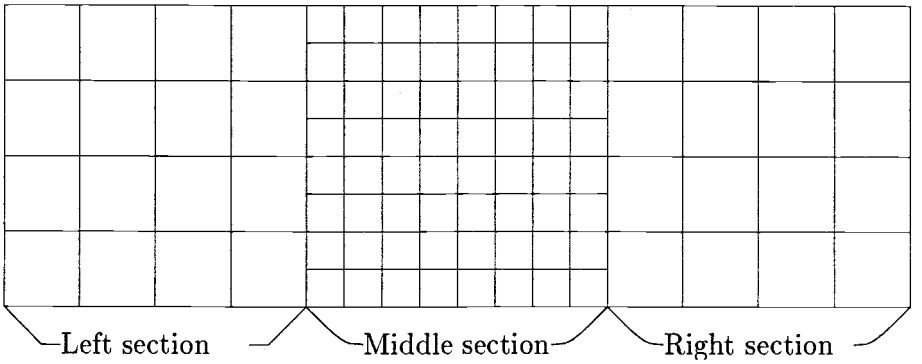


FIG. 8. *The domain is divided into three sections, where the grid in each section depends on the smoothness of the RHS in the Poisson equation.*

TABLE 17
The errors for the Poisson equation with the exact solution being a steep Gaussian with $\alpha = 10$.

$N_x \times N_y$ in middle section	$N_x \times N_y$ in side section	ε_{MAX}	ε_{MSQ}	$\varepsilon_{\mathcal{L}^2}$	$\frac{\varepsilon_{MAX}(N/2)}{\varepsilon_{MAX}(N)}$
32×32	16×16	1.4e-3	4.6e-4	2.1e-3	
64×64	32×32	5.9e-5	2.2e-5	9.7e-5	24.7
128×128	64×64	2.1e-6	7.4e-7	3.3e-6	28.1
256×256	128×128	8.4e-8	2.4e-8	1.0e-7	25.0
512×512	256×256	4.0e-9	8.4e-10	3.7e-9	21.0

5.1. Implementation of an adaptive algorithm. Let the domain be a rectangular $[0,3] \times [0,1]$ (see Figure 8). Consider the case when the steepest parts of the RHS are concentrated near the center of the domain. For example, the exact solution is a Gaussian (a sum of Gaussians) centered near (1.5,0.5). We divide the domain into three equal squares. The finest grid is chosen in the middle section to ensure the necessary resolution of the steep RHS. The grid in the side sections is coarser: it contains four times fewer points than in the middle section. Examples 7–8 illustrate the convergence and accuracy of the adaptive algorithm.

Example 7. We solve the Poisson equation with an exact solution in $[0,3] \times [0,1]$ being a Gaussian function $u(x,y) = \exp\left\{-10\left((x-1.5)^2 + (y-0.5)^2\right)\right\}$ (see Figure 8).

The first and the second columns of Tables 17 and 18 present the number of points in the middle and the side squares, respectively; columns 3–5 give the errors of the algorithm implementation. The last column computes the decrease of the maximal error in the global domain when the number of grid points in each direction is doubled. We recall that the expected accuracy of the algorithm is $O(h^4)$, i.e., the error should be 16 times less when the grid step is twice less than the previous grid step.

Example 8. We solve the Poisson equation with an exact solution in $[0,3] \times [0,1]$ being a sum of steep and smooth Gaussian functions

$$\exp\left\{-12\left((x-1.3)^2 + (y-0.6)^2\right)\right\}$$

$$+ \exp\left\{-25\left((x-1.5)^2 + (y-0.5)^2\right)\right\} + \exp\left\{-3\left((x-1.8)^2 + (y-0.3)^2\right)\right\}$$

TABLE 18
The errors for the Poisson equation with the exact solution being a sum of steep and smooth Gaussians with $\alpha = 18, 25, 3$

.

$N_x \times N_y$ in middle section	$N_x \times N_y$ in side section	ε_{MAX}	ε_{MSQ}	$\varepsilon_{\mathcal{L}^2}$	$\frac{\varepsilon_{MAX}(N/2)}{\varepsilon_{MAX}(N)}$
64×64	32×32	3.8e-4	7.0e-5	1.2e-4	
128×128	64×64	1.6e-5	2.7e-6	4.5e-6	23.8
256×256	128×128	7.4e-7	9.9e-8	1.6e-7	21.6
512×512	256×256	3.7e-8	4.1e-9	6.7e-9	20.0

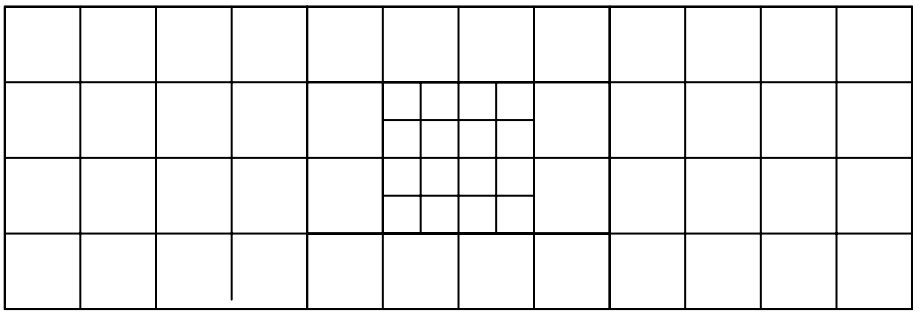


FIG. 9. *The grid in the small box in the middle is two times denser in each direction than in the remaining domain (each box contains several grid points according to the table.*

and with the same Figure 8. Table 18 presents the accuracies obtained.

In this and the previous examples one can observe that the error decays faster than $O(h^4)$ due to the adaptivity of the grid to the behavior of the RHS. The RHS is very steep; however, high accuracy is obtained for comparatively small number of grid points.

Example 9. We solve the Poisson equation with an exact solution in $[0,3] \times [0,1]$ being a Gaussian function $u(x,y) = \exp\left\{-15\left((x-1.5)^2 + (y-0.5)^2\right)\right\}$ (see Figure 9 and Table 19).

The domain is again divided into three sections, where the grid in each section may depend on the smoothness of the RHS in the Poisson equation.

The grid in the small box in the middle is two times denser in each direction.

Example 10. We solve the Poisson equation with an exact solution in $[0,3] \times [0,1]$ being a Gaussian function $u(x,y) = \exp\left\{-25\left((x-1.5)^2 + (y-0.5)^2\right)\right\}$ (see Figure 10 and Table 20).

The grid in the small box in the middle is four times denser in each direction.

5.2. The matching step for adaptive algorithms. In this section we consider the domains' matching procedure using multiwavelets and evaluate the number of operations that are necessary for its implementation in the adaptive case. The domain is divided into subdomains, possibly of different sizes and different resolution in each subdomain (see Figure 11). In the beginning we will consider the case where the subdomain is divided into unequal squares of different resolutions. Then we will consider the case where the domain consists of rectangular subdomains of different aspect ratios and resolutions.

5.2.1. Matching of square boxes. We compute the effect of the double layer on the interface between boxes 1 and 2,3 on the left side of box 4 (see Figure 12). We

TABLE 19
The errors for the Poisson equation, where the exact solution is a steep Gaussian ($\alpha = 15$).

$N_x \times N_y$ in middle section	$N_x \times N_y$ in side section	ε_{MAX}	ε_{MSQ}	$\varepsilon_{\mathcal{L}^2}$	$\frac{\varepsilon_{MAX}(N/2)}{\varepsilon_{MAX}(N)}$
1792	32×32	1.6e-4	5.2e-5	2.8e-4	
3584	64×64	4.9e-6	1.7e-6	9.0e-6	32.7
7168	128×128	1.5e-7	5.2e-8	2.8e-7	32.7
14336	256×256	5.2e-9	1.6e-10	8.8e-9	28.8
28672	512×512	2.2e-10	5.3e-11	2.8e-10	23.6

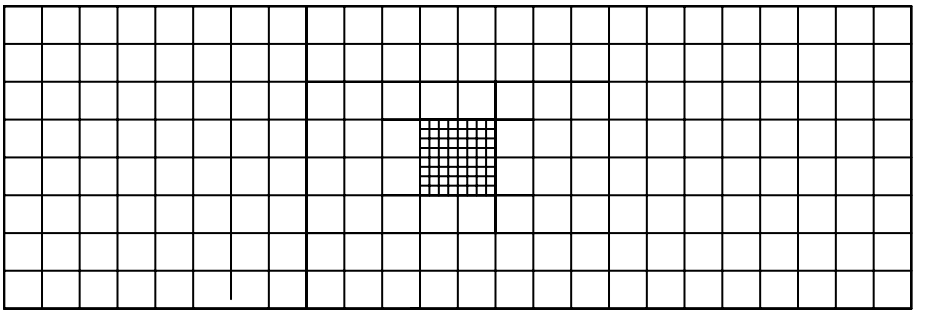


FIG. 10. *The grid in the small box in the middle is four times denser in each direction than in the remaining domain.*

TABLE 20
The errors for the Poisson equation with the exact solution being a steep Gaussian with $\alpha = 25$; the central subdomain is the 16th part of the middle section.

$N_x \times N_y$ in middle section	$N_x \times N_y$ in side section	ε_{MAX}	ε_{MSQ}	$\varepsilon_{\mathcal{L}^2}$	$\frac{\varepsilon_{MAX}(N/2)}{\varepsilon_{MAX}(N)}$
2032	32×32	4.8e-5	1.3e-5	9.4e-5	
4064	64×64	2.7e-6	7.3e-7	5.1e-6	17.8
8128	128×128	9.7e-8	2.6e-8	1.8e-7	27.8
16256	256×256	3.4e-9	8.3e-10	5.7e-9	28.5
32512	256×256	1.3e-10	2.6e-11	1.8e-10	26.2
65024	512×512	5.1e-12	7.7e-13	5.3e-12	25.5

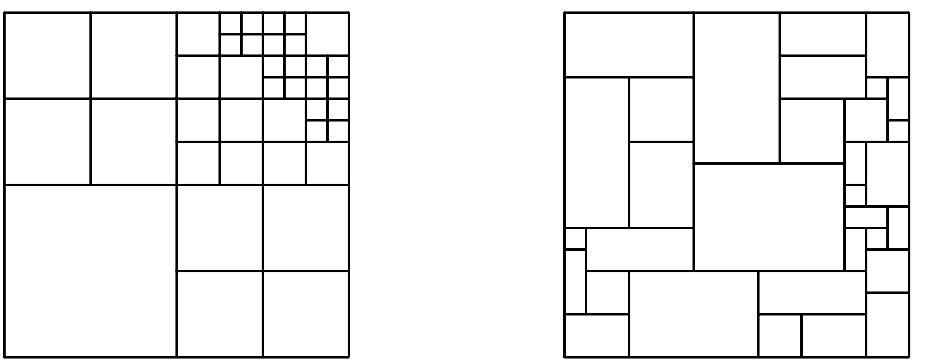


FIG. 11. *The domain is divided into subdomains appropriate to the sparsity pattern of the RHS. The domains can be square or of any arbitrary rectangular form.*

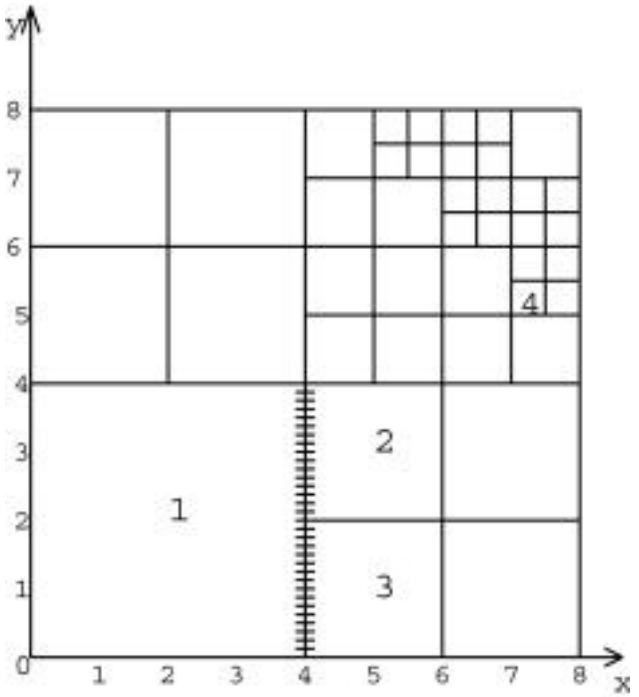


FIG. 12. *The domain is divided into square subdomains with different resolutions in each subdomain. Influence of double layer introduced at the boundary between boxes* 1 *and* 2,3 *evaluated at the left and bottom of the boundaries of box* 4.

use the middle point integration. Multiwavelet coefficients of the decomposed matrix which are below the threshold are ignored.

Tables 21–26 describe cases where the number of points on the interface with the double layer is not equal to the number of points on the boundary where the effect is computed. We calculate the effect of the double layer at the closest parallel boundary of the chosen box.

We compute the effect of the double layer on the interface between boxes 1 and 2,3 on the left side of box 4. We use the middle point integration. The matrix representing the kernel of the integral

$$\int_0^4 \frac{y-4}{(y-4)^2+(x-5)^2}\tau(x)\,dx$$

is decomposed into the multiwavelet bases with four vanishing moments. Elements with absolute values which are below the threshold are ignored. We first take the same number of points on both interfaces.

5.2.2. Matching of arbitrary rectangular boxes. In this section we consider the case when the domain is divided into arbitrary rectangular subdomains (see Figure 13).

Consider a domain which is divided into arbitrary rectangular subdomains and compute the influence of the double layer introduced at one of the interfaces on a certain boundary (for example, the boundary of box 4). Consider the influence of the double layer at the interface between box 1 and 2,3 on the left and the upper

TABLE 21

The number of points on the interface is given in the row labeled "number of points." The number of points on the interface is equal to the number of points on the boundary where the effect of the double layer is computed. The boundary is parallel to the interface where the double layer is introduced (the influence of the double layer between boxes 1 *and* 2,3 *is computed on the left boundary of box* 4*). "Nonzero coefficients" refers to the number of elements above the threshold in the matrix which represents the influence of the double layer. This number is considered the degree of sparsity of the matrix. The average number of nonzero coefficients per row is also computed. The multiwavelet basis is chosen with* 4 *vanishing moments. The accuracy is computed for a density of the double layer* $\exp\{-5(x-0.2)^2\} + \exp\{-50(x-0.7)^2\}$.

Threshold	Number of points	64	128	256	512	1024	2048
1e-5	nonzero coef.	7	6	7	6	7	18
	nonzero coef. per row	0.11	0.05	0.03	0.007	0.01	0.009
	accuracy	2.5e-5	7.6e-5	6.5e-5	1.0e-4	2.0e-4	2.2e-4
1e-6	nonzero coef.	13	16	11	26	11	12
	nonzero coef. per row	0.20	0.13	0.04	0.05	0.01	0.006
	accuracy	4.1e-6	5.2e-6	5.1e-6	1.1e-5	1.2e-5	1.7e-5
1e-7	nonzero coef.	19	20	23	26	68	69
	nonzero coef. per row	0.30	0.16	0.09	0.05	0.07	0.03
	accuracy	2.5e-6	5.8e-7	3.4e-7	3.4e-7	1.6e-6	5.7e-7
1e-8	nonzero coef.	24	32	45	43	42	67
	nonzero coef. per row	0.38	0.36	0.18	0.08	0.04	0.03
	accuracy	2.6e-6	6.7e-7	2.1e-7	1.2e-7	4.9e-8	6.7e-8
1e-9	nonzero coef.	37	44	40	118	108	92
	nonzero coef. per row	0.58	0.34	0.16	0.23	0.11	0.04
	accuracy	2.6e-6	6.4e-7	1.7e-7	4.7e-8	1.8e-8	1.8e-8

TABLE 22

The same computation as in Table 21*, but here the number of points on the interface* N_1 *is twice the number* N_2 *of points on the boundary where the influence of the double layer is computed.*

Threshold	N_1	128	256	512	1024	2048	4096
	N_2	64	128	256	512	1024	2048
1e-5	nonzero coef.	6	7	7	8	15	-
	accuracy	5.7e-5	8.0e-5	1.4e-4	2.1e-4	2.7e-4	8.0e-3
1e-6	nonzero coef.	13	10	22	10	10	38
	accuracy	2.9e-6	4.8e-6	6.8e-6	1.2e-5	1.4e-5	3.6e-5
1e-7	nonzero coef.	19	21	23	56	59	77
	accuracy	1.1e-6	6.3e-7	5.3e-7	1.7e-7	1.6e-6	1.6e-6
1e-8	nonzero coef.	28	35	35	32	55	332
	accuracy	6.5e-7	2.0e-7	1.0e-7	5.1e-7	6.9e-8	4.5e-7
1e-9	nonzero coef.	37	33	96	97	81	124
	accuracy	6.4e-7	1.6e-7	5.0e-8	2.4e-8	2.6e-8	2.5e-8

TABLE 23

The same computation as in Table 21*, but here the number of points on the interface* N_1 *is* 4 *times greater than the number* N_2 *of points on the boundary where the influence of the double layer is computed.*

Threshold	N_1	256	512	1024	2048	4096
	N_2	64	128	256	512	1024
1e-5	nonzero coef.	6	6	6	12	-
	accuracy	3.9e-5	1.1e-4	1.9e-4	2.3e-4	7.9e-3
1e-6	nonzero coef.	10	18	10	10	25
	accuracy	5.2e-6	6.9e-6	1.3e-5	1.4e-5	3.3e-5
1e-7	nonzero coef.	16	18	39	48	60
	accuracy	3.0e-7	3.9e-7	3.1e-7	5.3e-7	1.6e-6
1e-8	nonzero coef.	33	32	28	45	268
	accuracy	2.9e-7	2.4e-7	1.6e-7	1.9e-7	2.6e-7
1e-9	nonzero coef.	30	69	75	63	89
	accuracy	1.6e-7	4.5e-8	1.5e-8	1.8e-8	1.7e-8

TABLE 24

The same computation as in Table 21*, but here the number of points on the interface N_1 is two times less than the number N_2 of points on the boundary where the influence of the double layer is computed.*

Threshold	N_1	32	64	128	256	512	1024
	N_2	64	128	256	512	1024	2048
1e-5	nonzero coef.	8	9	7	8	7	9
	accuracy	1.4e-5	3.5e-5	6.8e-5	8.2e-5	1.0e-4	1.8e-4
1e-6	nonzero coef.	13	17	19	13	36	13
	accuracy	1.1e-5	4.4e-6	5.3e-6	1.5e-5	1.6e-5	2.0e-5
1e-7	nonzero coef.	22	20	21	25	28	81
	accuracy	1.0e-5	2.4e-6	5.1e-7	2.6e-7	2.6e-7	2.1e-6
1e-8	nonzero coef.	25	27	32	50	49	49
	accuracy	1.0e-5	2.6e-6	6.7e-7	1.9e-7	1.1e-7	4.5e-8
1e-9	nonzero coef.	41	41	60	52	147	141
	accuracy	1.0e-5	2.6e-6	6.5e-7	1.7e-7	5.3e-8	2.3e-8

TABLE 25

The same computation as in Table 21*, but here the number of points on the interface N_1 is four times less than the number N_2 of points on the boundary where the influence of the double layer is computed.*

Threshold	N_1	16	32	64	128	256	512
	N_2	64	128	256	512	1024	2048
1e-5	nonzero coef.	8	9	12	9	10	8
	accuracy	3.6e-5	3.6e-5	4.0e-5	8.3e-5	6.8e-5	1.2e-4
1e-6	nonzero coef.	12	12	18	19	11	43
	accuracy	3.3e-5	1.0e-5	3.8e-6	3.5e-6	4.5e-6	7.7e-6
1e-7	nonzero coef.	25	27	26	26	34	38
	accuracy	3.3e-5	1.0e-5	2.4e-6	5.7e-7	7.6e-7	9.2e-7
1e-8	nonzero coef.	31	30	34	37	65	54
	accuracy	3.3e-5	1.0e-5	2.6e-6	6.6e-7	1.9e-7	1.0e-7
1e-9	nonzero coef.	36	44	43	50	40	170
	accuracy	3.3e-5	1.0e-5	2.6e-6	6.4e-7	1.6e-7	4.3e-8

TABLE 26

The boundary is perpendicular to the interface where the double layer is introduced. (*The influence of the double layer between boxes* 1 *and* 2,3 *is computed on the bottom boundary of box* 4.) *Notation and parameters are the same as in Table* 21.

Threshold	Number of points	64	128	256	512	1024	2048
1e-5	nonzero coef.	23	24	15	18	19	10
	per row	0.36	0.19	0.06	0.035	0.02	0.005
	accuracy	3.5e-5	2.6e-5	7.7e-5	1.1e-4	4.7e-5	1.9e-4
1e-6	nonzero coef.	19	25	36	22	180	71
	per row	0.30	0.20	0.14	0.04	0.18	0.03
	accuracy	1.1e-5	6.6e-6	4.3e-6	6.4e-6	3.1e-5	3.4e-5
1e-7	nonzero coef.	35	41	50	106	68	46
	per row	0.55	0.32	0.20	0.21	0.07	0.02
	accuracy	9.0e-6	2.3e-6	5.9e-7	1.0e-6	1.1e-6	8.3e-7
1e-8	nonzero coef.	55	74	87	114	167	311
	per row	0.86	0.58	0.34	0.22	0.16	0.15
	accuracy	9.1e-6	2.3e-6	5.9e-7	2.1e-7	1.2e-7	8.3e-7
1e-9	nonzero coef.	100	113	167	160	268	306
	per row	1.56	0.88	0.65	0.31	0.26	0.15
	accuracy	9.1e-6	2.3e-6	5.7e-7	1.4e-7	4.8e-8	3.4e-8

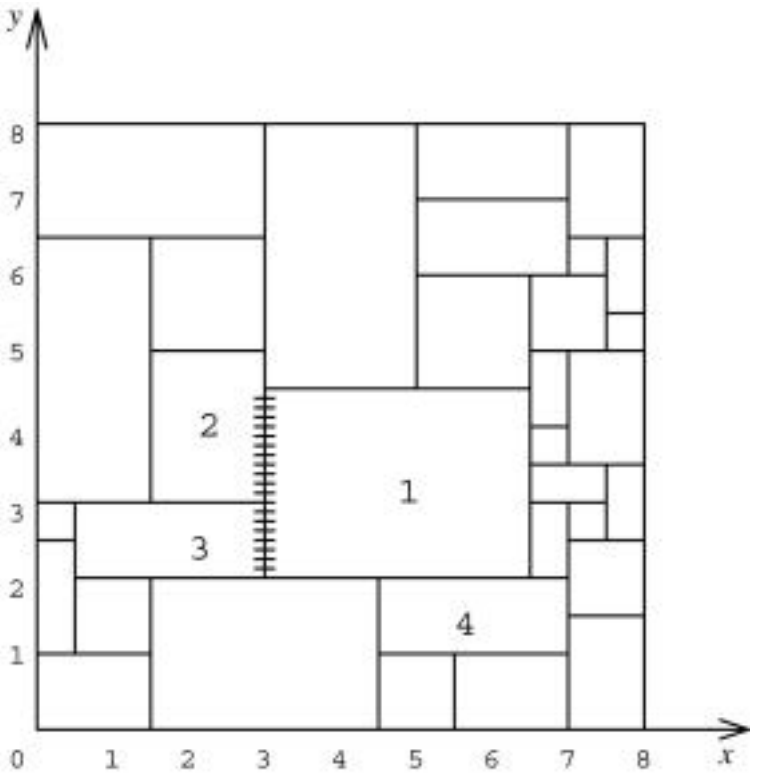


FIG. 13. *The domain is divided into some subdomains of an arbitrary rectangular form with different resolution. We compute the influence of the double layer introduced on the boundary between boxes* 1 *and* 2,3. *The influence is evaluated at the left and top boundaries of box* 4.

boundaries of box 4. It is expressed by the integral

$$\int_2^5 \frac{1.5}{2.25 + (x-y)^2} \tau(x)\, dx, \tag{5.1}$$

at the left boundary of box 4, and

$$\int_2^5 \frac{y-3}{(x-2)^2 + (y-3)^2} \tau(x)\, dx, \tag{5.2}$$

at the top boundary of box 4, where the density of the double layer was chosen as

$$\tau(x) = \exp\left\{5(x-3.2)^2\right\} + \exp\left\{50(x-2.7)^2\right\} + x(x-1). \tag{5.3}$$

Table 27 presents the sparsity of matrices representing the kernels in the multiwavelet bases with four vanishing moments and the computation error for the left boundary of box 4. Table 28 presents the same data for the top boundary. We note that in the computations the intermediate results below a certain threshold were omitted. The error is stipulated by the error of the middle point integration ($\sim (1/n)^2$), which is dominant for a small number of points n and the error of "threshold truncation," which is dominant for large n and thresholds. (In the latter case the round-off error also grows.) Therefore, for each n such a threshold exists for which the error is minimal.

TABLE 27

The boundary is parallel to the interface where the double layer is introduced. (The influence of the double layer between boxes 1 *and* 2,3 *is computed on the left boundary of box* 4.*) Notation and parameters are the same as in Table* 21*, except for the density of the double layer which was chosen as* $\tau(x) = \exp 5\{(x-3.2)^2\} + \exp\{50(x-2.7)^2\} + x(x-1)$*. The number of points is equal on the interface and the boundary where the effect is computed.*

Threshold	Number of points	64	128	256	512	1024	2048
1e-5	nonzero coef.	34	40	52	76	104	146
	per row	0.53	0.31	0.20	0.15	0.10	0.07
	accuracy	1.6e-4	1.2e-4	3.1e-4	8.8e-4	3.8e-3	4.4e-3
1e-6	nonzero coef.	68	98	129	104	158	227
	per row	1.1	0.77	0.50	0.20	0.15	0.11
	accuracy	1.4e-4	1.5e-4	2.1e-4	2.4e-4	2.2e-4	2.1e-4
1e-7	nonzero coef.	101	113	191	264	237	274
	per row	1.57	0.88	0.75	0.52	0.23	0.13
	accuracy	6.5e-5	1.7e-5	1.8e-5	2.7e-5	3.2e-5	3.4e-5
1e-8	nonzero coef.	200	243	275	414	717	784
	per row	3.13	1.90	1.07	0.81	0.70	0.38
	accuracy	6.5e-5	1.7e-5	4.5e-6	3.4e-6	3.6e-6	4.3e-6
1e-9	nonzero coef.	351	397	451	477	865	1881
	per row	5.48	3.10	1.76	0.93	0.84	0.91
	accuracy	6.5e-5	1.6e-5	4.2e-6	1.2e-6	7.0e-7	6.8e-7

TABLE 28

The boundary is perpendicular to the interface where the double layer is introduced. (The influence of the double layer between boxes 1 *and* 2,3 *is computed on the top boundary of box* 4.*) Notation and parameters are the same as in Table* 27*.*

Threshold	Number of points	64	128	256	512	1024	2048
1e-5	nonzero coef.	43	52	67	88	84	120
	per row	0.67	0.41	0.26	0.17	0.08	0.06
	accuracy	1.1e-3	1.5e-3	1.6e-3	1.6e-3	2.0e-3	9.2e-3
1e-6	nonzero coef.	81	90	103	126	308	364
	per row	1.27	0.70	0.40	0.25	0.30	0.18
	accuracy	2.2e-4	2.1e-4	2.2e-4	2.3e-4	3.0e-4	3.3e-4
1e-7	nonzero coef.	146	151	223	282	351	557
	per row	2.28	1.18	0.87	0.55	0.34	0.27
	accuracy	6.4e-5	3.6e-5	2.8e-5	2.3e-4	2.7e-5	2.7e-5
1e-8	nonzero coef.	263	271	314	626	830	930
	per row	4.11	2.12	1.22	1.22	0.81	0.45
	accuracy	6.5e-5	1.7e-5	5.3e-6	2.5e-6	4.7e-6	6.9e-6
1e-9	nonzero coef.	447	511	561	685	1637	2292
	per row	6.98	4.0	2.19	1.34	1.59	1.11
	accuracy	6.4e-5	1.6e-5	4.1e-6	1.1e-6	4.4e-7	6.9e-7

6. Parallel implementation. The multiple domain algorithm was implemented on MIMD parallel computers with minor modifications. We assume that the topology of the domain and subdomains is described by Figure 4. Each domain is assigned to a processor. In the parallel implementations the Dirichlet problem was solved in each subdomain and single layers were introduced afterwards to match the first derivatives. The parallel algorithm includes the following steps.

1. *Preprocessing step.*
   Initially, the kernel matrices are downloaded into the memory of each processor. This step also contains obtaining sparse representation for these matrices in multiwavelet bases.
2. *Solution of the nonhomogeneous equation in each subdomain.*
   In each subdomain the RHS of the Poisson equation is defined and the smooth

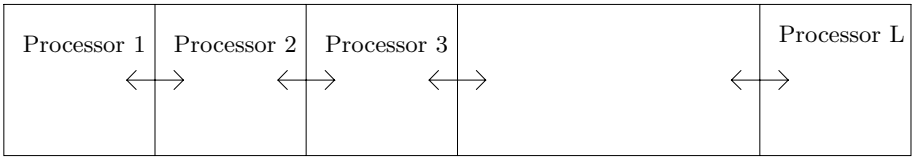


FIG. 14. *The domain is decomposed into L subdomains. Each domain is assigned to a certain processor. Double/single layers are introduced on the $L-1$ interfaces. Adjacent processors communicate to compute the values of the layers.*

boundary conditions are determined. Then the particular solution of the nonhomogeneous equation with the above boundary conditions is found in each subdomain.

3. *Computation and transmission of discontinuities.*
   In order to remove the discontinuities in the derivatives between the subdomains, the differences between the first derivatives on the boundaries are found. This requires communication between adjacent processors only. Then, each single layer density is represented in the multiwavelet basis. The multiwavelet coefficients are transmitted from each processor to each adjacent processor in order to compute the influence of all the double layers on its boundaries. In practice this is implemented as follows. Each processor communicates only with its neighbors: it obtains the information on the layer from its left neighbor and passes the information to its right neighbor. After $L-1$ communication steps, all the processors have the complete information on the layers. Only the sparse data structures that contain the coefficients above the threshold are transmitted. Thus, the sparsity of the multiwavelet coefficients vector reduces the amount of the communication.
4. *Computation of final boundary conditions in subdomains.*
   To avoid the communication bottleneck at the last (global) step and reduce the number of computations, the above algorithm can be improved in the following way. Two last steps of the original (nonimproved) algorithm are changed by this and the following ones. Each processor computes the influence of each double layer on the boundaries of the domain. The correction at the global boundaries, which is necessary to satisfy the original boundary conditions, is computed. Each processor then computes the corrections at its own boundaries, as a consequence of the solution of the global Laplace equation with the corrected boundary conditions. We combine these boundary condition corrections with the previously computed corrections due to the matching step. The accumulation of all the above corrections determines the final local boundary conditions.
5. *Solution of the Laplace equations in subdomains.*
   The Laplace equation is solved in each subdomain with the boundary conditions which were computed at the previous step. Combining the result with the solution of the Poisson equation leads to a smooth global solution satisfying the initial (global) boundary conditions.

The current parallel implementation assumes the geometry of Figure 14.

The rectangular domain is divided into L boxes in a line that correspond to L processors. The algorithm was performed on the parallel SGI computer of type Origin 2000 and IBM SP2. The parallel program differs from the serial one only by

TABLE 29

Speed-up and efficiency for the parallel implementation of the solution of the Poisson equation with the Dirichlet boundary conditions on IBM SP2 parallel computer; the domain decomposition parallel program is compared to the multiple domain serial program.

Number of points in each domain	Number of processors	Time T_{ser}, sec	Time T_{par},sec	Speed-up	Efficiency
128×128	4	20.4	5.2	3.92	0.98
	6	31.1	5.6	5.55	0.93
	8	45.1	6.3	7.2	0.89
	10	56.3	6.5	8.6	0.86
	12	68.5	6.7	10.2	0.85
	16	93.4	7.0	13.3	0.83
256×256	4	77.2	19.4	3.98	0.99
	6	143.1	24.3	5.9	0.98
	8	192.8	26.1	7.4	0.92
	10	247.5	26.9	9.2	0.92
	12	302.2	27.0	11.2	0.93
	16	396.4	27.1	14.6	0.91

TABLE 30

Speed-up and efficiency for the parallel implementation of the solution of the Poisson equations with the Dirichlet boundary conditions on SGI parallel computer; the domain decomposition parallel program is compared to the multiple domain serial program, each subdomain contains 256×256 *grid points.*

The number of processors L	Time T_{ser}, sec	Time T_{par} sec	Speed-up	Efficiency
4	16.2	4.5	3.6	0.9
6	26.9	5.1	5.3	0.88
8	35.7	5.3	6.7	0.84
10	47.8	5.7	8.4	0.84
12	58.3	6.0	9.7	0.81
16	84.7	6.6	12.8	0.80

the communication constructions from MPI.

Each processor communicates only with its neighbors: it obtains the information on the layer from its left neighbor and passes the information to its right neighbor. After $L-1$ communication steps all the processors possess the complete information on the layers. Note that this is not an efficient communication scheme for SP2 and even one of the worst for the SGI computer which is a shared memory system. In the future we plan to use the native collective communication like gather, scatter, etc.

Let T_{par} be the total computation time for the parallel program. The most important measures of performance for a parallel program are the speed-up and efficiency. The speed-up S is defined as the ratio of the time, T_{ser}, required to run the serial program for a problem of a given size, to the time, T_{par}, required for an equivalent calculation on a parallel computer. The efficiency is equal to $T_{ser}/(T_{par}L)$, where L is the number of processors.

First, we investigate the parallel implementation of the multiple domain algorithm on two computers: IBM SP2 and SGI Origin 2000. The speed-up is computed with respect to the serial implementation of the same algorithm (see Tables 29, 30).

A question may be raised concerning the serial time used in the definition of speed-up. Often the speed-up is computed using the best serial algorithm; therefore, we should investigate the performance of the multiple domain algorithm when compared to the single domain algorithm. The latter algorithm was described in [4] and uses the

TABLE 31

The theoretical ratio of times required for the implementation of the serial programs for the primary solution of the Poisson equation with and without domain decomposition T_{sd}/T_{md}. Here N is the number of grid points in subdomains in each direction; N_ϵ is the number of extension points.

N	N_ϵ	Number of subdomains L									
		4	6	8	12	16	24	50	100	10^3	10^5
128	64	0.68	0.65	0.64	0.63	0.63	0.64	0.66	0.68	0.78	0.99
256	128	0.67	0.64	0.63	0.62	0.62	0.63	0.64	0.66	0.75	0.93
1024	512	0.66	0.63	0.62	0.61	0.61	0.60	0.62	0.63	0.71	0.85
128	10	1.01	1.04	1.05	1.08	1.10	1.14	1.19	1.25	1.45	1.85
256	10	1.06	1.08	1.10	1.13	1.16	1.19	1.24	1.30	1.49	1.87
1024	10	1.08	1.11	1.13	1.16	1.18	1.20	1.26	1.30	1.47	1.79

first two steps of the multiple domain algorithm implemented in one global domain.

Consider the theoretical performance of the first step of the multiple domain algorithm (solving the Poisson equation in all the subdomains as compared to its solution in one global domain). For example, let the domain $[0, L] \times [0, 1]$ be divided into L subdomains, $N \times N$ points in each, and let the extension be N_ϵ in each direction. Then the solution of the Poisson equation in subdomains requires

$$O(L(N + 2N_\epsilon)^2 \log(N + 2N_\epsilon)$$

operations, the solution of the Laplace equation in subdomains requires $O(LN^2 \log N)$, the computation of the influence of layers—$O(L^2 N \log N)$, and the global solution of the Laplace equation—$O(LN^2(\log N + \log(LN)))$. On the other hand, the primary solution of the Poisson equation in the global domain requires

$$O((N + 2N_\epsilon)(NL + 2N_\epsilon)(\log(NL + 2N_\epsilon) + \log(N + 2N_\epsilon)))$$

operations. Let us compare the time T_{md} above for the multiple domain algorithm with the time T_{sd} for the same step in the single domain. This ratio is equal to

$$\frac{T_{sd}}{T_{md}} = \frac{(NL + 2N_\epsilon)(\log(NL + 2N_\epsilon) + \log(N + 2N_\epsilon))}{2L(N + 2N_\epsilon) \log(N + 2N_\epsilon)}. \tag{6.1}$$

In the case $N_\epsilon = N/2$ one gets

$$\frac{T_{sd}}{T_{md}} = \frac{(L + 1)(\log(L + 1) + 2 \log N + 1)}{4L(\log N + 1)}.$$

Note that the number of points $N_\epsilon = N/2$ chosen for extension is excessive (see Example 3). Table 31 presents the theoretical efficiency of the local Poisson step for $N_\epsilon = N/2, 10$ and various values of L and N.

It is to be emphasized that the efficiency of the algorithm (see Table 31) can exceed one. This means that the multiple domain algorithm becomes more efficient than the global algorithm. This effect was employed in [9]. In our case the local algorithm for the solution of the Poisson equation requires $O(N^2 \log N)$ operations. The $\log N$ factor does not scale in proportion to the area and is the cause of the deterioration of the one domain case.

Now we investigate the efficiency of the multiple domain algorithm when the number of extension points is 11 when compared to the single domain algorithm. The measurements of the running times for two serial programs were implemented on an SGI computer. The results are summarized in Table 32 and Figure 15.

TABLE 32

The comparison of the running times for two serial programs: multiple domain and single domain. The number of points in each subdomain is 128×128, $N_\epsilon = 11$.

The number of subdomains L	Time T_{md} for single domain alg.	Time T_{md} for multiple domain alg.
4	1.9	3.6
6	3.0	5.4
8	5.3	7.4
10	6.1	9.2
12	7.2	11.1
16	11.7	14.9
20	16.4	19.3
25	25.9	25.1
30	34.5	30.7
40	53.6	44.1
50	72.3	56.8
60	97.9	73.1
80	138.8	106.5
100	182.9	141.3

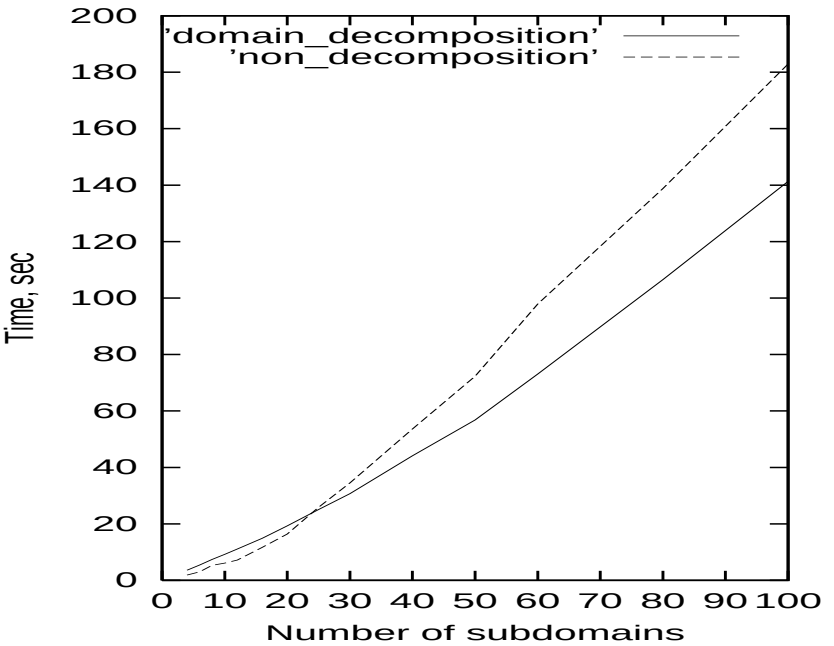


FIG. 15. *The comparison of the times necessary for the implementation of two serial programs on SGI computer: with and without domain decomposition. The number of points in each subdomain is* 128×128*; the number of extension points is* 11.

We can see from Table 32 that the domain decomposition algorithm is competitive as the number of subdomains exceeds 25. This can be achieved either by the choice of the parallel computer with more than 25 processors or by the additional decomposition of the domains assigned to a certain processor.

It is to be emphasized that the information transmitted between the processors is not more than $O(N)$, while the number of grid points is $O(N^2)$. Thus the communication did not degrade the efficiency (see Tables 29 and 30) of the parallel algorithm

in our implementation. However for large systems the communication and the computation of the influences of the matching layers increases as L^2 does (in a naive implementation) and may affect the efficiency. On the other hand, we have investigated in previous sections the compression effect of the multiwavelet representation. It results in a reduction of the number of coefficients required for the computation of the influence of the matching layers. Thus the growth of the effort required for matching will be considerably reduced. In the present computer implementations this effect was not utilized in view of the rather modest requirements of the task.

However the present implementation of the parallel does not employ a sparse data structure. If we use the sparse data structure for computing effect of the layers, the running time for this part of the algorithm will be considerably reduced. The evaluation of this effect for double layers was given in the previous sections. For a single layer, the results were presented in Table 4.

7. Summary and conclusions. We developed an algorithm for fast computation of the influence of the double/single layer on domain interfaces. This is a part of a more general algorithm for the solution of elliptic equations in a domain which is decomposed into subdomains. The computation of the influence of singularity layers is necessary in the matching step. The present algorithm enjoys the following properties:

1. The algorithm is adaptive since the resolution in a subdomain depends on the smoothness of the RHS. Moreover, the domain decomposition can be adaptive as in [9].
2. The kernels of the integrals which represent singularity layers are sparse in multiwavelet bases. The sparse matrix-vector multiplication reduces the number of operations to $O(N)$, where N is the number of grid points on the interface. In fact, the number of operations is even less if the density of the double layer is smooth and both the double layer and the interface have a simple geometry. The representation of functions in wavelet/multiwavelet bases provides an automatic adaptation to its "problematic," such as nonsmooth, oscillating points. The representation of the kernels of the integrals in multiwavelet bases is done once during the preprocessing step. This is possible since they depend only on the geometry of subdomains. These are the reasons why the algorithm is fast.
3. The accuracy of the algorithm is preserved near the points where the double layer intersects the interface due to the subtraction technique which was applied. Here we used the fact that we know the analytic expression for the kernel of the integral.
4. The algorithm can be evaluated in parallel. The parallelization is effective since in the computational process only few coefficients, which are representatives of layers densities, have to be transmitted between neighboring processors.

The method can be extended to 3-D cases when a double layer is introduced on the plane. This will become a step in the algorithm for fast solution of 3-D elliptic equations using domain decomposition. The first step for a fast 3-D solution of the equation in each subdomain is described in [8].

Appendix: Construction and properties of discrete multiwavelet bases. Let $S = \{x_1, x_2, \ldots, x_n\} \subset \Re$ be an increasing set of n distinct points (discretization). For simplicity we assume $n = k \cdot 2^l$, where k and l are positive integers. The basis constructed enjoys the following properties:

1. All but k basis vectors have k vanishing moments.

A vector $b = (b_1, b_2, \ldots, b_n)$ is said to have k vanishing moments if

$$\sum_{i=1}^{n} b_i x_i^j = 0, \ j = 0, 1, \ldots, k-1.$$

2. The basis vectors are nonzero on different scales.

These properties of local support and vanishing moments lead to efficient representation of functions which are smooth except at a finite set of singularities. Besides, the conditions of "local" support and zero moments determine the basis vectors uniquely (up to the sign) if, out of the k vectors nonzero on $x_1, \ldots, x_{2k}$, we require that one has k vanishing moments, a second $k+1$, a third has $k+1$, and so on, and the kth has $2k-1$ vanishing moments. When constructing such bases we define the $2k \times 2k$ moment matrices, for $i = 1, \ldots, n/(2k)$ by the formula

$$M_{1,i} = \begin{pmatrix} 1 & x_{s_i+1} & \ldots & x_{s_i+1}^{2k-1} \\ 1 & x_{s_i+2} & \ldots & x_{s_i+2}^{2k-1} \\ \vdots & & & \vdots \\ 1 & x_{s_i+2} & \ldots & x_{s_i+2k}^{2k-1} \end{pmatrix},$$

where $s_i = (i-1)2k$. For a $(2k \times 2k)$-matrix V, we let V^U and V^L denote two $k \times 2k$-matrices, V^U consisting of the upper k rows and V^L the lower k rows of V, i.e.,

$$V = \begin{pmatrix} V^U \\ V^L \end{pmatrix}.$$

The first basis matrix U_1 is given by the formula

$$U_1 = \begin{pmatrix} U_{1,1}^L & & & \\ & U_{1,2}^L & & \\ & & \ddots & \\ & & & U_{1,n/(2k)}^L \\ U_{1,1}^U & & & \\ & U_{1,2}^U & & \\ & & \ddots & \\ & & & U_{1,n/(2k)}^U \end{pmatrix},$$

where the transposed $U_{1,i}^T$ is obtained by the column-by-column Gram–Schmidt orthogonalization of M_1, i.

Further, the following basis matrices are constructed successively. The jth basis matrix, $j = 2, \ldots, \log_2(n/k)$, is a product $U_j \ldots U_1$, with U_j defined by the formula

$$U_j = \begin{pmatrix} I_{n-n/2^{j-1}} & \\ & U_j' \end{pmatrix},$$

where I_m is the $m \times m$ identity matrix and U'_j is given by the formula

$$U_j = \begin{pmatrix} U^L_{j,1} & & & & \\ & U^L_{j,2} & & & \\ & & \ddots & & \\ & & & U^L_{j,n/(2^j k)} & \\ U^U_{j,1} & & & & \\ & U^U_{j,2} & & & \\ & & \ddots & & \\ & & & U^U_{j,n/(2^j k)} & \end{pmatrix},$$

where $U^T_{j,i}$ is obtained by the orthogonalization of $M_{j,i}$, and

$$M_{j,i} = \begin{pmatrix} U^U_{j-1,2i-1} & M_{j-1,2i-1} \\ U^U_{j-1,2i} & M_{j-1,2i} \end{pmatrix}.$$

The final basis matrix $U = U_l \dots U_1$, where $l = \log_2(n/k)$, represents the multiwavelet basis of parameter k on $x_1, \dots, x_n$.

It is apparent that the application of the matrix U to an arbitrary vector of length n may be accomplished in order $O(n)$ operations by the application of $U_1, \dots, U_l$ in turn. Similarly, $U^{-1} = U^T$ may be applied to a vector in order $O(n)$ operations. The construction and application of discrete multiwavelets is described in detail in [2].

Acknowledgments. The authors are very grateful to Dr. L. Ioffe for her contribution to the parallel implementation of the algorithm and to the referees for their valuable remarks.

REFERENCES

[1] B. K. Alpert, *A class of bases in L^2 for the sparse representation of integral operators*, SIAM J. Math. Anal., 24 (1993), pp. 246–262.

[2] B. Alpert, G. Beylkin, R. Coifman, and V. Rokhlin, *Wavelet-like bases for the fast solution of second-kind integral equations,* SIAM J. Sci. Comput., 14 (1993), pp. 159–184.

[3] A. Averbuch, M. Israeli, and L. Vozovoi, *On fast direct elliptic solver by modified Fourier method*, Numer. Algorithms, 15 (1997), pp. 287–313.

[4] A. Averbuch, M. Israeli, and L. Vozovoi, *A fast Poisson solver of arbitrary order accuracy in rectangular regions*, SIAM J. Sci. Comput., 19 (1998), pp. 933–952.

[5] A. Averbuch, G. Beylkin, R. Coifman, and M. Israeli, *Multiresolution Solution of Elliptic and Parabolic PDEs,* The Samuel Neaman Workshop on Signal and Image Representation in Combined Space, Technion, Haifa, Israel, May 8–11, 1994, in Signal and Image Representation in Combined Spaces, Y. Zeevi and R. Coifman, eds., Academic Press, San Diego, CA, 1998, pp. 341–359.

[6] A. Averbuch, G. Beylkin, R. Coifman, P. Fischer, M. Israeli, *Adaptive solution of multidimensional PDEs via tensor product wavelet decomposition*, Appl. Comput. Harmon. Anal., submitted.

[7] G. Beylkin, R. Coifman, and V. Rokhlin, *Fast wavelet transforms and numerical algorithms* I, Comm. Pure Appl. Math., 44 (1991), pp. 141–183.

[8] E. Braverman, M. Israeli, A. Averbuch, and L. Vozovoi, *A fast 3-D Poisson solver of arbitrary order accuracy*, J. Comput. Phys., 144 (1998), pp. 109–136.

[9] L. Greengard and J.-Y. Lee, *A direct adaptive Poisson solver of arbitrary order accuracy*, J. Comput. Phys., 125 (1996), pp. 415–424.

[10] M. Israeli, L. Vozovoi, and A. Averbuch, *Spectral multi-domain technique with Local Fourier basis,* J. Sci. Comput., 8 (1993), pp. 135–149.

[11] A. Sidi, *A new variable transformation for numerical integration,* Internat. Ser. Numer. Math., 112 (1993), pp. 359–373.

[12] L. Vozovoi, M. Israeli, and A. Averbuch, *Spectral multi-domain technique with Local Fourier basis* II: *Decomposition into cells,* J. Sci. Comput., 9 (1994), pp. 311–326.

SIAM J. SCI. COMPUT.
Vol. 22, No. 3, pp. 1087–1116
© 2000 Society for Industrial and Applied Mathematics

ON SOME APPLICATIONS OF THE SUPERPOSITION PRINCIPLE WITH FOURIER BASIS*

MARC GARBEY†

Abstract. This paper presents several applications of (local) Fourier basis combined with corrector techniques via the superposition principle to compute solutions of boundary value problems. Our methodology is inspired by the well-known corrector technique used in asymptotic singular perturbation theory—see, for example, [W. Eckhaus, *Asymptotic Analysis of Singular Perturbations*, North-Holland, Amsterdam, 1979]. We build solvers for time dependent boundary value problems and/or singular perturbation problems using Fourier expansions combined to additional convenient set of time independent basis functions to enforce the boundary conditions. The algorithms are very well suited for parallel computing because they rely mainly on FFTs and basis functions given analytically or computed (in parallel) once and for all. Our method can be spectral accurate for simple geometry. We obtain in addition interesting new results for boundary value problems with complex geometry. We describe in this paper the implementation of the method and the numerical results for many linear and nonlinear problems: our goal is to test the advantages and the limits of our approach in order to motivate further theoretical investigations. Our method applied to steady one-dimensional problems is similar to the work of Israeli, Vozovoi, and Averbuch [*J. Sci. Comput.*, 8 (1993), pp. 135–149] on domain decomposition with local Fourier basis for the Helmotz problem. However our methodology for time-dependent problem and/or two space dimensions geometry is rather different in its spirit and complements previous investigations of Israeli, Vozovoi, and Averbuch.

Key words. Fourier expansion, superposition principle, singular perturbation, corrector technique, spectral method

AMS subject classifications. 65N06, 42A15, 41A05, 41A25

PII. S1064827597328133

1. Introduction.

Let us consider the linear elliptic problem:

$$L[u] = f \text{ in } \Omega \subset \mathbb{R}^n,\ B[u] = g \text{ on } \partial\Omega, \tag{1}$$

with Ω such that $\Omega \subset [0\ ,\ 2\pi]^n$,

$$L[u] \equiv \sum_{i,j=1,\ldots,n} a_{i,j}(x)\frac{\partial^2 u}{\partial x_i \partial x_j}\ +\ \sum_i b_i(x)\frac{\partial u}{\partial x_i}\ +\ c(x)u,$$

and all datas $a,\ b,\ c,\ f,\ g$ in $C^\infty(\Omega)$. We suppose that (1) has a unique smooth solution. Let us consider the following two problems:

$$\bar{L}[v] = \bar{f}\ in\ [0\ ,\ 2\pi]^n,\ v\ 2\pi \text{ periodic}, \tag{2}$$

and

$$L[w] = 0 \text{ in } \Omega \subset \mathbb{R}^n,\ B[w] = g - B[v] \text{ on } \partial\Omega. \tag{3}$$

In (2), $\bar{L}$ denotes the operator

$$\bar{L}[v] \equiv \sum_{i,j=1,\ldots,n} \bar{a}_{i,j}(x)\frac{\partial^2 v}{\partial x_i \partial x_j}\ +\ \sum_i \bar{b}_i(x)\frac{\partial v}{\partial x_i}\ +\ \bar{c}(x)v$$

*Received by the editors October 1, 1997; accepted for publication (in revised form) January 22, 1999; published electronically September 27, 2000. This work was supported by Région Rhone Alpes.
http://www.siam.org/journals/sisc/22-3/32813.html

†Université Claude Bernard Lyon I, CDCSP, ISTIL, 69622 Villeurbanne cedex, France (mgarbey@cdcsp.univ-lyon1.fr).

and $\bar{f}$ (resp., $\bar{a}$, $\bar{b}$, $\bar{c}$) is a smooth periodic extension of f (resp., a, b, c), i.e., $\bar{f}$ 2π periodic with respect to each variable and $\bar{f} \equiv f$ in Ω. Then clearly the superposition principle gives

$$u \equiv v + w \text{ in } \Omega.$$

From the numerical point of view, the decomposition (2)–(3) of (1) seems to be rather useless at first sight since it increases the complexity of the computation. However, it might be the best way to approximate numerically the solution of (1) for the following reasons:

1. One has to compute (1) for many right-hand sides (rhs) f, *and* w solution of (3) can be approximated using a set of basis functions w_k computed *once and for all*.
2. (1) and (3) have solutions with essentially different behaviors, and therefore one wants to use different discretizations and/or possibly different schemes to compute v and w. A special case corresponds to problems in which w is obtained analytically.
3. $\partial\Omega$ has some complex geometrical structure, w is stiff and vanishes exponentially inside Ω: in such case one wants to approximate w on a local stretched coordinate system.
4. In order to obtain a *parallel* algorithm, we replace (1) with an equivalent *domain decomposition* formulation as follows. Let $\bigcup_{i=1,\dots,N} \Omega_i$ be a nonoverlapping domain decomposition of Ω:

$$L[u_i] = f \text{ in } \Omega_i,\ B[u_i] = g \text{ on } \partial\Omega \bigcap \Omega_i,\ i = 1, \dots, N, \tag{4}$$

$$u_i = u_j \text{ on } \partial\Omega_i \bigcap \partial\Omega_j,\ i, j = 1, \dots, N, i \neq j, \tag{5}$$

$$\frac{\partial}{\partial n} u_i = \frac{\partial}{\partial n} u_j \text{ on } \partial\Omega_i \bigcap \partial\Omega_j,\ i, j = 1, \dots, N, i \neq j. \tag{6}$$

The analogue of the decomposition (2) + (3) for each subproblem (4), (5), and (6) is written:

$$\bar{L}[v_i] = \bar{f}_i \text{ in } [0\ ,\ 2\pi]^n,\ v\ 2\pi \text{ periodic}, \tag{7}$$

and

$$\begin{aligned} L[w_i] = 0 \text{ in } \Omega_i,\ B[w_i] = g - B[v_i] \text{ on } \partial\Omega \bigcap \Omega_i,\ i = 1, \dots, N, \\ v_i + w_i = v_j + w_j \qquad \text{on } \partial\Omega_i \bigcap \Omega_j,\ i, j = 1, \dots, N, i \neq j, \\ \frac{\partial}{\partial n}(v_i + w_i) = \frac{\partial}{\partial n}(v_j + w_j) \qquad \text{on } \partial\Omega_i \bigcap \Omega_j,\ i, j = 1, \dots, N, i \neq j. \end{aligned} \tag{8}$$

Each problem of type (7) can be solved in parallel. To be more precise, each subdomain Ω_i is mapped into a subset of $[0\ ,\ 2\pi]^n$ in order to increase the efficiency of the parallel computation. So (7) is in fact obtained after an appropriate rescaling and shifting of the subdomain Ω_i. Further, we observe that the problems of type (8) are weakly coupled when the terms w_i have a fast decay property in space. We refer to the pioneer work of Israeli, Vozovoi,

and Averbuch in [15], [16], [5] and Vozovoi, Israeli, and Averbuch in [25] that makes intensive use of this approach. Our recent work in [10], [12] gives an example of the good parallel efficiency of this method for combustion problems.

Classical examples that fall in one of the above categories are:

- *Computation of evolution problems.*
  Let us consider formally the time-dependent PDE

$$\frac{\partial u}{\partial t} = \Delta u + F(u) \text{ in } \Omega$$

  with some appropriate initial and boundary conditions. A semi-implicit Euler scheme is written

$$\frac{u^{n+1} - u^n}{dt} = \Delta u^{n+1} + F(u^n).$$

  So one has to solve for each time step:

$$-dt\ \Delta u^{n+1} + u^{n+1} = u^n + dtF(u^n).$$

  Then the arguments (1) and (4) of the previous list may apply to this time-dependent problem.
- *Computation of singular perturbation problems.*
  Let us consider formally the following abstract linear singular perturbation problem:

$$\epsilon \mathcal{A}[u] + \mathcal{B}[u] = f \text{ in } \Omega,\ u = g \text{ on } \partial\Omega.$$

  It is well known that such problems may have boundary layers when a solution of the formal limit problem

$$\mathcal{B}[u] = f \text{ in } \Omega$$

  cannot satisfy the Dirichlet boundary condition $u = g$ on $\partial\Omega$. In the meantime solutions of (2) may have no boundary layer. So the decomposition (2) and (3) of problem (1) can be viewed as a numerical application of the so-called *corrector* technique well known in singular perturbation theory [3], [18]. Then the arguments (2) and (3) of the previous list may apply to this singular perturbation problem.

The goal of this paper is to give some examples of the use of the superposition principle combined to (local) Fourier expansions in numerical schemes for time dependent problems as well as singular perturbation problems. The implementation of the method used well-separated modules: one for FFT to approximate solution of (2), one to build appropriate periodic extensions of rhs of the equations and coefficients of the operators, one to approximate the corrector, solution of (3). To be valuable, the cost of the computation in CPU time should be dominated by the cost of FFT. We will refer then to this method as the *modulefou* method. We are mainly interested to find out how good this method is from the numerical point of view. The benefit of the method for parallel computing is discussed in [10], [11], [12]. So we have looked for test cases for which the method works as well as test cases for which the method does not work. Our conclusion is that the numerical efficiency of the method might be worthwhile in many situations. In recent works, we have been able to obtain a

high-order accurate code to simulate a nontrivial two-dimensional (2D) combustion problem that works for periodic as well nonperiodic boundary conditions with the *same data structure* thanks to the *modulefou* method [8]. The applications that we consider in this paper are less specific and we hope that this preliminary study will serve as a motivation for further theoretical investigations. The plan of the paper is as follows; In section 2, we illustrate the method with the Poisson problem on a disk. In section 3, we give a rigorous presentation of the method in one space dimension. In section 4, we give some examples of applications to time-dependent problem as well as time-dependent singular perturbation problems in one space dimension. We give in particular some details on the implementation of *modulefou* on time-dependent schemes that are critical in order to avoid some Gibbs phenomenon. In section 5 we apply the *modulefou* method to a two space dimension classical singular perturbation problems on a domain with a smooth boundary. We then consider different extensions of the method to time-dependent problems.

2. Poisson problem on a disk. We consider the Poisson problem on a disk $B(0,R)$ of radius $R < \frac{L}{2}$ and center $O = (\frac{L}{2}, \frac{L}{2})$ included in$(O,L)^2$

$$\Delta u \ = \ f \text{ in } B(O,R) \subset (-\pi,\pi)^2, \ u \ = \ g \text{ on } \partial\Omega. \tag{9}$$

A classical numerical approach is to write this problem in polar coordinates (ρ,θ). Then one look for the approximate solution of (9) as

$$u_{N,M} = \sum_{l=-N,N} \sum_{k=0,M} a_{k,l} T_k(\rho) \exp(i\ l\ \theta),$$

with T_k Chebyshev polynomial of degree k. However, this method has two disadvantages: first the grid points are needlessly concentrated at the origin, second the pseudospectral approximation of the corresponding operator in radial direction,

$$\frac{\partial^2}{\partial\rho^2} + \frac{1}{\rho}\frac{\partial}{\partial\rho} + \frac{l^2}{\rho^2} Id,$$

with Dirichlet boundary condition in $\rho = R$ and homogeneous Neuman boundary condition in $\rho = 0$, has a condition number that grows very fast with the degree M of the Chebyshev polynomial. Matlab, for example, gives an estimation of the condition number of order 10^6 for $M = 20$ and $l = 1$; of order 10^9 for $M = 100$ and $l = 1$.

On the contrary, it is extremely easy to compute with Fourier an accurate approximation v_N of a solution of the problem

$$\Delta v \ = \ \bar{f} \text{ in } (O,L)^2, \tag{10}$$

as long as $\bar{f}$ is a smooth periodic given function of $(O,L)^2$. We choose, for example,

$$v_N(x,y) = \sum_{l_1,l_2=-N,N,\ l_1\neq 0,\ l_2\neq 0} - \frac{L^2}{4\,\pi^2}\ \frac{\hat{f}_{l_1,l_2}}{l_1^2+l_2^2} \exp\left(i\ \frac{2\,\pi}{L}\ (l_1\ x\ +\ l_2\ y)\ +\ \frac{1}{4}\hat{f}_{0,0}\ (x^2+y^2)\right),$$

when

$$\bar{f}_N(x,y) = \sum_{l_1,l_2=-N,N} \hat{f}_{l_1,l_2} \exp\left(i\ \frac{2\,\pi}{L}\ (l_1\ x\ +\ l_2\ y)\right).$$

Let

$$h_N(\theta) = \sum_{q=-N,N} \hat{h}_q \exp(i\ q\ \theta)$$

be a discrete Fourier approximation of $g - v_N$ on $\partial B(O,R)$. The solution of the homogeneous problem

$$\Delta w \ = \ 0 \text{ in } B(O,R),\ w = h_N \text{ on } \partial B(O,R) \tag{11}$$

is then given *explicitly* by the formula

$$w_N(\rho,\theta) = \sum_{q=-N,N} \hat{h}_q\ \rho^{|q|} \exp(i\ q\ \theta).$$

The *modulefou* method to compute approximation of (9) then follows these three steps:

1. Construct a smooth periodic extension of f on $(O,L)^2$.
2. Compute v_N via FFTs.
3. Add corrector w_N.

The sensitive part of this algorithm is the numerical construction of $\bar{f}$. We proceed as follows: we assume for simplicity that f is in $C^\infty(B(O,R))$. Let $(x_{l_1}, y_{l_2})_{l_1,l_2=0,2N-1}$ be the uniform grid of $(O,L)^2$ used in Fourier approximation. Let d be positive so that $R + d < \frac{L}{2}$. For every grid point of polar coordinates (ρ,θ) located in the ring $B(0,(R,R+d))$ bounded by the circles of radius R and $R+d$ centered at the origin O, we compute the value of the polynomial of Hermite interpolation $P_\theta(\rho) \in \mathbb{P}^{2n+1}$ that satisfies

$$\frac{\partial^j P_\theta}{\partial \rho^j}(R) = \frac{\partial^j f}{\partial \rho^j}(R,\theta),\ j = 0,\ldots,n, \tag{12}$$

$$\frac{\partial^j P_\theta}{\partial \rho^j}(R+d) = 0,\ j = 0,\ldots,n.$$

We then let $\bar{f} \equiv f$ in $B(0,R)$, $\bar{f} \equiv P_\theta$ in $B(0,(R,R+d))$, and $\bar{f} \equiv 0$ elsewhere.

We have implemented this method and forced the rhs of the Poisson equation to be such that the formula of some exact infinitely smooth solution is given in $B(O,R)$. In addition, since f is known *only* inside $B(O,R)$, we approximate the normal derivatives of f in (12) with one-side finite differences. Thanks to the coding of Boulin, we have tested the accuracy of our method on many different solutions. Table 1 that follows gives a representative example of the accuracy of our method. In this example, the exact solution is

$$u(x,y) = 20\ (x^6 + y^5) \exp(-(\log(6)\ (x^2 + y^2)),\ R = 1,$$

and the error is given in maximum norm. Nevertheless the space step in radial direction used to obtain numerical approximation of normal derivatives of $f = \Delta u$ at $\partial B(0,R)$ is 10^{-3}.

TABLE 1
Error with modulefou *for the Poisson problem on the disk.*

N	$n=0$	$n=2$	$n=4$	$n=6$
8	$2.5\ 10^{-2}$	$1.5\ 10^{-3}$	$8.3\ 10^{-4}$	$2.6\ 10^{-3}$
16	$2.7\ 10^{-3}$	$2.3\ 10^{-5}$	$8.6\ 10^{-6}$	$3.8\ 10^{-5}$
24	$3.6\ 10^{-4}$	$2.2\ 10^{-6}$	$8.5\ 10^{-7}$	$1.2\ 10^{-5}$
32	$2.7\ 10^{-4}$	$5.6\ 10^{-7}$	$4.\ 10^{-7}$	$4.4\ 10^{-6}$
64	$2.4\ 10^{-5}$	$2.7\ 10^{-8}$	$4.4\ 10^{-8}$	$1.\ 10^{-6}$

This first set of experiments demonstrates that the *modulefou* method is worthwhile at least with circular domains.

3. Theoretical basis of the method in one space dimension. In this section we are going to estimate the accuracy of the *modulefou* method on the two-point boundary value problem

$$-u'' + \lambda u = f \text{ in } \mathcal{I} = [0, L],\ u(0) = 0,\ u(L) = 0 \tag{13}$$

with λ a positive real number and f an arbitrary smooth function, i.e., f is not necessarily periodic on $\mathcal{I}$ and $f \in C^\infty(\mathcal{I})$. We introduce then the following decomposition

$$-v'' + \lambda v = \bar{f} \text{ in } \mathcal{J} = [0, L + d],\ v\ (L + d) \text{ periodic}, \tag{14}$$

and

$$-w'' + \lambda w = 0 \text{ in } \mathcal{I} = [0, L],\ w(0) = -v(0),\ w(L) = -v(L). \tag{15}$$

The extension $\bar{f}$ of f is built in such a way that

$$\bar{f}(x) = f(x),\ x \in \mathcal{I},$$

$$\bar{f}(x) = P(x),\ P \in \mathbb{P}^{2n+1},\ x \in \mathcal{J} \setminus \mathcal{I},$$

$$\bar{f}\ (L + d) \text{ periodic},\ \bar{f} \in C^n(\mathbb{R}),$$

where $\mathbb{P}^{2n+1}$ is the set of polynomial of degree $2n + 1$. Let v_N be the discrete Fourier approximation of solution v of (14). Let u_N be $v_N + w_N$, where w_N is the *exact* solution of

$$-w'' + \lambda w = 0 \text{ in } \mathcal{I} = [0, L],\ w(0) = -v_N(0),\ w(L) = -v_N(L). \tag{16}$$

We are going to estimate $||u - u_N||_{\infty,\mathcal{I}}$, the error in maximum norm on the interval $\mathcal{I}$. A classical Fourier estimate is [14]

$$||v - v_N||_{\infty,\mathcal{J}} \leq C^t \frac{|v|_q}{N^q} \left\{ \frac{\text{meas}(\mathcal{J})}{2\pi} \right\}^q,$$

with C^t a positive real number independent of N, $|v|_q = ||v^{(q)}||_{\infty,\mathcal{J}}$, $\text{meas}(\mathcal{J}) = L + d$, and $v^{(q)}$ the q^{ieme} derivatives of v.

It is easy to obtain from (14) the estimate

$$|v|_q \leq C^t ||\bar{f}||_{q-2},$$

where C^t is a positive number independent of q and $||\bar{f}||_i = \sum_{j=0,i} |\bar{f}|_j$. However, the key point is to obtain an estimate of the error $||v - v_N||_{\infty,\mathcal{J}}$ as a function of some norm of f on $\mathcal{I}$ only. We have

$$|\bar{f}|_{j,\mathcal{J}} \leq \ (|f|_{j,\mathcal{I}} + \ 2\ ||\bar{f}||_{H^{j+1}(\mathcal{J}\setminus\mathcal{I})}),$$

and consequently

$$||v - v_N||_{\infty,\mathcal{J}} \leq \frac{C^t}{N^q} \ \left\{\frac{\text{meas}(\mathcal{J})}{2\pi}\right\}^q \ (|f|_{q-2,\mathcal{I}} + ||\bar{f}||_{H^{q-1}(\mathcal{J}\setminus\mathcal{I})}).$$

Let $\{h_i^0,\ h_i^1,\ i = 0..n\}$ be the set of basis functions of $\mathbb{P}^{2n+1}$ used in Hermite interpolation such that

$$\bar{f}_{|\mathcal{J}\setminus\mathcal{I}} = \sum_{i=0,\ldots,n} f^{(i)}(L)h_i^0(x) + f^{(i)}(0)h_i^1(x).$$

We have for $q \leq n+2$,

$$||\bar{f}||_{H^{q-1}(\mathcal{J}\setminus\mathcal{I})} \leq \sum_{i=0,\ldots,n} (|f^{(i)}(L)| + |f^{(i)}(0)|)||h_i^0||_{H^{q-1}(\mathcal{J}\setminus\mathcal{I})})$$

and

$$||h_i^0||^2_{H^{q-1}(\mathcal{J}\setminus\mathcal{I})} = ||h_i^1||^2_{H^{q-1}(\mathcal{J}\setminus\mathcal{I})}) = \sum_{l=0,\ldots,q-1} d^{2i+1-2l}\ C^i_{l,n},$$

where

$$C^i_{l,n} = \int_0^1 \left(\frac{\partial^l g_i}{\partial \xi^l}(\xi)\right)^2 d\xi$$

and $g_i \in \mathbb{P}^{2n+1}(0,1)$ such that

$$\frac{\partial^l g_i}{\partial \xi^l}(0) = \delta^i_l,\ \frac{\partial^l g_i}{\partial \xi^l}(1) = 0,\ i,l = 0,\ldots,n.$$

Since the corrector w_N solution of (16) is given analytically as a function of the trace $v_{N,|\partial\mathcal{I}}$,

$$||u - u_N||_{\infty,\mathcal{I}} \leq \ 2\ ||v - v_N||_{\infty,\mathcal{J}}.$$

We have then finally for $q \leq n+2$,

(17)

$$||u - u_N||_{\infty,\mathcal{I}} \leq \frac{C^t}{N^q} \ \left\{\frac{L+d}{2\pi}\right\}^q \left(|f|_{q-2,\mathcal{I}} + \left(\sum_{i=0,\ldots,n} (|f^{(i)}(L)| + |f^{(i)}(0)|)^2 G_i^n(d)\right)^{\frac{1}{2}}\right),$$

with

$$G_i^n(d) = \left(\sum_{l=0,\ldots,q-1} d^{2i+1-2l}\ C^i_{l,n}\right).$$

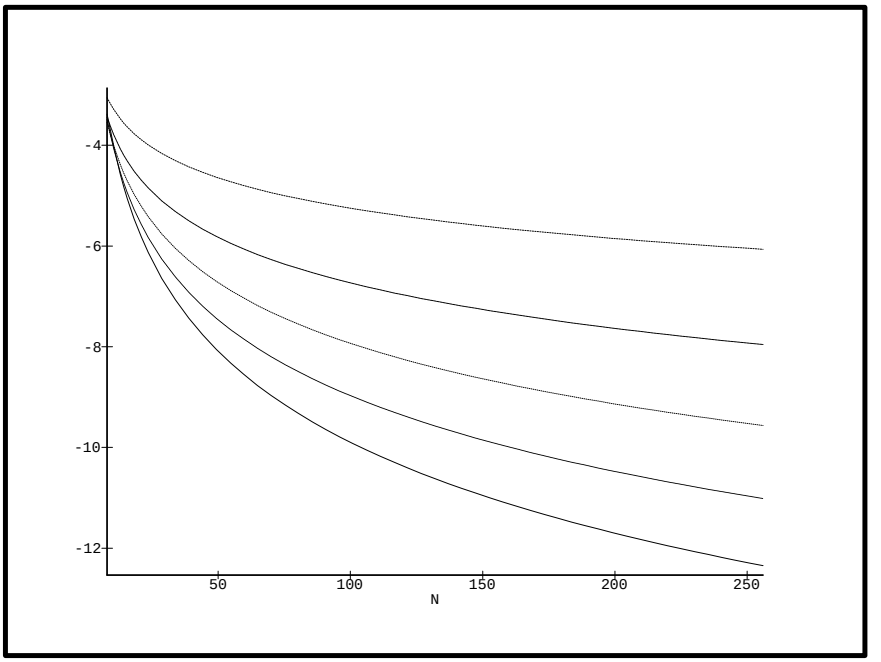


FIG. 1. *Estimate of the error in maximum norm with* $L = 1.6\,\pi,\ d = 0.25$.

The method *modulefou* is consequently of order $n+2$ if $\bar{f}$ is in $C^n(\mathbb{R})$. In addition a direct computation of

$$H(l,d) = \left\{\frac{L+d}{2\pi}\right\}^q \left(\sum_{i=0,\ldots,n} (G_i^n(d))\right)^{\frac{1}{2}}$$

with Maple shows that this quantity is relatively unsensitive to the measure of the extension d for large N; see Figures 1 and 2. We have implemented the method *modulefou* on problem (13) with $f(x) = \sin(x)$. In each run, $L+d$ and L are chosen so that f is not a smooth L-periodic function and $\bar{f}$ is not the sine function on the extension of the domain. The numerical error in maximum norm that we have obtained then is compatible with our estimate; see Figures 3 and 4.

4. Applications in one space dimension.

4.1. Singular perturbation problems. Let us consider the following elementary singular perturbation problems:

$$L_{2,0}[u] = -\epsilon u'' + \gamma(x)u = f(x),\ x \in (0,L), u(0) = \alpha,\ u(L) = \beta \tag{18}$$

and

$$L_{2,1}[u] = -\epsilon u'' + \gamma(x)u' = f(x),\ x \in (0,L), u(0) = \alpha,\ u(L) = \beta. \tag{19}$$

Both problems exhibit boundary layers and possibly turning points.

We recall that by using domain decomposition, we could divide the domain into subdomains so that there is no layer inside the subdomains. Consequently we restrict

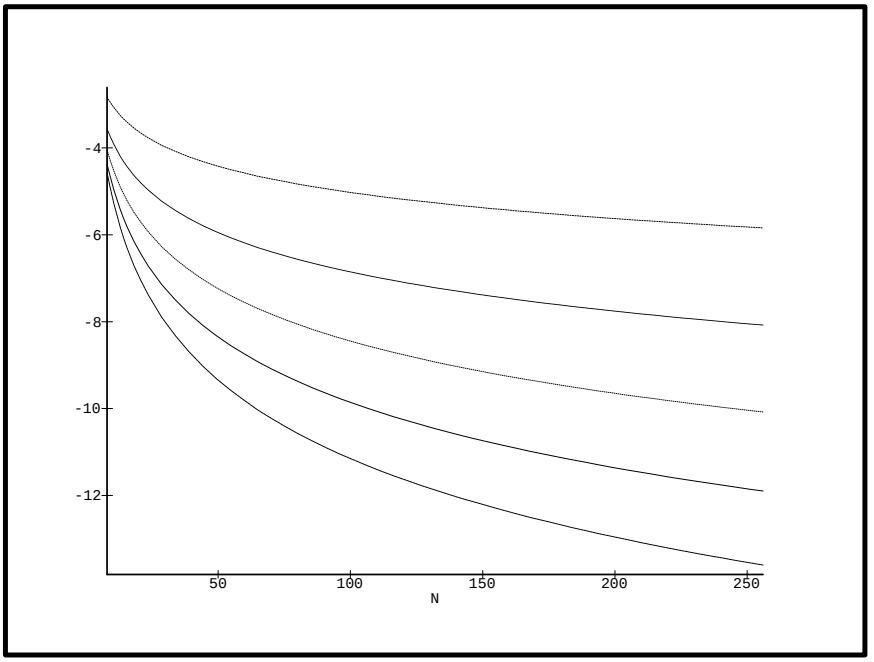


FIG. 2. *Estimate of the error in maximum norm with* $L = 1.6\ \pi,\ d = 1$.

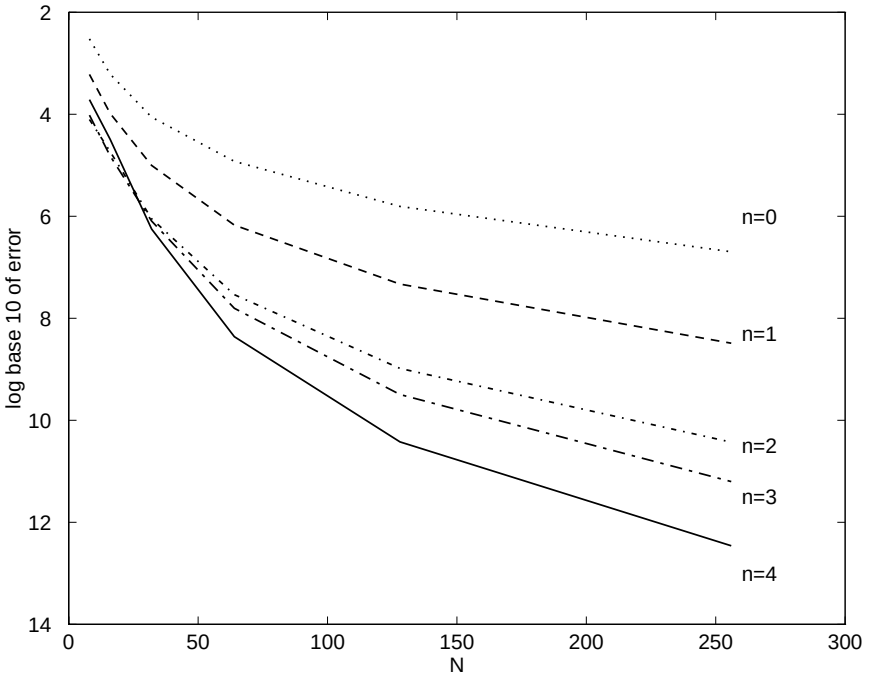


FIG. 3. *Numerical error in maximum norm with* $L = 1.6\ \pi,\ d = 0.25$.

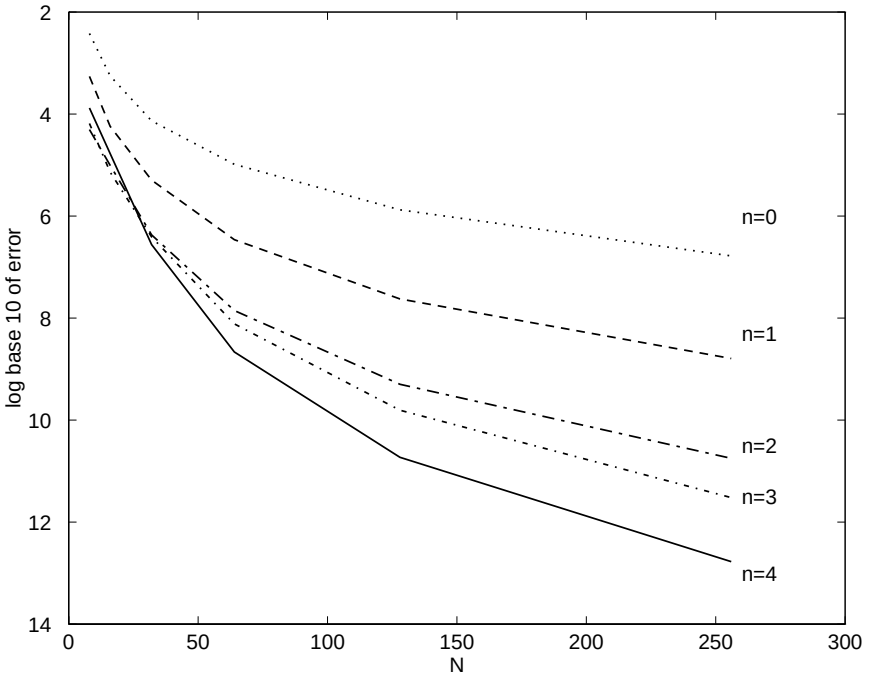


FIG. 4. *Numerical error in maximum norm with* $L = 1.6\,\pi,\ d = 1$.

ourselves to the case of boundary layers possibly at both ends of the interval. Let us consider the problems (18) or (19) with periodic boundary conditions

$$\bar{L}_{2,k}[v] \ = \ \bar{f}(x),\ x \in (0, L+d),\ v\ (L+d) \text{ periodic}, \tag{20}$$

with $\bar{f}$ (resp., $\bar{\gamma}$) a smooth $(L+d)$ periodic extension of f (resp., γ) in $C^n(\mathbb{R})$. We observe that the solutions of (20) have no boundary layer. Although the solution of (20) with $k = 1$ is defined up to a constant, the computation of the solution is easy by Fourier approximation. The solution of the corrector problem

$$L_{2,k}[w] \ = \ 0,\ x \in (0, L), w(0) = \alpha - v(0), w(L) = \beta - v(L) \tag{21}$$

belongs to a 2D vector space spanned by (w_1,w_2) the solutions of

$$L_{2,k}[w_1] \ = \ 0,\ x \in (0, L), w_1(0) = 1, w_1(L) = 0 \tag{22}$$

and

$$L_{2,k}[w_2] \ = \ 0,\ x \in (0, L), w_2(0) = 0, w_2(L) = 1. \tag{23}$$

The set of basis functions w_1,w_2 can be computed numerically with a specially-designed scheme to approximate boundary layers functions on refined grids or symbolically with, for example, Maple. We have implemented the *modulefou* method for the basic problems (18) and (19) with constant coefficients. We can estimate the error as in section 2: we obtain easily an estimate similar to (17) except that the rhs has a factor ϵ^{-1} because of the definition of the operator $L_{2,k}$. The direct numerical simulation confirms this result; see Figures 5 and 6 corresponding to the constant

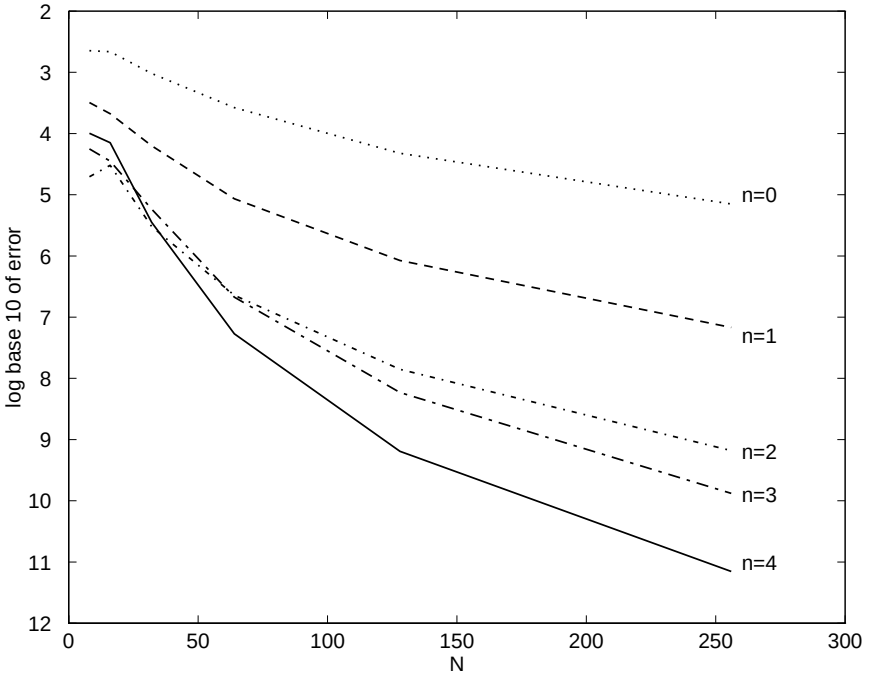


FIG. 5. *Error in maximum norm with* $k = 0,\ f \equiv \sin(x),\ L = 1.6\,\pi,\ d = 1,$ *and* $\epsilon = 0.1$.

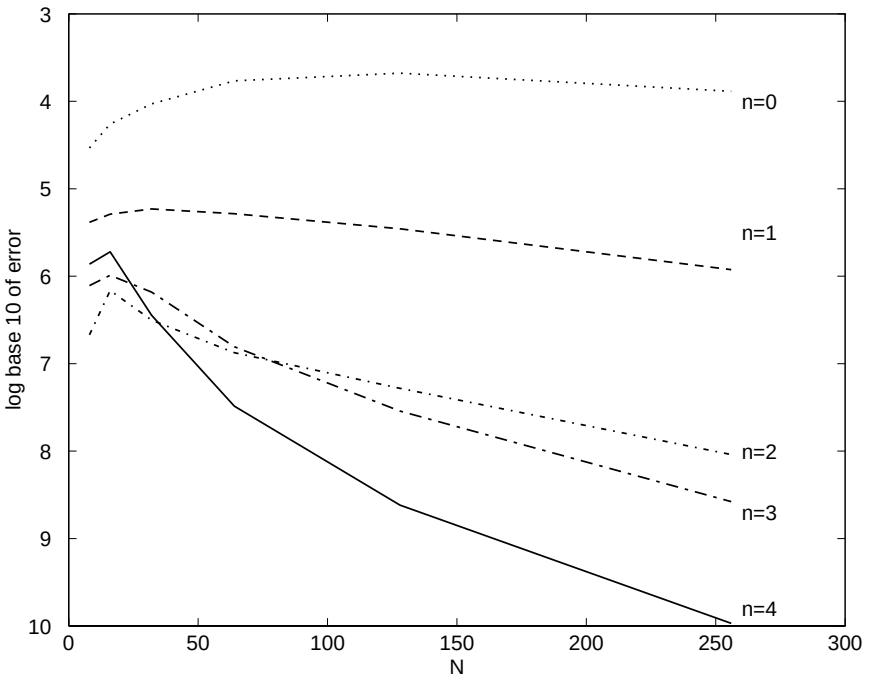


FIG. 6. *Error in maximum norm with* $k = 0,\ f \equiv \sin(x),\ L = 1.6\,\pi,\ d = 1,$ *and* $\epsilon = 0.01$.

coefficient $\gamma = 1$ case. In particular, we do not have any Gibbs phenomenon. One interesting feature of the numerical results of Figure 6 is that the accuracy is better than expected for small N. The reason for that is that the error is computed at the grid point and that there is no grid point in the layer for small N.

4.2. Heat equation. Let us consider the heat problem:

$$\frac{\partial u}{\partial t} - \frac{\partial^2 u}{\partial x^2} = f,\ x \in (0,1), t \in (0,T),$$

$$u(x,0) = u_0(x),\ x \in (0,1),$$

$$u(0,t) = g(t),\ u(1,t) = h(t),\ t \in (0,T).$$

We consider the following semidiscrete second-order (resp., second- and third-order) scheme in time:

$$\frac{u^n - u^{n-1}}{dt} = \frac{1}{2}(u^n_{xx} + f^n) + \frac{1}{2}(u^{n-1}_{xx} + f^{n-1}), \tag{24}$$

respectively,

$$\frac{\frac{3}{2}u^n - 2u^{n-1} + \frac{1}{2}u^{n-2}}{dt} - u^n_{xx} = f^n, \tag{25}$$

respectively,

$$\frac{\frac{11}{6}u^n - 3u^{n-1} + \frac{3}{2}u^{n-2} - \frac{1}{3}u^{n-3}}{dt} - u^n_{xx} = f^n. \tag{26}$$

u_{xx} denotes the second-order derivative with respect to the space variable x. Equation (24) is the Crank–Nicolson scheme, and (25), (26) are multilevel time-step schemes. These three schemes are unconditionally stable when u_{xx} is approximated by second-order centered finite difference formulas. For each time step t^n, we have to solve the two point boundary value problems:

$$-\frac{dt}{2}\, u^n_{xx} + u^n = rhs_1^{n-1}, u^n(0,t^n) = g(t^n), u^n(1,t^n) = h(t^n),$$

respectively,

$$-dt\, u^n_{xx} + \frac{3}{2}u^n = rhs_2^{n-1}, u^n(0,t^n) = g(t^n), u^n(1,t^n) = h(t^n),$$

respectively,

$$-dt\, u^n_{xx} + \frac{11}{6}u^n = rhs_3^{n-1}, u^n(0,t^n) = g(t^n), u^n(1,t^n) = h(t^n),$$

with

$$rhs_1^{n-1} = u^{n-1} + \frac{dt}{2}(u^{n-1}_{xx} + f^{n-1} + f^n),$$

respectively,

$$rhs_2^{n-1} = 2u^{n-1} - \frac{1}{2}u^{n-2} + dt\, f^{n-1},$$

respectively,

$$rhs_3^{n-1} = 3u^{n-1} - \frac{3}{2}u^{n-2} + \frac{1}{3}u^{n-3} + dt\ f^{n-1}.$$

The application of *modulefou* raises three questions: First, the accuracy of *modulefou* to solve the singular perturbation problem has been demonstrated in section 3.1, but the stability of the corresponding time-dependent scheme is not obvious. Second, rhs_1^n in the Crank–Nicolson scheme contains a second-order derivative in space that needs to be computed numerically. Third, how do we extend the rhs rhs^n defined on (0,1) into a smooth periodic function $r\bar{h}s^n$ of period $(1+d)$ on the interval $(0,\ 1+d)$, where d is the size of the extension, since rhs^n is known only *numerically* at the grid points $(x_i)_{i=1,\ldots,N}$ of (0,1)?

The second question is the easiest to answer. We have investigated two possibilities: let v^n (resp., w^n) be the periodic solution on the extended domain, (resp., the corrector) at time step t_n. We can use the splitting $u_{xx}^n = v_{xx}^n + w_{xx}^n$, where u_{xx}^n is computed with Fourier expansion and w_{xx}^n is given analytically as a function of w^n. Another solution is to derive u_{xx}^n from the equation of the scheme itself at previous time steps, i.e.,

$$u_{xx}^{n-1} = \frac{2}{dt}(u^{n-1} - rhs_1^{n-2})$$

in order to avoid the use of differentiation in the construction of the rhs.

The third question is solved in the following way.

Let $r\bar{h}s^n \equiv rhs^n$ *on* $(0,1)$; we interpolate numerically $r\bar{h}s^n$ on $(1, 1+d)$, so that

$$r\bar{h}s^n(1) = rhs^n(1),\ r\bar{h}s^n(1+d) = rhs^n(0), \tag{27}$$

and

$$\frac{\partial^j r\bar{h}s^n}{\partial x^j}(1) = \frac{\partial^j rhs^n}{\partial x^j}(1)\ +\ O(h^k),\ j = 1, \ldots, q, \tag{28}$$

$$\frac{\partial^j r\bar{h}s^n}{\partial x^j}(1+d) = \frac{\partial^j rhs^n}{\partial x^j}(0)\ +\ O(h^k),\ j = 1, \ldots, q. \tag{29}$$

We proceed by Hermite interpolation based on one side *k-order* finite difference approximations of the derivatives of rhs^n at the end of the interval of computation $(0,1)$ using only discrete values of rhs^n at the grid points inside $(0,1)$.

Finally we have investigated the question of stability of the resulting time-dependent scheme with several numerical experiments. More precisely, we have implemented these three schemes with the *modulefou* technique and compared the accuracy of the solution to finite difference implementations with centered second-order finite differences in space. In our comparisons, we use the same space step for all methods.

We define $f,\ u_o,\ g,\ h$ in the heat equation problem such that $u(x,t) = \sin(t)\ \cos(x)$ is the exact solution. The Hermite interpolation of $r\bar{h}s^n$ defined in (27)–(29) satisfies $k = q = 2$; we observe that $\bar{f}$ in the numerical scheme is not close to the periodic function f in $(1, 1+d)$.

Figure 7 gives the error in maximum norm for $t \leq \frac{\pi}{2}$. In each plot, the curves with a "+" (resp., "o") correspond to second-order finite differences in space (resp., *modulefou*). The flat part of curves corresponds to the interval of time step where the

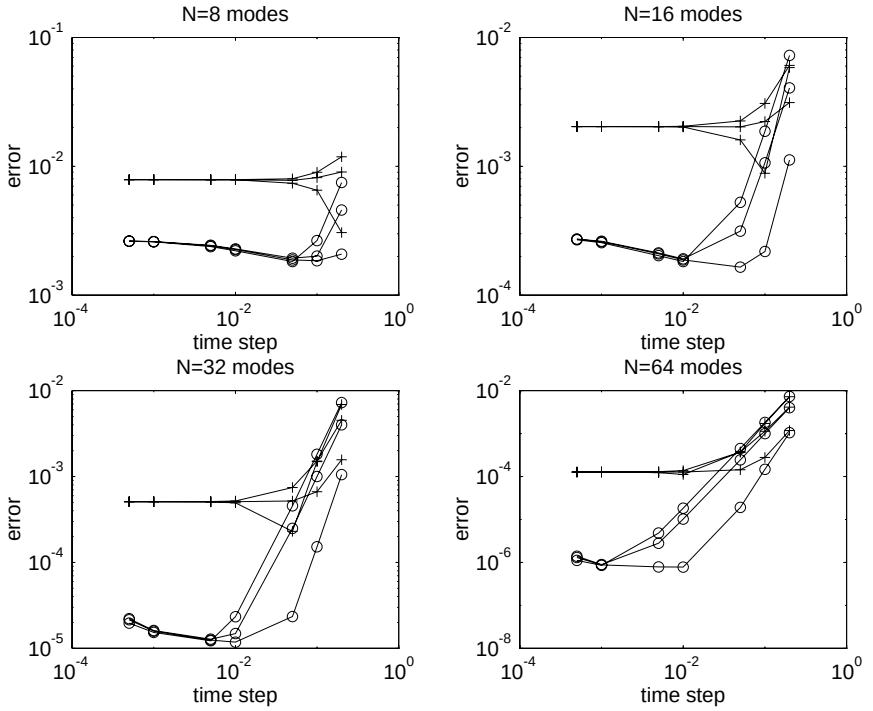


FIG. 7. *Numerical error in maximum norm for the heat equation problem.*

accuracy is limited by the spatial discretization. This numerical experiment shows a real advantage of *modulefou* compared to the classical second-order finite difference in space. In our experiments, the Crank–Nicolson scheme is slightly less accurate than the second-order scheme (25) for both types of space discretizations. The third-order scheme (26) gives effectively the best accuracy with respect to the time step. Although we had not observed any stability problem with *modulefou* even for very large time step, it is interesting to notice that the accuracy of the *modulefou* method for a fixed number of modes may deteriorate when the time step becomes too small. As a matter of fact, the operator $-dtu_{xx} + u$ does correspond to a singular perturbation problem similar to (18), and we have seen in section 3.1 that the accuracy in space with Fourier approximation deteriorates when the small perturbation parameter goes to zero.

4.3. Wave equation. We have seen that the *modulefou* method works quite well with the heat operator. However, the stability of the numerical method might be a consequence of the very nice smoothing properties of the heat operator. This should not be the case with the wave operator that we do consider now. Let us consider the wave problem

$$\frac{\partial^2 u}{\partial t^2} - c^2 \frac{\partial^2 u}{\partial x^2} = f,\ x \in (0,1), t \in (0,T),$$
$$u(x,0) = u_0(x),\ x \in (0,1),$$
$$\frac{\partial u}{\partial t}(x,0) = u_1(x),\ x \in (0,1),$$

$$u(0,t) = g(t),\ u(1,t) = h(t),\ t \in (0,T).$$

We consider the following semidiscrete second-order scheme in time:

$$\frac{u^{n+1} - 2u^n + u^{n-1}}{dt^2} = \frac{c^2}{2} u_{xx}^{n+1} + \frac{c^2}{2} u_{xx}^{n-1} + f^n. \tag{30}$$

u_{xx} is the notation for the second-order derivative with respect to the space variable x. The procedure to implement *modulefou* for this scheme is entirely analogous to the procedure used for the heat equation problem. We solve for each time step t^{n+1} the two point boundary value problems:

$$-\frac{dt^2 c^2}{2} u_{xx}^n + u^n = rhs_2^{n-1}, u^n(0,t^n) = g(t^n), u^n(1,t^n) = h(t^n),$$

with

$$rhs_2^n = c^2 \frac{dt^2}{2} u_{xx}^{n-1} + dt^2 f^n + 2u^n - u^{n-1}.$$

u_{xx}^n in the rhs can be either derived from the scheme at previous time step as follows

$$u_{xx}^{n-1} = \frac{2}{c^2 dt^2}(u^{n-1} - rhs_2^{n-2}),$$

or computed using the splitting $u^n = v^n + w^n$. We notice that (30) is unconditionally stable when we use a second-order finite difference in space. We have compared in our numerical experiments the *modulefou* technique with this finite difference version for several test cases. With a second-order Hermite interpolation, we found a better accuracy with *modulefou*. The following tables give the error in maximum norm when the exact solution of the wave problem is $u(x,t) = \sin(2\ \pi\ p\ (x - x_0))\ \cos(2\ \pi\ p\ t)$ with the wave number $p = 3$, the size of the extension $d = 0.2$, and the interval of time $(0, \pi)$. We restrict ourselves to use the same number $2\ M$ of discretization points for each method.

Error with second-order finite differences.

Time step	$M = 8$	$M = 16$	$M = 32$	$M = 64$
0.05	1.12	0.27	0.12	0.076
0.01	1.05	0.21	0.054	0.015
0.005	1.05	0.21	0.052	0.013
0.001	1.05	0.21	0.052	0.012
0.0005	1.05	0.21	0.052	0.012

Error with modulefou.

Time step	$M = 8$	$M = 16$	$M = 32$	$M = 64$
0.05	0.387	0.078	0.067	0.064
0.01	0.322	0.024	0.005	0.0028
0.005	0.320	0.023	0.003	0.0009
0.001	0.320	0.023	0.003	0.0003
0.0005	0.320	0.023	0.003	0.0003

In addition, we found no stability constraint on the time step even for large time steps. Figure 8 gives the typical evolution in time of the error for both methods. The thicker line corresponds to *modulefou*.

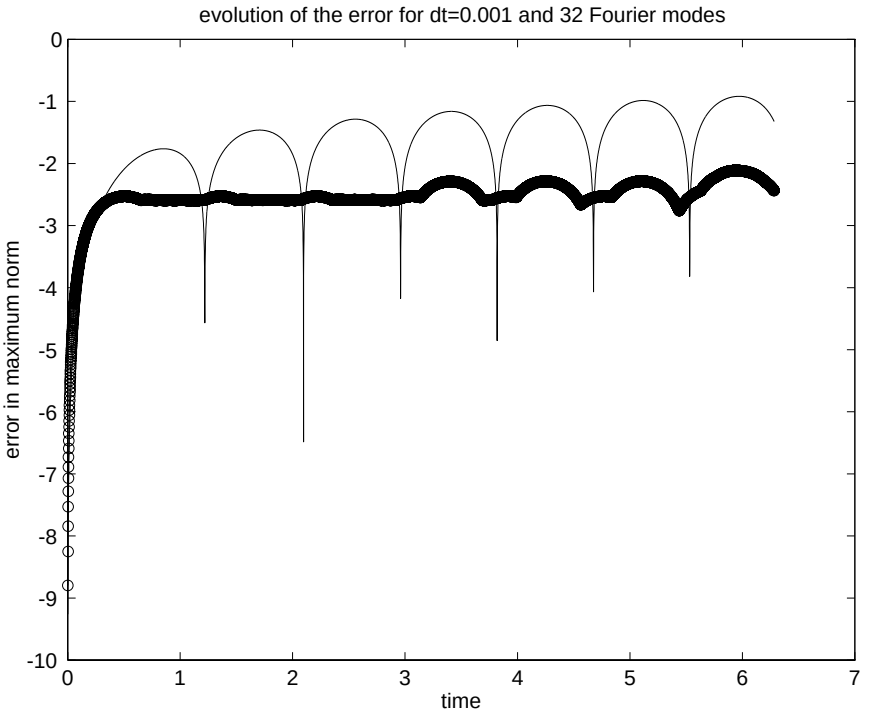


FIG. 8. *Numerical error in maximum norm for the wave equation problem.*

4.4. Hyperbolic-hyperbolic singular perturbation. Let us consider the following hyperbolic-hyperbolic singular perturbation problem:

$$\epsilon\left(\frac{\partial^2 u}{\partial t^2} - c^2\frac{\partial^2 u}{\partial x^2}\right) + \left(\frac{\partial u}{\partial t} + b\frac{\partial u}{\partial x}\right) = f,\ x \in (0,1), t \in (0,T),$$

$$u(x,0) = u_0(x),\ x \in (0,1),$$

$$\frac{\partial u}{\partial t}(x,0) = u_1(x),\ x \in (0,1),$$

$$u(0,t) = g(t),\ u(1,t) = h(t),\ t \in (0,T).$$

This model problem describes the propagation of waves with wave hierarchies [26] and occurs in many physical problems [24], [2]. We assume for simplicity that b and c are positive constants. We assume that the operator is time-like, i.e.,

$$-|c| < b < |c|.$$

The formal limit of this singular perturbation problem when $\epsilon \to 0$ is written

$$\frac{\partial u^0}{\partial t} + b\frac{\partial u^0}{\partial x} = f,\ x \in (0,1), t \in (0,T),$$

$$u^0(x,0) = u_0(x),\ x \in (0,1),$$

$$\frac{\partial u^0(x,0)}{\partial t} = u_1(x),\ x \in (0,1),$$

$$u^0(0,t) = g(t),\ u^0(1,t) = h(t),\ t \in (0,T).$$

Both initial conditions cannot be satisfied at the same time, and u exhibits an initial layer of ϵ thickness in time. Both boundary conditions cannot be satisfied as well, and u exhibits a boundary layer of ϵ thickness in space, at one end of the interval depending on the sign of the convection factor b. At first order, for b positive and for time of order one, u is approximated outside the initial layer in time and boundary layer in the neighborhood of 1 by

$$\frac{\partial u^0}{\partial t} + b\frac{\partial u^0}{\partial x} = f,\ x \in (0,1), t \in (0,T),$$

$$u^0(x,0) = u_0(x),\ x \in (0,1),$$

$$u^0(0,t) = g(t).$$

In addition, there are some other singular perturbation effects for large time, if the domain is large in space variable or unbounded: the second-order hyperbolic perturbation has then a diffusive effect. We refer to [4], [13] for the detailed analysis of the singular perturbation problem. We have tested the *modulefou* approach on this nontrivial singular perturbation problem with the following second-order scheme in time:

$$\epsilon\left[\frac{u^{n+1} - 2u^n + u^{n-1}}{dt^2} - \frac{c^2}{2}(u_{xx}^{n+1} + u_{xx}^{n-1})\right] + \frac{u^{n+1} - u^{n-1}}{2dt} + \frac{b}{2}(u_x^{n+1} + u_x^{n-1}) = 0. \tag{31}$$

A Fourier analysis shows that this time-dependent scheme combined with second-order finite differences in space for second-order spatial derivative and one side appropriate first-order derivative for the convective term is stable for our test case. The implementation of the *modulefou* technique is straightforward. In particular all derivatives that have to be computed in the rhs of (31) use the splitting $u^{n+1} = v^{n+1} + w^{n+1}$ at previous time steps. v^{n+1} is the solution of the periodic solution of the nonhomogeneous problem on an extension of the domain, w^{n+1} is the corrector.

We can make few observations:

- If one is not interested in having an accurate representation of the solution in the initial layer, it is convenient to start the computation with a scheme that is unconditionally stable. As a matter of fact, the effect of the initial layer decreases asymptotically as $\exp(-\frac{t}{\epsilon})$.
- The corrector w^{n+1} in the *modulefou* schemes does not capture the phenomenon of boundary layer because the rhs in the time-dependent scheme for the periodic problem is stiff. With no filtering, we need experimentally at least two grid points in the layer to avoid spurious oscillations. Otherwise, the scheme becomes numerically unstable.

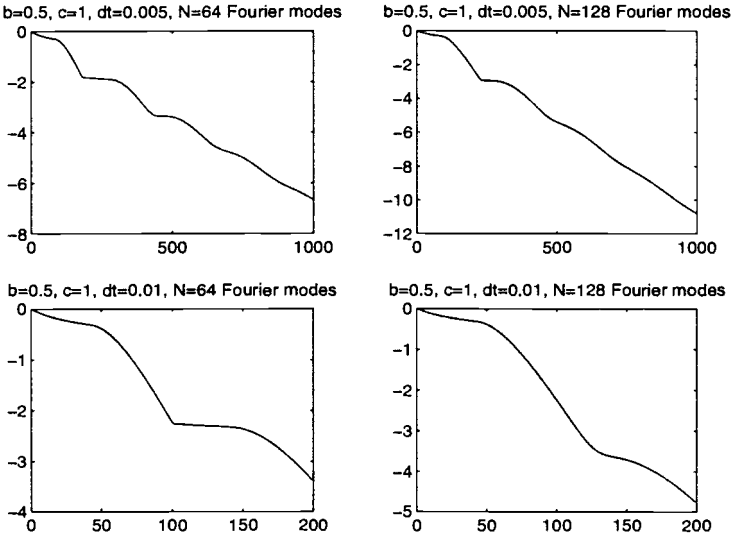


FIG. 9. *Variation in time of the amplitude of the wave for* $\epsilon = 0.05$.

Let us consider the following test case:

$$f \equiv 0,\ u_0(x) = \exp\left(-5\left(x - \frac{1}{2}\right)^2\right),\ u_1 \equiv 0,\ g \equiv 0,\ h \equiv 0,\ b = 0.5,\ c = 1,\ \epsilon = 0.05.$$

Our numerical experiment shows that the *modulefou* method can give inaccurate solutions when the space step in the layer is not small enough compared to the thickness of the layer. Figure 9 shows the evolution of the numerical value of $\log_{10}(\max_{x\in(0,1)} u(x,t))$ as a function of the space step for various discretization parameters in space and time. We observe an artificial wave that goes back and forth in time into the interval $(0,1)$: this signal is damped every time it hits the right boundary $x = 1$. The height of the second step strongly depends on the number of modes as well as the time step; it is also a lower bound on the numerical accuracy of the method. Our validation of this result has included comparison with second-order finite differences solutions as well as piecewise Chebyshev polynomials for which this phenomenon does not occur.

4.5. Burgers's equation and other nonlinear examples. Let us consider the initial boundary value problem with the Burgers equation

$$\frac{\partial u}{\partial t} = \epsilon \frac{\partial^2 u}{\partial x^2} + u\frac{\partial u}{\partial x}, \qquad x \in (-1,1), t \in (0,T), \tag{32}$$

$$u(x,0) = u_0(x),\ x \in (0,1); u(-1,t) = g(t),\ u(1,t) = h(t),\ t \in (0,T). \tag{33}$$

This simple model problem has a number of interesting properties that can make the numerical computation rather challenging [19], [20]. Let us illustrate these difficulties with some examples.

We suppose that we have balanced boundary conditions with $g(t) \equiv 1$ and $h(t) \equiv -1$; we assume also that u_0 is linear and continuous on $(-1,1)$ and satisfies the symetry $u_0(-x) = u_0(x)$. Then the solution $u(x,t)$ has the symmetry property $u(x,t) = -u(-x,t)$ for all time. After some evolution of the solution during a period of time

of order one, $u(x,t)$ will be exponentially close to the steady state solution [19]:

$$u(x,\infty) = \delta(\epsilon)\tanh\left(\frac{\delta(\epsilon)}{2\epsilon}x\right),$$

with δ solution of the transcendental equation

$$\delta = 1 + (\delta+1)\exp\left(\frac{-\delta}{\epsilon}\right).$$

Asymptotically we have

$$\delta(\epsilon) = 1 + 2\exp\left(-\frac{1}{\epsilon}\right) - \frac{2}{\epsilon}(2-\epsilon)\exp\left(-\frac{2}{\epsilon}\right) + O\left(\exp\left(-\frac{3}{\epsilon}\right)\right).$$

$u(x,t)$ is then stiff with a so-called shock layer of order ϵ thickness. This shock layer is formed at some time t_0 depending on the initial condition [21]. The fast variation of the solution in the shock layer is the first type of difficulty that the numerical method should overcome.

A second difficulty that should not be ignored in numerical tests with problem (32) is that the solution is supersensitive to boundary conditions.

As a matter of fact, let us change slightly the boundary conditions with, for example,

$$g(t) \equiv 1 + 2\,\exp\left(\frac{-b}{\epsilon}\right).$$

Then the steady state changes dramatically, more precisely [19],

$$u(x,\infty) \sim \tanh\left(\frac{x-1+b}{2\epsilon}\right).$$

This phenomenon is not limited to the steady solution; let us consider, for example, the balanced boundary conditions case defined above but with a new initial condition that is

$$u_0(x) = \tanh\left(\frac{x-x_0}{2\epsilon}\right).$$

Then $u(x,t)$ will stay asymptotically close to the travelling wave $u_0(x^*(t))$ with $\frac{d}{dt}(x^*)$ exponentially small. In particular, the zero of $u(x,t)$ will move exponentially slowly in time.

We have implemented the *modulefou* method to compute the solution of the initial boundary value problem (32) written in conservative form in a straightforward way with the Euler semi-implicit scheme

$$\frac{u^n - u^{n-1}}{dt} = \epsilon\frac{\partial^2 u^n}{\partial x^2} + \frac{1}{2}\frac{\partial (u^{n-1})^2}{\partial x}.$$

The problem is splitted at each iteration into a nonhomogeneous problem with periodic boundary conditions

$$-\epsilon\, dt\frac{\partial^2 v^n}{\partial x^2} + v^n = r\bar{h}s^{n-1},\ x \in (0,2\pi),$$

with

$$rhs^{n-1} = u^{n-1} + \frac{dt}{2}\,\frac{\partial (u^{n-1})^2}{\partial x}$$

and a homogeneous problem for the corrector with Dirichlet boundary conditions

$$-\epsilon\, dt \frac{\partial^2 w^n}{\partial x^2} + w^n = 0,\ w(-1) = g(t^n) - v^n(-1), w(1) = h(t^n) - v^n(1).$$

We first observe that rhs^n is stiff as soon as a shock is formed. So the solution v^n of the periodic problem is somewhat stiff as well. The results of section 3.1 in terms of accuracy therefore do not apply. We observe in addition that the layer for the corrector problem has a thickness of order $\sqrt{\epsilon dt}$, but the thickness of the shock layer is of order ϵ. In conclusion we cannot expect any smoothing process from *modulefou* since we do not approach the real corrector of Burgers's equation with this straightforward implementation.

In addition, in order to approximate $\frac{\partial (u^{n-1})^2}{\partial x}$, we first extend the flux into a periodic function and then apply Fourier differentiation. This process somehow deteriorates the accuracy of the scheme. An alternative approach is to implement a nonconservative scheme with *modulefou* and decompose $\frac{\partial u^{n-1}}{\partial x}$ into $\frac{\partial v^{n-1}}{\partial x} + \frac{\partial w^{n-1}}{\partial x}$, where the first term is computed via Fourier expansion and the second term is derived analytically from the representation of w^{n-1}. However, our experiment with a nonconservative scheme did not give good numerical accuracy.

We have summarized in the following tables our comparison of our implementation of *modulefou* with a classic second-order finite difference scheme. Since we want to investigate the spatial accuracy of our discretization and its stability, we have computed the steady solution of the symetric solution of Burgers's equation in the case of balanced boundary conditions. We found that the computation with the *modulefou* method is then accurate and does not show serious spurious oscillations when the space step is less than ϵ. On the contrary, if ϵ is too small compared to the space step, the *modulefou* scheme becomes unstable because of the Gibbs phenomenon. Because the theory of filtering with Fourier expansion is well known, we can work eventually with larger space step using some filters to remove the spurious oscillations.

Error with modulefou.

ϵ	$M = 8$	$M = 16$	$M = 32$	$M = 64$
0.1	10^{-3}	$5.\ 10^{-4}$	$6.\ 10^{-6}$	$6.\ 10^{-7}$
0.05	*diverge*	$8.\ 10^{-4}$	10^{-5}	$9.\ 10^{-7}$
0.01	*diverge*	*diverge*	*diverge*	$3.\ 10^{-3}$

Error with second-order finite differences.

ϵ	$M = 8$	$M = 16$	$M = 32$	$M = 64$
0.1	$1.8\ 10^{-2}$	$5.\ 10^{-3}$	$1.2\ 10^{-3}$	$3.\ 10^{-4}$
0.05	$8.\ 10^{-2}$	$1.7\ 10^{-2}$	$5.\ 10^{-3}$	10^{-3}
0.01	$1.5\ 10^{-1}$	$1.8\ 10^{-1}$	10^{-1}	$2.8\ 10^{-2}$

We have also investigated the accuracy of our scheme with respect to the boundary conditions. This computation is far more difficult and requires exponentially large T time to reach the steady solution as demonstrated in the following table. Therefore, a straightforward integration in time of the Burgers equation is by no means an

efficient way of solving the problem [23]. However, it is a good test on the stability of a numerical computation. We have compared *modulefou* to second-order finite differences computation with 32 discretization points, as well as adaptive pseudospectral Chebyshev with two subdomains and 39 Chebyshef points per subdomain. Our criterion is to compare the zero of our solution u with its asymptotic prediction. Since the computation was done with only 16 Fourier modes, *modulefou* seems to perform very well.

Test with supersensitivity.

δ	Modulefou	Finite differences	Chebysheff	Asymptotic	T
500 $\exp(\frac{-1}{\epsilon})$	0.5608	0.5668	0.5610	0.5521	300
100 $\exp(\frac{-1}{\epsilon})$	0.3933	0.4046	0.3937	0.3912	600
20 $\exp(\frac{-1}{\epsilon})$	0.2315	0.2483	0.23184	0.23025	1800

In order to improve the accuracy of the computation with respect to the stiffness of the solution, we have introduced an adaptive domain decomposition of (32) [6]. Let us consider the case of balanced boundary conditions with a continuous linear initial condition. We use three subdomains as in [7, p. 1191] except that there is no overlapping between the subdomains. In particular, there is a large ratio between the space step in the subdomain centered on the shock layer and the two other subdomains (see Figure 10) in such a way that the space step in the inner domain is of the order of ϵ. The interesting and somewhat surprising conclusion of our numerical experiment is that the adaptive domain decomposition allows us to effectively avoid oscillations in the zone where the solution is stiff, although very small spurious oscillations in the two subdomains where the solution is regular induce a shifting of the solution in the inner domains. This is a direct consequence of the supersensitivity of the inner solution on the interface conditions with the left and right regular domains. This unfortunate circumstance makes the solution inaccurate after all! Figure 10 gives some picture of the three subdomain computation. The shift of the solution in the shock layer compared to the exact steady state ("o" plots) is increasing very slowly in time as expected from the asymptotic theory.

Now we consider the simplified model of reacting flow of Majda [22], that is,

$$\frac{\partial u}{\partial t} + \frac{\partial}{\partial x}\left[\frac{1}{2}u^2 - Q_0 Z\right] = \epsilon \frac{\partial^2 u}{\partial x^2},$$

$$Z_x = \epsilon^{-1}\phi(u)Z,$$

where $\phi(u) = 1$ if $u > 0$ and 0 elsewhere. We refer to [22] and its references for the derivation of this model and its precise statement. This problem admits various travelling wave solutions with sharp transition layers of ϵ thickness. These travelling waves are written $u(\frac{x-s\,t}{\epsilon})$ with the asymptotic behavior $u_x \to u_{l/r}$ when $x \to \mp\infty$. We report here on the computation of this model problem with *modulefou* and a first-order semi-implicit Euler time stepping. We consider two cases: first a strong detonation wave that corresponds to $u_l = 1, u_r = -1.5, s = 0.7, Q_0 = 2.375$, second a weak detonation wave that corresponds to $u_l = 1, u_r = -0.407, s = 0.7, Q_0 = 0.568$. Strong detonations are very difficult to compute because the viscous terms and the reaction terms are of the same asymptotic order in the sharp reacting layer. Figure 11 shows the profile of the solutions for $\epsilon = 0.03$ after 2 units of time. The initial condition is $\frac{u_l - u_r}{2}(1 - \tanh(\frac{x-1}{0.1})) + u_r$. With a space step $h = 0.0236$ a time step as

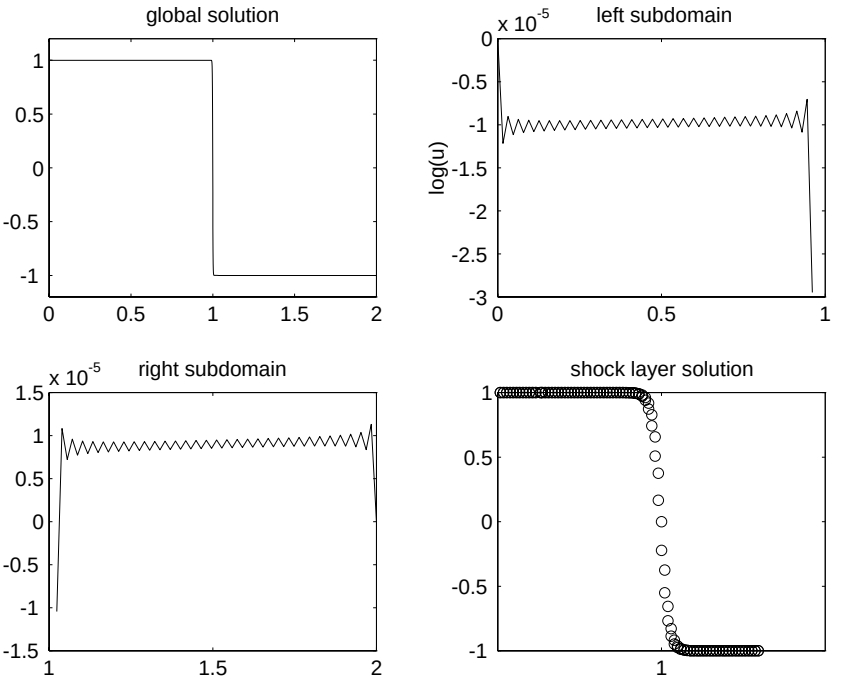


FIG. 10. *Adaptive domain decomposition for Burgers's problem.*

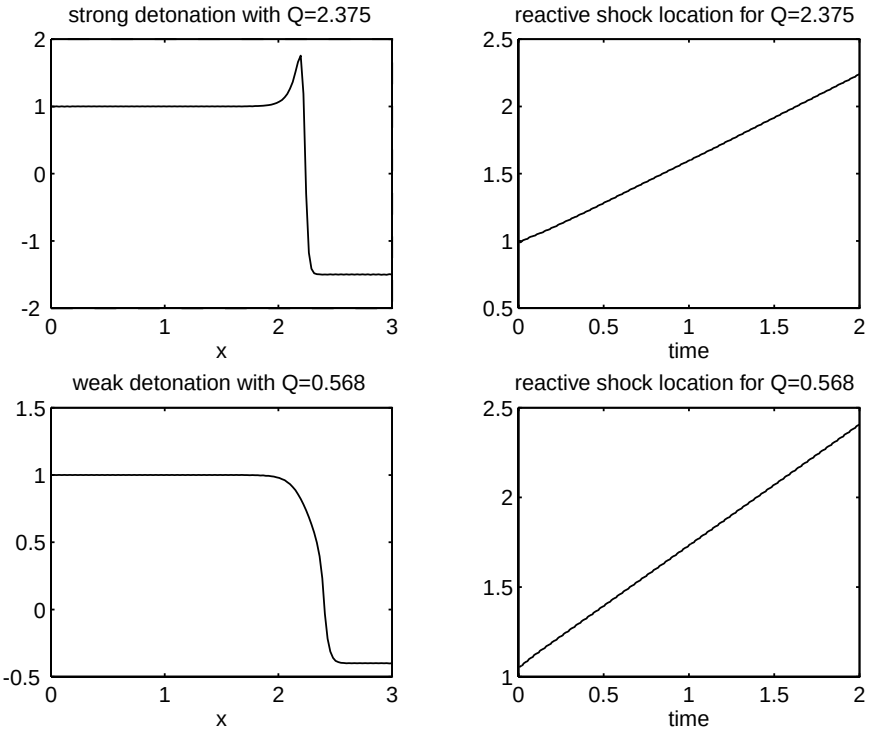


FIG. 11. *Majda's simplified model of reacting flow.*

small as 2.5 10^{-3} and 82 Fourier modes, the error on the speed velocity is about 10 percent in the first case and 5 percent in the second case. This accuracy is satisfactory considering the fact that we do not use any adaptivity in space. As a matter of fact, when one doubles the viscosity coefficient in the previous numerical experiment, i.e., $\epsilon = 0.06$, the observed numerical error on the front velocity is divided by two in each previous test case.

5. Extension to two-space dimension problems. We consider the singular perturbation problem

$$-\epsilon\Delta u + u = f \text{ in } \Omega \subset (0,\ 2\pi)^2,\ u = g \text{ on } \partial\Omega, \tag{34}$$

where Δ is the Laplace operator, Ω is a regular domain of $\mathbb{R}^2$, $f \in C^\infty(\Omega)$, and $g \in C^\infty(\partial\Omega)$. The solution of this singular perturbation problem has a boundary layer of $\sqrt{\epsilon}$ thickness. Let $\bar{f}$ be a smooth 2π periodic extension of f, with $\bar{f} \in C^n(\mathbb{R}^2)$. Then the 2π periodic solution of the following problem:

$$-\epsilon\Delta v + v = \bar{f} \text{ in } (0,\ 2\pi)^2 \tag{35}$$

has no boundary layer. On the other hand, the solution of the corrector problem

$$-\epsilon\Delta w + w = 0 \text{ in } \Omega \subset (0,\ 2\pi)^2,\ w = g - v \text{ on } \partial\Omega$$

can either be approximated by an asymptotic expansion [3] or computed numerically on an adapted stretched local coordinate system. We retrieve the solution of the original singular perturbation problem (34) using the superposition principle. We are first going to show using the arguments (2) and (3) of the introduction that *modulefou* is an attractive way to solve the singular perturbation problem (34). However, our real goal is to design solvers of (34) that are useful to compute the time-dependent problem. So second we are going to see how in practice one can implement the *modulefou* method on domain with smooth but relatively complex shape to serve this purpose.

5.1. Combined numerical-symbolical solution. We first construct numerically $\bar{f}$ an appropriate periodic extension of f, in a similar manner to section 2. Let (ρ, θ) be a local coordinate system valid in a ring Ω_e defined below, with θ a parametrization of $\partial\Omega$ and ρ the distance to $\partial\Omega$. We assume that $f_{|\partial\Omega}$ and its normal derivatives $\frac{\partial^i f}{\partial\rho^i}_{|\partial\Omega}$ up to order i are given on $\partial\Omega$. We assume that it exists $\rho_0 > 0$ such that

$$\Omega_e = \{M(x,y) \in \mathbb{R}^2 \setminus \Omega,\ \text{dist}(M, \partial\Omega) \le \rho_0\} \subset (0,\ 2\pi)^2$$

and the local coordinates system (ρ, θ) are well defined in Ω_e. Then we construct $\bar{f}$ as follows:

$$\bar{f}(x,y) = P_\theta(\rho),\ P_\theta \in \mathbb{P}^{2n+1},\ M(x,y) \in \Omega_e,$$

$$\frac{\partial^i P}{\partial\rho^i}(0) = \frac{\partial^i f}{\partial\rho^i} \text{ on } \partial\Omega,\ i = 0, \ldots, n,$$

$$\frac{\partial^i P}{\partial\rho^i}(\rho_0) = 0,\ i = 0, \ldots, n,$$

$$\bar{f} \equiv 0 \text{ in } (0,\ 2\pi)^2 \setminus \left(\Omega \bigcup \Omega_e\right),$$

$$\bar{f} \equiv f \text{ in } \Omega.$$

Second we approximate v solution of (35) by a discrete Fourier expansion v_N as in section 2 except that the solution is unique. The last step is the computation of w the solution of the corrector problem

$$-\epsilon \Delta w + u = f \text{ in } \Omega \subset (0,\ 2\pi)^2,\ w = g - v \text{ on } \partial\Omega, \tag{36}$$

based on an asymptotic expansion of w and the Fourier approximation $g - v_N$ of the boundary condition. Let us review briefly the construction of the asymptotic expansion of the corrector. The ρ coordinate is stretched according to the thickness of the boundary layer, i.e., $\xi = \frac{\rho}{\sqrt{\epsilon}}$. In these coordinates, the Laplace operator is written

$$\epsilon\Delta = \frac{\partial^2}{\partial\xi^2} \ +\ \epsilon\ q_0^1\ \frac{\partial^2}{\partial\theta^2} \ +\ \sqrt{\epsilon}\ q_0^{-1}\ q_1\ \frac{\partial}{\partial\xi} \ -\ \epsilon\ \ q_0^{-2}\ q_2\ \frac{\partial}{\partial\theta},$$

with $q_0 = 1 + 2\sqrt{\epsilon}\xi\phi + \epsilon\xi^2\psi$, $q_1 = \sqrt{\epsilon}\xi\psi + \phi$, $q_2 = \sqrt{\epsilon}\xi\phi' + \frac{1}{2}\epsilon\xi^2\psi'$, $\phi = \langle n', \hat{x}'\rangle$, $\psi = \langle n', n'\rangle$. " $'$ " denotes the differentiation with respect to θ, n the outward normal direction to $\partial\Omega$, and $\langle .,. \rangle$ the scalar product.

It is known [3] that w solution of (36) admits an asymptotic expansion of the following type:

$$w(\xi,\theta) = w_0(\xi,\theta) + \sqrt{\epsilon}w_1(\xi,\theta) + \epsilon w_2(\xi,\theta) + \cdots + \epsilon^{\frac{m}{2}} w_m(\xi,\theta) + O(\epsilon^{\frac{m+1}{2}}),$$

with arbitrary large order m. The validity of this expansion in maximum norm is proved using the maximum principle. The effective construction of this expansion can be done with a symbolic manipulation language as Maple. $w_i,\ i = 0, \ldots, 2$, for example, are solutions of the following ODE problems parametrized by θ:

$$-\frac{\partial^2 w_0}{\partial\xi^2}(\xi,\theta) + w_0(\xi,\theta) = 0,$$
$$w_0(0,\theta) = h(\theta),\ w_0(+\infty,\theta) = 0,$$

with $h(\theta) = g - v,\ on\ \partial\Omega$,

$$-\frac{\partial^2 w_1}{\partial\xi^2}(\xi,\theta) + w_1(\xi,\theta) = \phi(\theta)\frac{\partial w_0}{\partial\xi}(\xi,\theta),$$
$$w_1(0,\theta) = 0, w_1(+\infty,\theta) = 0,$$

$$-\frac{\partial^2 w_2}{\partial\xi^2}(\xi,\theta) + w_2(\xi,\theta)$$
$$= \phi(\theta)\ \frac{\partial w_1}{\partial\xi}(\xi,\theta) \ +\ \frac{\partial^2 w_0}{\partial\theta^2}(\xi,\theta) \ +\ (2\xi\phi^2(\theta)\xi\psi(\theta))\ \frac{\partial w_0}{\partial\xi}(\xi,\theta),$$
$$w_2(0,\theta) = 0,\ w_2(+\infty,\theta) = 0.$$

We obtain the explicit formula of the corrector at order $\epsilon^{\frac{3}{2}}$,

$$w_0(\xi,\theta) = h(\theta)e^{-\xi},$$

$$w_1(\xi,\theta) = h_1^1(\theta)\xi)e^{-\xi},$$

$$w_2(\xi,\theta) = (h_2^1(\theta)\xi + h_2^2(\theta)\xi^2)e^{-\xi},$$

with

$$h_1^1(\theta) = -\frac{1}{2}\phi(\theta)h(\theta),\ h_2^1(\theta) = \frac{1}{8}\psi(\theta)h(\theta) + \frac{1}{2}h''(\theta),\ h_2^2(\theta) = \frac{3}{8}\psi(\theta)h(\theta).$$

It can be proved by induction that

$$w_i = \sum_{j=1,\ldots,m} h_i^j(\theta)\xi^j e^{-\xi},$$

and that h_i^j depends on derivatives of $h(\theta)$ up to order i at most.

We can generate with Maple, the fortran code to compute the explicit formula of the asymptotic expansion of the corrector at any fixed order m. However, these formula require the computation of derivatives of $h(\theta)$ up to order m. In order to approximate these derivatives, we introduce the discretization in θ variable: $\theta_k = \frac{2\,k\,\pi}{2M+1}, k = 1,\ldots,2\,M$. Let v_N be the Fourier expansion of v solution of (35). We interpolate v_N on $\partial\Omega$ for $\theta = \theta_k,\ k = 1,\ldots,2\,M$, and derive a discrete Fourier expansion h_M of $h(\theta)$. The derivatives of $h(\theta)$ are then approximated with the corresponding derivatives of h_M. Since the i^{eme} term of the asymptotic expansion of the corrector depends on the i^{eme} derivative at most of h_M, the relative accuracy on w_i compared to the accuracy on w_0 is of order M^{-i}. The contribution of this term to the error in the asymptotic corrector formula is then of order $\epsilon^{\frac{i}{2}}M^{-i}$ times the error on w_0.

Therefore, the numerical approximation of the asymptotic expansion of the corrector $\sum_{i=0,\ldots,m}\epsilon^{\frac{i}{2}}w_i$ will not deteriorate (in theory) when m increases as long as $\sqrt{\epsilon}$ is of order M^{-1} at most.

This symbolic computation of the corrector w combined to the Fourier approximation of v seems to be quite natural to specifically solve the singular perturbation problem (34). However, we have to recall that the asymptotic expansion of w *does not converge* as the number of terms in the asymptotic expansion grows. For every ϵ and boundary condition h, there is an optimum value of the number of term of the asymptotic expansion that gives the best accurate solution. This does not sound like a practical solution to implement the *modulefou* method for the time-dependent problem.

We have checked for example that the solver obtained with the first-order asymptotic correction w_0 applied to the heat equation is numerically stable in our experiments. However, for the time being, we have not been able to obtain any convincing numerical convergence of this scheme to some exact specific steady solution of the heat equation problem when the number of Fourier modes grows and the time step goes to zero. Higher-order asymptotic corrections did not improve the situation significantly either.

5.2. A numerical solver for the time-dependent problem. The goal of this section is to generalize our approach for the time-dependent problem in one space dimension to two space dimensions. We are now going to compute the corrector of problem (34) *numerically* instead of asymptotically for the reasons mentioned above. So we focus our attention to the numerical solution of the following Dirichlet homogeneous problem:

$$-\epsilon\,\Delta w\ +\ w\ =\ 0 \text{ in } \Omega,\ w = h \text{ on } \partial\Omega. \tag{37}$$

For simplicity, we assume that the boundary $\partial\Omega$ of Ω can be defined in polar coordinates as follows:

$$\rho = S(\theta),\ \theta \in (0, 2\,\pi),$$

with S 2π periodic and $S(\theta) \in C^\infty[0, 2\,\pi]$. It is clear that when Ω is a disc and h is a finite discrete Fourier expansion, we just get analytically w using some Bessel functions. We could then reproduce the algorithm of section 2. In this paper, we would like to tackle the case of complex geometry. We introduce the change of variable $\hat{\rho} = S(\theta) - \rho,\ \hat{\theta} = \theta$ and rewrite (37) in this new coordinate system:

$$-\epsilon\left(\alpha(\hat{\rho},\hat{\theta})\,\frac{\partial^2\hat{w}}{\partial\hat{\rho}^2} + \beta(\hat{\rho},\hat{\theta})\,\frac{\partial\hat{w}}{\partial\hat{\rho}} + \gamma(\hat{\rho},\hat{\theta})\,\frac{\partial^2\hat{w}}{\partial\hat{\rho}}\partial\hat{\theta} + \delta(\hat{\rho},\hat{\theta})\,\frac{\partial^2\hat{w}}{\partial\hat{\theta}^2}\right) + w = 0 \tag{38}$$

$$\text{for } \hat{\rho} \in (0, S(\hat{\theta})),\ \hat{\theta} \in [0, 2\,\pi],$$

$$\hat{w}(0,\hat{\theta}) = h(\hat{\theta}),\ \hat{\theta} \in [0, 2\pi], \tag{39}$$

with

$$\alpha(\hat{\rho},\hat{\theta}) = 1 + \frac{S'^2(\hat{\theta})}{(S(\hat{\theta}) - \hat{\rho})^2},$$

$$\beta(\hat{\rho},\hat{\theta}) = \frac{S''(\hat{\theta})}{(S(\hat{\theta}) - \hat{\rho})^2} - \frac{1}{S(\hat{\theta}) - \hat{\rho}},$$

$$\gamma(\hat{\rho},\hat{\theta}) = 2\frac{S'(\hat{\theta})}{(S(\hat{\theta}) - \hat{\rho})^2},$$

$$\delta(\hat{\rho},\hat{\theta}) = \frac{1}{(S(\hat{\theta}) - \hat{\rho})^2}.$$

Because of the required asymptotic behavior of the corrector, we can close the problem with the following approximate boundary condition:

$$\hat{w}(L_\infty,\hat{\theta}) = 0,\ \hat{\theta} \in [0, 2\pi], \tag{40}$$

where L_∞ is a positive constant such that the local coordinate system $(\hat{\rho},\hat{\theta})$ is well defined in Ω. In theory, for ϵ small enough, we make an exponentially small error because we know that the true corrector w is exponentially small outside a boundary layer of $\sqrt{\epsilon}$ thickness. But since ϵ is a given number in our numerical computations, we have to choose carefully L_∞ large enough in such a way that the computation of the corrector becomes independent of L for $L > L_\infty$ within truncation error.

A more consistent approach is to use the stretched local coordinate system described in section 5.1. But the strip for which this local coordinate system is properly defined can be so narrow that a boundary condition similar to (40) does not work.

We have chosen therefore to approximate numerically the corrector on a structured regular grid in $\hat{\rho} = S(\theta) - \rho,\ \hat{\theta} = \theta$ variables.

We restrict ourselves to moderately small values of ϵ, let's say, in $(0.001,\ 0.1)$; we compute the solution of (38), (39), (40) by means of second-order central finite differences with a space step that is smaller than $\sqrt{\epsilon}$ in radial direction. If the space step in θ variable is significantly larger than the space step in radial direction, we can apply a fast (parallel) iterative solver of the discretized linear system [9] with appropriate relaxations.

In addition, we can compute once and for all a set of basis functions for the corrector $\hat{w}$ as follows: Let h_M be a discrete cosine expansion of h,

$$h_M = \sum_{j=0,\dots,M} a_j \cos(j\ \hat{\theta}).$$

Let $\hat{w}_j$ be a solution of (38) with boundary conditions

$$\hat{w}_j(0,\hat{\theta})\ =\ \cos(j\ \hat{\theta}),\ \hat{\theta} \in [0, 2\pi]$$

and (40) computed once for all. Then $\hat{w}$ is approximated by

$$\hat{w}_M = \sum_{j=0,\dots,M} a_j \hat{w}_j.$$

We illustrate now the *modulefou* technique to approximate the solution of the 2D heat equation in Ω with Dirichlet boundary equations

$$\frac{\partial u}{\partial t}\ =\ \Delta u\ +\ F \text{ in } \Omega,\ u\ =\ g \text{ on } \partial\Omega.$$

Let us consider an implicit first-order Euler scheme in time. For each time step, we have to solve

$$-dt\ \Delta u^{n+1} + u^{n+1} = u^n + dt\ F^{n+1} \text{ in } \Omega,\ u^{n+1}\ =\ g^{n+1} \text{ on } \partial\Omega.$$

We use the *modulefou* technique exactly as in the one space dimension case, except that v is computed by 2D Fourier transform, and w the corrector is computed numerically. In addition, one needs at each time step to extend u^n into a 2π periodic function $\bar{u}^n$ on $(0,\ 2\pi)^2$. To get the continuity of this extension is easy by linear interpolation. We obtain a better accuracy by using C^2 Hermite interpolation, based on decentered finite difference approximation of the normal derivatives of $\frac{\partial v^n}{\partial \rho^2}$ and $\frac{\partial w^n}{\partial \rho^2}$. It is not wise to compute second-order normal derivatives of v^n at $\partial\Omega$ by means of Fourier expansion because of the Gibbs phenomenon. Small oscillations are then amplified by the Hermite interpolation and the scheme becomes unstable.

Let us denote by N the number of Fourier modes used in each direction to approximate v^{n+1} and by M the number of Fourier mode used to approximate the boundary condition $h = g^{n+1} - \bar{v}^{n+1}$ on $\partial\Omega$ for the corrector problem.

We have tested the spatial accuracy of our method against an exact *steady* solution of the heat problem defined as

$$U(x,y)\ =\ \alpha \exp(-\beta(x^2+y^2))\ +\ \zeta,$$

with $\alpha,\ \beta,\ \zeta$ some real numbers. In addition the boundary of the domain is defined in this test case as $S(\theta) = 1\ +\ \gamma \cos(p\Theta)\ +\ \delta\ \exp(-\sin^2(\theta))$.

We have tried many different parameters: the following table gives a representative example of the error in maximum norm on the Cartesian grid inside the domain Ω with $\alpha = -0.5,\ \beta = 2,\ \zeta = 0.2,\ \gamma = 0.15,\ p = 3,\ \delta = 0.1$. We fix $M = 4\ N$ and 88 discretization points in radial direction for the computation of the corrector. The time step is 0.01. A smaller time step deteriorates the numerical accuracy for the *steady solution*. In order to have a rough idea of the efficiency of the method, we have included in the following table a comparison with second-order finite difference methods that use fitted grids at the boundary and the same regular grid as the *modulefou* method inside the domain.

Error with modulefou *for* $S(\theta) = 1 + 0.15\cos(3\theta) + 0.15\exp(-\sin^2(\theta))$.

N	$n = 0$	$n = 2$	$n = 4$	2^{nd} order finite differences
8	$1.3\ 10^{-3}$	$1.4\ 10^{-3}$	$1.6\ 10^{-3}$	$1.4\ 10^{-2}$
16	$5.2\ 10^{-4}$	$8.3\ 10^{-4}$	$6.1\ 10^{-4}$	$3.5\ 10^{-3}$
24	$2.7\ 10^{-4}$	$6.4\ 10^{-4}$	$3.9\ 10^{-4}$	$1.5\ 10^{-3}$
32	$3.0\ 10^{-4}$	$2.0\ 10^{-4}$	$1.7\ 10^{-4}$	$8.6\ 10^{-4}$
64	$2.5\ 10^{-4}$	$1.7\ 10^{-4}$	$1.9\ 10^{-4}$	$2.2\ 10^{-4}$

We observe that the numerical error stays almost the same when the number of modes is larger than 24. We speculate that our lower-order numerical approximation of the stiff corrector problem limits the accuracy of our method. This should be investigated further and improved with, for example, a pseudospectral discretization of (38). In particular, the same test case but for the disk geometry gives better results, but not as good as for the Poisson problem in section 2, for which the corrector is given by an analytical formula.

Error with modulefou *for the disk.*

N	$n = 0$	$n = 2$	$n = 4$
8	$2.6\ 10^{-3}$	$7.2\ 10^{-4}$	$1.6\ 10^{-3}$
16	$4.9\ 10^{-4}$	$3.5\ 10^{-5}$	$2.9\ 10^{-5}$
24	$1.\ 10^{-4}$	$2.0\ 10^{-5}$	$1.7\ 10^{-5}$
32	$1.1\ 10^{-4}$	$1.8\ 10^{-5}$	$1.6\ 10^{-5}$
64	$3.5\ 10^{-5}$	$1.3\ 10^{-5}$	$1.1\ 10^{-5}$

Similar results can be obtained with the Crank–Nicolson scheme instead of the implicit first-order Euler scheme.

Our second test is the computation of the wave equation in Ω with homogeneous Dirichlet boundary conditions and the second-order implicit scheme in time:

$$\frac{u^{n+1} - 2u^n + u^{n-1}}{dt^2} - \frac{\Delta u^{n+1} + \Delta u^{n-1}}{2} = F^n \text{ in } \Omega.$$

As mentioned before, this test case is of particular interest to investigate the eventual numerical unstability problems. The principle is the same except that we have to compute the Laplacian of u at each time step to get the rhs. Once again, we use the splitting $\Delta u^n = \Delta v^n + \Delta w^n$. The equation satisfied by the corrector give us $\Delta w^n = \frac{2\,w^n}{dt^2}$, and therefore the scheme is written

$$-\frac{dt^2}{2}\Delta u^{n+1} + u^{n+1} = 2u^n - u^{n-1} + \frac{dt^2}{2}\Delta v^{n-1} + w^{n-1}.$$

This can be written as

$$-\frac{dt^2}{2}\Delta u^{n+1} + u^{n+1} = 2u^n - v^{n-1} + \frac{dt^2}{2}\Delta v^{n-1}.$$

We finally obtain Δv^{n-1} either from its Fourier expansion or from the equation satisfied by v^{n-1}. Our numerous numerical experiments suggest the following heuristic: the *modulefou* scheme is numerically stable as long as the number of Fourier modes is large enough to represent the underline boundary layer of dt thickness in the periodic problem satisfied by v_N. For example, when $dt = 0.1$ and the diameter of Ω is two, we need 64 Fourier modes at least. The accuracy of our solution of the wave equation is therefore rather limited by the time discretization and the *modulefou* method is not a good method.

Finally, we observe that the arithmetic complexity of our algorithm for our two previous examples is dominated by the complexity of the 2D FFTs, once the set of basis function w_j defined above is computed. In particular, the linear combination of the w_j and the interpolation technique to extend the rhs into a periodic smooth rhs should concern only a small fraction of the grid points close to the boundary of the domain. In addition, our algorithm seems to be suitable for parallel computing for mainly two reasons: first FFT [17] has been very well optimized and standardized on parallel computers. Alternatively, one can avoid use of global FFT by using wavelet transforms [1]. Second, all extra operations as interpolation to smooth out the rhs and linear combination of the basis function w_j to correct v^n are local operators in the neighborhood of $\partial\Omega$.

6. Conclusion. The goal of this paper was to explore the implementation of various iterative schemes with the *modulefou* technique. This technique combines FFT and a solution of local problems via analytical or asymptotical or numerical technique. Numerical accuracy and stability of the methods has been tested against a large number of linear and nonlinear test cases. The results of our numerical experiments are quite encouraging and should motivate further analysis. More recent works show that these methods can be extended successfully to parallel computing by combined use of filtering and domain decomposition [10], [11], [12].

REFERENCES

[1] A. Averbuch, G. Beylkin, R. Coifman, and M. Israeli, *Multiscale inversion of elliptic operators*, the Samuel Neamen Workshop on Signal and Image Representation in Combined Space, Y. Zeevi and R. Coifman, eds., Academic Press, New York, 1998, pp. 341–359.

[2] E. M. De Jager, *Singular perturbation of hyperbolic type*, Nieuw Arch. Wisk. (3), 23 (1975), pp. 145–172.

[3] W. Eckhaus, *Asymptotic Analysis of Singular Perturbations*, North-Holland, Amsterdam, 1979.

[4] W. Eckhaus and M. Garbey, *Asymptotic analysis on large timescales for singular perturbation problems of hyperbolic type*, SIAM J. Math Anal., 21 (1990), pp. 867–883.

[5] A. Averbuch, M. Israeli, and L. Vozovoi, *Parallel implementation of non-linear evolution problems using parabolic domain decomposition method*, Parallel Comput., 21 (1995), pp. 1151–1183.

[6] M. Garbey, *Domain decomposition to solve layers and asymptotics*, SIAM J. Sci. Comput., 15 (1994), pp. 866–891.

[7] M. Garbey, *A Schwarz alternating procedure for singular perturbation problems*, SIAM J. Sci. Comput., 17 (1996), pp. 1175–1201.

[8] M. Garbey and D. Tromeur-Dervout, *A new parallel solver for the nonperiodic incompressible Navier-Stokes equations with a Fourier method: Application to frontal polymerization*, J. Comput. Phys., 145 (1998), pp. 316–331.

[9] M. Garbey and H. G. Kaper, *Heterogeneous domain decomposition for singularly perturbed elliptic boundary value problems*, SIAM J. Numer. Anal., 34 (1997), pp. 1513–1544.
[10] M. Garbey and D. Tromeur-Dervout, *Domain decomposition with local Fourier basis applied to combustion problems*, Parallel CFD Taiwan 98, A. Ecer et al., eds., North-Holland, Amsterdam, 1999, pp. 199–206.
[11] M. Garbey and D. Tromeur-Dervout, *Adaptive coupling codes: Application to combustion problems*, Parallel CFD Taiwan 98, A. Ecer et al., eds., North-Holland, Amsterdam, 1999, pp. 93–100.
[12] M. Garbey and D. Tromeur-Dervout, *Domain decomposition with local Fourier bases applied to frontal polymerisation problems*, in Proceedings DD11, Ch.-L. Lai et al., eds., DDM, Manchester, UK, 1998, pp. 242–250.
[13] R. Geel, *Singular Perturbation of Hyperbolic Type*, Thesis, University of Amsterdam, 1979.
[14] D. Gottlieb, M. Y. Hussaini, and S. A. Orzag, *Theory and applications of spectral methods*, in Spectral Methods for Partial Differential Equations, SIAM, Philadelphia, 1984, pp. 1–54.
[15] M. Israeli, L. Vozovoi, and A. Averbuch, *Domain decomposition methods with local Fourier basis for parabolic problems*, Contemp. Math., 157 (1994), pp. 223–230.
[16] M. Israeli, L. Vozovoi, and A. Averbuch, *Spectral multidomain technique with local Fourier basis*, J. Sci. Comput., 8 (1993), pp. 135–149.
[17] D. Keyes, A. Sameh, and V. Venkatakrishnan, eds., *Parallel Numerical Algorithms*, Kluwer Academic Publishers, 1997.
[18] J.-L. Lions, *Perturbations singulières dans les problèmes aux limites et en contrôle optimal*, Lecture Notes in Math. 323, Springer-Verlag, New York, 1973.
[19] J. G. Laforgue and R. E. O'Malley, Jr., *Supersensitive boundary value problems*, in Asymptotic and Numerical Methods for PDEs with Critical Parameters, NATO ASI Series C, Vol. 384, 1992, pp. 215–223.
[20] J. G. Laforgue and R. E. O'Malley, Jr, *Viscous shock motion for advection-diffusion equations*, Stud. Appl. Math., 95 (1995), pp. 147–170.
[21] P. D. Lax, *Hyperbolic Systems of Conservation Laws and the Mathematical Theory of Shock Waves*, SIAM Regional Conf. Ser. in Appl. Math. 11, SIAM, Philadelphia, 1973.
[22] A. J. Majda, V. Roytburd, and P. Colella, *Theoretical and numerical structure of reacting shock waves*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 1059–1080.
[23] L. G. Reyna and M. Ward, *On the exponentially slow motion of viscous shock*, Comm. Pure Appl. Math., 48 (1995), pp. 79–120.
[24] M. Roseau, *Asymptotic Wave Theory*, North-Holland Ser. Appl. Math. Mech. 20, North-Holland, Amsterdam, 1976.
[25] L. Vozovoi, M. Israeli, and A. Averbuch, *Spectral multidomain technique with local Fourier basis* II: *Decomposition into cells*, J. Sci. Comput., 9, (1994), pp. 311–325.
[26] G. B. Whitham, *Linear and Nonlinear Waves*, Wiley-Interscience, New York, 1974.

SIAM J. SCI. COMPUT.
Vol. 22, No. 3, pp. 1117–1138
© 2000 Society for Industrial and Applied Mathematics

EXTENSIBLE LATTICE SEQUENCES FOR QUASI-MONTE CARLO QUADRATURE*

FRED J. HICKERNELL†, HEE SUN HONG†, PIERRE L'ÉCUYER‡, AND CHRISTIANE LEMIEUX‡

Abstract. Integration lattices are one of the main types of low discrepancy sets used in quasi-Monte Carlo methods. However, they have the disadvantage of being of fixed size. This article describes the construction of an infinite sequence of points, the first b^m of which forms a lattice for any nonnegative integer m. Thus, if the quadrature error using an initial lattice is too large, the lattice can be extended without discarding the original points. Generating vectors for extensible lattices are found by minimizing a loss function based on some measure of discrepancy or nonuniformity of the lattice. The spectral test used for finding pseudorandom number generators is one important example of such a discrepancy. The performance of the extensible lattices proposed here is compared to that of other methods for some practical quadrature problems.

Key words. discrepancy, good lattice point sets, multidimensional, spectral test

AMS subject classifications. 65D30, 65D32

PII. S1064827599356638

1. Introduction.

Multidimensional integrals appear in a wide variety of applications in finance [4, 50, 52], physics and engineering [30, 43, 51, 60], and statistics [8, 12, 13]. The integration domain may often be assumed, after some appropriate transformation, to be the unit cube, in which case the integral takes the form

$$I(f) \equiv \int_{[0,1)^s} f(x)\,dx$$

for some known integrand, f.

Adaptive methods, such as [2], have been developed for approximating multidimensional integrals, but their performance deteriorates as the dimension increases. For finance problems the dimension can be in the hundreds or even thousands. An alternative to adaptive quadrature is Monte Carlo methods, where the integral is approximated by the sample mean of the integrand evaluated on a set, P, of N independent random points drawn from a uniform distribution on $[0,1)^s$:

$$Q(f) \equiv \frac{1}{N}\sum_{z\in P} f(z). \tag{1.1}$$

The quadrature error for Monte Carlo methods is typically $\mathrm{O}(N^{-1/2})$. One reason for this relatively low accuracy is that the points in P are chosen independently of each other. Thus, some parts of the integration domain contain clumps of points, while other parts are empty of points.

*Received by the editors June 18, 1999; accepted for publication (in revised form) April 14, 2000; published electronically September 27, 2000.

http://www.siam.org/journals/sisc/22-3/35663.html

†Department of Mathematics, Hong Kong Baptist University, Kowloon Tong, Hong Kong SAR, China (fred@hkbu.edu.hk, http://www.math.hkbu.edu.hk/~fred). The research of the first two authors was supported by a HKBU FRG grant 96-97/II-67.

‡Département d'Informatique et de Recherche Opérationnelle, Université de Montréal C.P. 6128, Succ. Centre-Ville, Montréal, QB, Canada, H3C 3J7 (lecuyer@iro.umontreal.ca, lemieux@iro.umontreal.ca).

To obtain greater accuracy one may replace the random set P by a carefully chosen deterministic set that is more uniformly distributed on $[0,1)^s$. As is explained in section 3, one may define a *discrepancy* that measures how much the empirical distribution function of P differs from the continuous uniform distribution. Then one chooses P in quadrature rule (1.1) to have as small a discrepancy as possible. The quadrature methods based on low discrepancy sets are called quasi-Monte Carlo methods. They are discussed in several review articles [3, 16, 41, 61] and monographs [28, 45, 55].

Two important families of low discrepancy sets are

i. integration lattices [45, Chap. 5] and [55], and
ii. digital nets and sequences [45, Chap. 4].

These two families are introduced in section 2. One advantage of the second family is that any number of consecutive points from a good digital sequence has low discrepancy. If one needs more points, one may use additional terms from the digital sequence without discarding the original ones. On the other hand, until now the number of points in an integration lattice has had to be specified in advance. So far, there has been no systematic way of adding points to an integration lattice while still retaining its lattice structure.

The purpose of this article is to provide a method for constructing *infinite lattice sequences*, thereby eliminating the need to know N, the number of points, in advance. Although the emphasis is on rank-1 lattices, the method may be applied to integration lattices of arbitrary rank. Given an infinite lattice sequence one may approximate a multidimensional integrand with a quadrature rule of the form (1.1) for a moderate number of points N_0. If the error estimate is unacceptably high, then one may choose an additional $N_1 - N_0$ points from the lattice sequence to obtain a quadrature rule with N_1 points and so on.

The following section describes the new method for obtaining infinite lattice sequences. Section 3 briefly reviews some results on discrepancy and quadrature error analysis for quasi-Monte Carlo methods. These are used to find the generating vectors for the new lattice sequences in section 4. The issue of error estimation is addressed in section 5. Two practical examples are explored in section 6, where the new lattice sequences are compared with existing quadrature methods. The last section contains some concluding remarks.

2. Integration lattices and digital sequences. This section begins by introducing integration lattices. Next, digital sequences and (t,s)-sequences are described. Finally, the idea underlying digital sequences is used to produce infinite lattice sequences.

2.1. Integration lattices. Rank-1 lattices, also known as good lattice point (GLP) sets, were introduced by Korobov [32] and have been widely studied since then (see [28, 45, 55] and the references therein). The formula for a shifted rank-1 lattice set is simply

$$P = \{\{ih/N + \Delta\} : i = 0, \dots, N-1\}, \tag{2.1}$$

where N is the number of points, h is an s-dimensional generating vector of integers (a GLP) that depends on N, Δ is an s-dimensional shift vector in $[0,1)^s$, and $\{x\}$ denotes the fractional part of a vector x, i.e., $\{x\} = x \mod 1$. Sloan [54] and Sloan and Kachoyan [56] later generalized GLP sets by introducing more than one generating vector. A shifted integration lattice with $N = N_1 \dots N_p$ points based on generating

vectors $h^{(1)}, \ldots, h^{(p)}$ is

$$P = \left\{ \{ i_1 h^{(1)}/N_1 + \cdots + i_p h^{(p)}/N_p + \Delta \} : i_k = 0, \ldots, N_k - 1,\ k = 1, \ldots, p \right\}.$$

Integration lattices and their use for quadrature are discussed in the monograph [55].

For a given N there is the problem of choosing good generating vectors. Although theoretical constructions exist for $s = 1$ and 2, in higher dimensions one typically finds generating vectors by minimizing a discrepancy or measure of nonuniformity of the lattice. Several examples of discrepancies are given in section 3.

2.2. Digital nets and sequences. Digital nets and sequences are another method of constructing low discrepancy sets (see [33] and [45, Chapter 4]). Let b denote a positive integer greater than one. For any nonnegative integer one may extract the digits of its base b representation, $i_1, i_2, \ldots \in \{0, \ldots, b-1\}$, only finitely many of which are nonzero:

$$i = \ldots i_3 i_2 i_1 \text{ (base } b) = i_1 + i_2 b + i_3 b^2 + \cdots . \tag{2.2a}$$

The ith term of a digital net or sequence is given by

$$z^{(i)} = (z_1^{(i)}, \ldots, z_s^{(i)}), \tag{2.2b}$$

$$z_j^{(i)} = 0.\zeta_{j1}^{(i)} \zeta_{j2}^{(i)} \ldots \text{(base } b) = \zeta_{j1}^{(i)} b^{-1} + \zeta_{j2}^{(i)} b^{-2} + \cdots , \tag{2.2c}$$

where

$$\begin{pmatrix} \zeta_{j1}^{(i)} \\ \zeta_{j2}^{(i)} \\ \vdots \end{pmatrix} = \mathbf{C}_j \begin{pmatrix} i_1 \\ i_2 \\ \vdots \end{pmatrix} \mod b, \quad j = 1, \ldots, s. \tag{2.2d}$$

If the generating matrices $\mathbf{C}_1, \ldots, \mathbf{C}_s$ are $m \times m$, then this construction yields a digital net $\{z^{(i)} : i = 0, \ldots, b^m - 1\}$ with b^m points. If the generating matrices are $\infty \times \infty$, i.e., each $\mathbf{C}_j$ has entries c_{jkl} defined for $k, l = 1, 2, \ldots$, then one has a digital sequence $\{z^{(i)} : i = 0, 1, \ldots\}$.

The prototype digital sequence is the one-dimensional Van der Corput sequence, $\{\phi_b(i) : i = 0, 1, \ldots\}$. This is defined by taking $s = 1$ and $\mathbf{C}_1$ equal to the identity matrix:

$$\phi_b(i) = 0.i_1 i_2 \ldots \text{ (base } b) = i_1 b^{-1} + i_2 b^{-2} + \cdots . \tag{2.3}$$

In essence, the Van der Corput sequence takes the b-ary representation of an integer and reflects it about the decimal point.

2.3. (t, m, s)-nets and (t, s)-sequences. Similar to integration lattices one has the problem of how to choose the generating matrices $\mathbf{C}_j$ in (2.2). Usually this is done to optimize the quality factor of the net or sequence. For any nonnegative s-vector k of integers, and for a base b, consider the following set of disjoint boxes whose union is the unit cube:

$$\mathcal{B}_k = \{[n_1 b^{-k_1}, (n_1 + 1) b^{-k_1}) \times \cdots \times [n_s b^{-k_s}, (n_s + 1) b^{-k_s}) : n_j = 0, \ldots, b^{k_j} - 1\}. \tag{2.4}$$

Each such box in $\mathcal{B}_k$ has volume $b^{-k_1-\cdots-k_s}$. A (t,m,s)-net in base b is a set of $N=b^m$ points in $[0,1)^s$ such that every box in $\mathcal{B}_k$ contains $b^{m-k_1-\cdots-k_s}$ of these points for any k satisfying $m-k_1-\cdots-k_s \ge t$. Thus, any function that is piecewise constant on the boxes in $\mathcal{B}_k$ will be integrated exactly according to quadrature rule (1.1) if

$$k_1+\cdots+k_s \le m-t. \tag{2.5}$$

The integer parameter t is called the quality parameter of the net and it takes values between 0 and m. A smaller value of t means a better net.

A (t,s)-sequence is an infinite sequence of points in $[0,1)^s$ such that the b^m points numbered lb^m to $(l+1)b^m-1$ always form a (t,m,s)-net for any nonnegative integer l for all integers $m \ge t$. By using a (t,s)-sequence to do quasi-Monte Carlo calculations, one need not know the number of points required in advance. If the first b^{m_1} points do not give sufficient accuracy, then one may add the next $b^{m_2}-b^{m_1}$ points in the sequence to get a net with b^{m_2} points, without throwing away the first b^{m_1} points.

There is a connection between digital nets and sequences as defined above and (t,m,s)-nets and (t,s)-sequences [33]. Let $c_{ji\bullet}^{(m)}$ denote the vector containing the first m elements of the ith row of the generating matrix $\mathbf{C}_j$. Given a nonnegative integer s-vector k, let $\mathcal{C}(m,k)$ be the following set of the first k_j rows of the jth generating matrix for $j=1,\ldots,s$:

$$\mathcal{C}(m,k)=\left\{\mathbf{c}_{ji\bullet}^{(m)} : i=1,\ldots,k_j,\ j=1,\ldots,s\right\}.$$

Furthermore, let

$$\begin{aligned} T(m,s) &= \min_t\{t : \mathcal{C}(m,k) \text{ is a linearly independent set of vectors} \\ &\qquad\qquad \text{for all } k \text{ with } k_1+\cdots+k_s=m-t\}, \\ T(s) &= \max_m T(m,s). \end{aligned}$$

The following theorem (see [33]) gives the condition for which a digital net is a (t,m,s)-net and a digital sequence is a (t,s)-sequence.

THEOREM 2.1. *For prime bases b the digital net defined above in* (2.2) *is a $(T(m,s),m,s)$-net. If, in addition, $T(s)$ is finite, then the digital sequence defined in* (2.2) *is a $(T(s),s)$-sequence.*

Finding good generating matrices for digital nets and sequences is an active area of research. Virtually all generators found so far have been based on number theoretic arguments. Early sequences include those of Sobol' [58], Faure [9], and Niederreiter [44]. Algorithms for these sequences can be found in the ACM Transactions on Mathematical Software collection. The FINDER software developed at Columbia University by Traub and Papageorgiou implements generalized Sobol' and generalized Faure sequences. New constructions with smaller t values are given by Niederreiter and Xing [48].

2.4. Infinite lattice sequences. The idea underlying digital sequences may be extended to integration lattices to obtain infinite lattice sequences. The ith term of a rank-1 lattice, which is $\{ih/N\}$, depends inherently on the number of points, N. Thus, the formula for a lattice must be rewritten in a way that does not involve N explicitly. A way to do this was first suggested in [24].

Suppose that the number of points, N, is some integer power of a base $b \geq 2$, i.e., $N = b^m$. This is the same assumption as for a digital or (t, m, s)-net. The first N values of the Van der Corput sequence, defined in (2.3), are

$$\frac{0}{b^m}, \ldots, \frac{b^m - 1}{b^m},$$

although in a different order. Therefore, the term $i/N = i/b^m, i = 0, \ldots, N-1$, that appears in the definition of the rank-1 lattice may be replaced by $\phi_b(i)$, a term that does not depend on N.

The s-dimensional generating vector h in (2.1) typically depends on N also. It may be expressed in b-ary form as

$$h = (h_{1m} \ldots h_{11},\ h_{2m} \ldots h_{21},\ \ldots,\ h_{sm} \ldots h_{s1}) \quad (\text{base } b),$$

where $h_{jk} \in \{0, \ldots, b-1\}$ are digits. For $k > m$ the digits h_{jk} do not affect the definition of the rank-1 lattice set with $N = b^m$ points since they contribute only integers to the product $\phi_b(i)h$. Therefore, each component of h may be written (in principle) as an infinite string of digits:

$$h = (\ldots h_{12}h_{11},\ \ldots h_{22}h_{21},\ \ldots,\ \ldots h_{s2}h_{s1}) \quad (\text{base } b). \tag{2.6}$$

This single "infinite" generating vector may serve for all possible values of m.

The preceding paragraphs provide the basis for defining an infinite rank-1 lattice sequence. Altering the original definition in (2.1) leads to the following definition.

DEFINITION 2.2. *An infinite rank-1 lattice sequence in base b with generating vector h of the form* (2.6) *and shift Δ is defined as*

$$\{\{\phi_b(i)h + \Delta\} : i = 0, 1, 2, \ldots\}. \tag{2.7}$$

The first b^m terms of the infinite rank-1 lattice sequence (2.7) are rank-1 lattices. Moreover, just as certain subsets of a (t, s)-sequence are (t, m, s)-nets, so subsets of an infinite rank-1 lattice sequence are shifted rank-1 lattices.

THEOREM 2.3. *Suppose that P is the set consisting of the $(l+1)$st run of b^m terms of the infinite lattice rank-1 sequence defined in* (2.7).

$$P = \{\{\phi_b(lb^m + i)h + \Delta\} : i = 0, \ldots, b^m - 1\}, \quad l = 0, 1, \ldots.$$

Then, P is a rank-1 lattice with shift $\phi_b(l)b^{-m-1}h + \Delta$, i.e.,

$$P = \left\{\{\phi_b(i)h + \phi_b(l)b^{-m-1}h + \Delta\} : i = 0, \ldots, b^m - 1\right\}.$$

Proof. The proof follows directly from the definition of the Van der Corput sequence. For all $i = 0, \ldots, b^m - 1$, note that

$$\phi_b(lb^m + i) = \phi_b(i) + \phi_b(lb^m) = \phi_b(i) + \phi_b(l)b^{-m-1}.$$

Substituting the right-hand side into the definition of P completes the proof. ∎

The definition of an infinite rank-1 lattice sequence may be extended to integration lattices of arbitrary rank.

DEFINITION 2.4. *An infinite lattice sequence (of arbitrary rank) with bases $b_1, \ldots, b_p$ and generating vectors $h^{(1)}, \ldots, h^{(p)}$ of the form* (2.6) *and shift Δ is defined as*

$$\left\{\{\phi_{b_1}(i_1)h^{(1)} + \cdots + \phi_{b_p}(i_p)h^{(p)} + \Delta\} : i_1, \ldots, i_p = 0, 1, 2, \ldots\right\} \tag{2.8}$$

A practical complication for an integration lattice of rank greater than 1 is that there are multiple indices, i_k, each of which may or may not tend to infinity, and each at its own rate. Because of this complication we will focus on rank-1 lattices in the sections that follow. Theorem 2.3 also has a natural extension to infinite lattice sequences of arbitrary rank, and its proof is similar.

3. Discrepancy. Unlike (t, m, s)-nets for which there exist explicit constructions of the generating matrices $\mathbf{C}_j$, there are no such explicit constructions of generating vectors h for rank-1 lattices for arbitrary s. Tables of generating vectors for lattices that do exist (see [8, 14, 28]) are usually obtained by minimizing some measure of nonuniformity, or discrepancy, of the lattice. This section describes several useful discrepancy measures.

Let $\mathrm{Err}(f; P)$ denote the quadrature error for a rule of the form (1.1) for an arbitrary set P. Worst-case error analysis of the quadrature error leads to a Koksma–Hlawka-type inequality of the form [23]

$$|\mathrm{Err}(f; P)| \equiv |I(f) - Q(f)| \le D(P)V(f), \tag{3.1}$$

where $D(P)$ is the discrepancy or measure of nonuniformity of the point set defining the quadrature rule, and $V(f)$ is the variation or fluctuation of the integrand, f. The precise definitions of the discrepancy and the variation depend on the particular space of integrands.

In the traditional Koksma–Hlawka inequality (see [27] and [45, Thm. 2.11]), the variation is the variation in the sense of Hardy and Krause, and the discrepancy is the $\mathcal{L}_\infty$-star discrepancy:

$$D^*(P) = \|F_{\mathrm{unif}}(x) - F_P(x)\|_\infty = \left\| x_1 \dots x_s - \frac{|P \cap [0, x]|}{N} \right\|_\infty . \tag{3.2}$$

Here F_{unif} is the uniform distribution on the unit cube, F_P is the empirical distribution function for the sample P, and $|\cdot|$ denotes the number of points in a set. The notation $\|\cdot\|_p$ denotes the $\mathcal{L}_p$-norm or the ℓ_p-norm, depending on the context. Error bounds of the form (3.1) involving the $\mathcal{L}_p$-star discrepancy have been derived by [59, 64]. Error bounds involving generalizations of the star discrepancy appear in [21, 22, 23, 57].

When the integrands belong to a reproducing kernel Hilbert space, error bound (3.1) may be easily obtained [21, 23]. The discrepancy may be written in terms of the reproducing kernel

(3.3)

$$\begin{aligned} D(P) &= \left\{ \int_{[0,1)^{2s}} K(x, y)\ d(F_{\mathrm{unif}} - F_P)(x)\ d(F_{\mathrm{unif}} - F_P)(y) \right\}^{1/2} \\ &= \left\{ \int_{[0,1)^{2s}} K(x, y)\ dx\ dy - \frac{2}{N} \sum_{z \in P} \int_{[0,1)^s} K(z, y)\ dy + \frac{1}{N^2} \sum_{z,t \in P} K(z, t) \right\}^{1/2} . \end{aligned}$$

For example, the $\mathcal{L}_2$-star discrepancy, whose formula was originally derived in [62], is a special case of the above formula with $K(x, y) = \prod_{j=1}^{s} [1 - \max(x_j, y_j)]$. An advantage of considering reproducing kernel Hilbert spaces of integrands is that the computational complexity of the discrepancy is relatively small (at worst $\mathrm{O}(N^2)$ operations). By contrast the $\mathcal{L}_\infty$-star discrepancy requires $\mathrm{O}(N^s)$ operations to evaluate.

The discrepancy of type (3.3) can also be interpreted as an average-case quadrature error [25, 42, 63]. Suppose that the integrand is a random function lying in the sample space $\mathcal{A}$, and suppose that the integrand has zero mean and covariance kernel, $K(x, y)$, i.e.,

$$E_{f\in\mathcal{A}}[f(x)] = 0 \text{ and } E_{f\in\mathcal{A}}[f(x)f(y)] = K(x, y) \quad \forall x, y \in [0, 1)^s.$$

Then the root mean square quadrature error over $\mathcal{A}$ is the discrepancy as defined in (3.3):

$$\sqrt{E_{f\in\mathcal{A}}[\mathrm{Err}(f; P)]^2} = \sqrt{E_{f\in\mathcal{A}}\left|I(f) - Q(f)\right|^2} = D(P). \tag{3.4}$$

If P is a simple random sample, then the mean square discrepancy is [25]

$$E_P\{[D(P)]^2\} = \frac{1}{N}\left\{\int_{[0,1)^s} K(x, x)\ dx - \int_{[0,1)^s\times[0,1)^s} K(x, y)\ dy\ dx\right\}. \tag{3.5}$$

This formula serves as a benchmark for other (presumably superior) low discrepancy sets. Since the mean square discrepancy is $\mathrm{O}(N^{-1})$, the discrepancy itself is typically $\mathrm{O}(N^{-1/2})$ for a simple random sample. The variance of a function, f, may be defined as

$$\mathrm{Var}(f) \equiv \int_{[0,1)^s} f^2\ dx - \left(\int_{[0,1)^s} f\ dx\right)^2.$$

The mean value of the variance over the space of average-case integrands can be shown to be [25]

$$E_{f\in\mathcal{A}}[\mathrm{Var}(f)] = \int_{[0,1)^s} K(x, x)\ dx - \int_{[0,1)^{2s}} K(x, y)\ dy\ dx, \tag{3.6}$$

which is just the term in braces in (3.5).

It may seem odd at first that the discrepancy can serve both as an average-case and worst-case quadrature error. The explanation is that the space of integrands, $\mathcal{A}$, in the average-case analysis is much larger than the space of integrands, $\mathcal{W}$, in the worst-case analysis. See [21, 25] and the references therein for the proofs of the above results as well as further details.

There are some known asymptotic results for the discrepancies of (t, m, s)-nets. The $\mathcal{L}_\infty$-star discrepancy of any (t, m, s)-net is $\mathrm{O}(N^{-1}[\log N]^{s-1})$ [45, Thm. 4.10]. Moreover, the typical (in the sense of an average taken over all possible nets) $\mathcal{L}_2$-star discrepancy of (t, m, s)-nets is $\mathrm{O}(N^{-1}[\log N]^{(s-1)/2})$ [19, 26]—the best possible for any set. For discrepancies of the form (3.3) with sufficiently smooth kernels, typical (t, m, s)-nets have $\mathrm{O}(N^{-3/2}[\log N]^{(s-1)/2})$ discrepancy [25, 26].

Lattice rules, the topic of this article, are known to be particularly effective for integrating periodic functions. Suppose that the integrand has an absolutely convergent Fourier series with Fourier coefficients $\hat{f}(k)$:

$$f(x) = \sum_{k\in\mathbf{Z}^s} \hat{f}(k)e^{2\pi ik'x}, \qquad \hat{f}(k) = \int_{[0,1)^s} f(x)e^{-2\pi ik'x}\ dx. \tag{3.7}$$

Here $k'x$ denotes the dot product of the s-dimensional wavenumber vector k with x.

The quadrature error for a particular integrand with an absolutely convergent Fourier series is simply the sum of the quadrature errors of each term:

$$\mathrm{Err}(f;P) = -\sum_{\substack{k\in\mathbf{Z}^s\\ k\neq 0}} \hat{f}(k)\left[\frac{1}{N}\sum_{z\in P} \mathrm{e}^{2\pi i k'z}\right],$$

where the term corresponding to $k=0$ does not enter because constants are integrated exactly. One can multiply and divide by arbitrary weights $w(k)$ inside this sum. Then by applying Hölder's inequality one has the following error bound of the form (3.1) [23]:

$$|\mathrm{Err}(f;P)| \le D_{\mathcal{F},p}(P) V_{\mathcal{F},q}(f), \quad \frac{1}{p}+\frac{1}{q}=1, \tag{3.8a}$$

$$D_{\mathcal{F},p}(P) = \left\| \left(\frac{1_{\{k\neq 0\}}}{w(k)} \left[\frac{1}{N}\sum_{z\in P} e^{2\pi i k'z} \right] \right)_{k\in\mathbf{Z}^s} \right\|_p, \tag{3.8b}$$

$$V_{\mathcal{F},q}(f) = \left\| \left(1_{\{k\neq 0\}} w(k)\hat{f}(k)\right)_{k\in\mathbf{Z}^s} \right\|_q. \tag{3.8c}$$

Here $1_{\{\cdot\}}$ denotes the indicator function. In order to insure that the discrepancy is finite we assume that the weights increase sufficiently fast as k tends to infinity:

$$\left\| \left(\frac{1}{w(k)} \right)_{k\in\mathbf{Z}^s} \right\|_p < \infty.$$

If P is the node set of an integration lattice, then it is known that trigonometric polynomials are integrated exactly for all nonzero wavenumbers *not* in the dual lattice, $L^\perp$ (see [55, Lem. 2.7]):

$$\frac{1}{N}\sum_{z\in P} e^{2\pi i k'z} = 1_{\{k\in L^\perp\}}.$$

The dual lattice is the set of all k satisfying $k'z = 0 \pmod 1$ $\forall z \in P$. Thus, for node sets of lattices the definition of discrepancy above may be simplified to

$$D_{\mathcal{F},p}(P) = \left\| \left(\frac{1_{\{0\neq k\in L^\perp\}}}{w(k)} \right)_{k\in\mathbf{Z}^s} \right\|_p.$$

Certain explicit choices of $w(k)$ have appeared in the literature. For example, one may choose

$$w(k) = \left[\overline{(\beta_1^{-1}k_1)}\cdots\overline{(\beta_s^{-1}k_s)}\right]^\alpha, \quad \alpha>0, \tag{3.9}$$

where the over-bar notation is defined as

$$\bar{k}_j = \begin{cases} |k_j| & \text{for } k_j \neq 0, \\ 1 & \text{for } k_j = 0, \end{cases} \tag{3.10}$$

$\beta_1,\ldots,\beta_s$ are arbitrary positive weights, and α is a measure of the assumed smoothness of the integrand. If $\beta_1 = \cdots = \beta_s = 1$ and P is the node set of a lattice, then

$D_{\mathcal{F},p}(P) = [P_{\alpha p}(L)]^{1/p}$ for $1 \le p < \infty$, where $P_\gamma(L)$ is a traditional figure of merit for lattices [55]. Furthermore, for $p = \infty$ the discrepancy is $D_{\mathcal{F},\infty}(P) = [\rho(L)]^{-\alpha}$, where $\rho(L)$ is the *Zaremba figure of merit* for lattice rules [45, Def. 5.31]. The more general case of P that is not a lattice is considered in [20], and the case of unequal weights β_j is discussed in [22].

If the weight function $w(k)$ takes the form (3.9) for *positive integer* α, then for $p = q = 2$ the infinite sum defining the discrepancy in (3.8b) may be written as a finite sum:

$$D_{\mathcal{F},2}(P) = \left\{ -1 + \frac{1}{N^2} \sum_{z,t \in P} \prod_{j=1}^{s} \left[1 - \frac{(-4\pi^2 \beta_j^2)^\alpha}{(2\alpha)!} B_{2\alpha}(\{z_j - t_j\}) \right] \right\}^{1/2}, \tag{3.11a}$$

for general sets P, where $B_{2\alpha}$ denotes the Bernoulli polynomial of degree 2α [1, Chap. 23]. When P is the node set of an integration lattice, the double sum can be simplified to a single sum:

$$D_{\mathcal{F},2}(P) = \left\{ -1 + \frac{1}{N} \sum_{z \in P} \prod_{j=1}^{s} \left[1 - \frac{(-4\pi^2 \beta_j^2)^\alpha}{(2\alpha)!} B_{2\alpha}(z_j) \right] \right\}^{1/2}. \tag{3.11b}$$

Another choice for $w(k)$ is a weighted ℓ_r-norm of the vector k to some power:

$$w(k) = \left[\left(\frac{k_1}{\beta_1} \right)^r + \cdots + \left(\frac{k_s}{\beta_s} \right)^r \right]^{\alpha/r}, \quad \alpha > 0,$$

again for arbitrary positive weights $\beta_1, \ldots, \beta_s$. When these weights are unity, P is the node set of a lattice, and $p = \infty$, then

$$D_{\mathcal{F},\infty}(P) = \max_{0 \ne k \in L^\perp} \|k\|_r^{-\alpha} = \left\{ \min_{0 \ne k \in L^\perp} \|k\|_r \right\}^{-\alpha}. \tag{3.12}$$

For $r = 2$ this discrepancy is equivalent to the *spectral test*, commonly employed to measure the quality of linear congruential pseudorandom number generators [31, 34]. The spectral test has been used to select lattices for quasi-Monte Carlo quadrature in [7, 35, 36, 37]. The case $r = 1$, which one might call an ℓ_1-spectral test, is also interesting. We will return to these two cases in the next section.

4. Good generating vectors for lattice sequences. As mentioned at the beginning of the previous section, finding good generating vectors for lattices typically requires optimizing some discrepancy measure. In this subsection we propose some loss functions and optimization algorithms for choosing good generating vectors for extensible rank-1 lattice sequences.

In principle one would like to have an $\infty \times \infty$ array of digits h_{jk}. However, in practice it is only necessary to have an $s_{\max} \times m_{\max}$ array of digits h_{jk}, where $N_{\max} = b^{m_{\max}}$ is the maximum number of points and $s_{\max}$ is the maximum dimension to be considered. In finance calculations, for example, the necessity of timely forecasts may constrain one to a budget of $10^3 - 10^4$ points [49].

For simplicity we consider generating vectors h that are of the form originally proposed by Korobov, i.e.,

$$h = (1, \eta, \eta^2, \ldots, \eta^{s-1}). \tag{4.1}$$

This means that only the digits $\eta_1, \eta_2, \ldots, \eta_{m_{\max}}$ need to be chosen, for which there are $b^{m_{\max}} = N_{\max}$ choices. The generating vector is tested for dimensions up to $s_{\max}$, but in fact it can be extended to any dimension.

The number η defining the generating vector is chosen by minimizing a loss function, G, of the form

$$G(\eta) = \max_{m,s} \tilde{G}(\eta, m, s). \tag{4.2}$$

Here the function $\tilde{G}(\eta, m, s)$ is related to one of the measures of discrepancy introduced in section 3, and the maximum over some range of values of m and s insures that the resulting generating vector is good for a range of numbers of points and dimensions. However, since the discrepancy itself depends significantly on the number of points and the dimension, it must be appropriately scaled to arrive at the function $\tilde{G}$. The details of this scaling are given below.

4.1. Generating vectors based on minimizing P_α. The discrepancy defined in (3.11), which is a generalization of the P_α figure of merit for lattice rules, has the advantage of requiring only $\mathrm{O}(sN)$ operations to evaluate for lattices. To remove some of the dimension dependence of this discrepancy it is divided by the square root of right-hand side of (3.6). The root mean square of this scaled discrepancy for a random sample is then $N^{-1/2}$, independent of s. The formula for the scaled discrepancy of the node set of a lattice with $\alpha = 1$ and $4\pi^2\beta_j^2 = 6$ is

$$D(P) = \left[\left(\frac{3}{2}\right)^s - 1\right]^{-1/2} \left\{-1 + \frac{1}{N}\sum_{z\in P}\prod_{j=1}^{s}\left[1 + 3B_2(z_j)\right]\right\}^{1/2}.$$

The specific choice of the value of β_j here is not crucial but seems to give good results.

For one-dimensional lattices, i.e., evenly spaced points on the interval $[0, 1)$, this discrepancy is N^{-1}, and one would expect that as the dimension increases this scaled discrepancy would tend to (or at least do no worse than) $N^{-1/2}$. Therefore, to remove this remaining dimension dependence the above scaled discrepancy is divided by the function

$$D_{\mathrm{asy}}(m, s) = b^{-m}\left(1 + \frac{m\log b}{s-1}\right)^{(s-1)/2} = N^{-1}\left(1 + \frac{\log N}{s-1}\right)^{(s-1)/2}.$$

For a fixed s, D_{asy}(m,s) is asymptotically $\mathrm{O}(b^{-m}m^{(s-1)/2}) = \mathrm{O}(N^{-1}[\log N]^{(s-1)/2})$ as N tends to infinity. This is the asymptotic order for $(0, m, s)$-nets [25], and what we hope to achieve for lattice sequences. Furthermore, $N^{-1} = b^{-m} \le D_{\mathrm{asy}}(m, s) \le b^{-m/2} = N^{-1/2}$ for any m and s. In summary, the resulting loss function is

$$\begin{aligned}\tilde{G}_1(\eta, m, s) = \frac{D(P)}{D_{\mathrm{asy}}(m, s)} &= b^m\left(1 + \frac{m\log b}{s-1}\right)^{-(s-1)/2}\left[\left(\frac{3}{2}\right)^s - 1\right]^{-1/2} \\ &\quad\times\left\{-1 + \frac{1}{b^m}\sum_{i=0}^{b^m-1}\prod_{j=1}^{s}\left[1 + 3B_2(\{\phi_b(i)\eta^{j-1}\})\right]\right\}^{1/2}.\end{aligned} \tag{4.3}$$

The optimal values of η found by minimizing $G_1(\eta)$ for $b = 2$ and for different ranges of m and s are given in Table 4.1. The algorithm for optimizing $G_1(\eta)$ may

be described as an intelligent exhaustive search. One need not compute $G_1(\eta)$ for all possible values of η. Suppose at any stage of the optimization η_* is the best known value of η, and one finds that $\tilde{G}_1(\tilde{\eta}, m, s) > G_1(\eta_*)$ for some $\tilde{\eta}$. Since $\tilde{G}_1(\eta, m, s)$ depends only on the first $m-1$ digits of η, one can immediately eliminate from consideration all η that have the same first $m-1$ digits as $\tilde{\eta}$. This same search strategy is also used for the other loss functions described below.

TABLE 4.1
Values of η defining generating vectors of the form (4.1) *for rank-*1 *lattice sequences with base $b = 2$, and minimizing* (4.9).

i	m_0	m_1	s_1	η	$G_i(\eta, m_0, m_1, s_1)$	$G_{i,\min}(\eta, m_0, m_1, s_1)$
1	0	17	32	17797	4.11	0.78
1	0	12	32	203	2.33	0.78
1	0	16	32	17797	3.57	0.78
1	13	20	32	407641	5.86	0.97
2	0	17	25	1267	2.05	1.00
2	1	16	34	12915	1.90	1.00
2	6	12	33	1571	1.88	1.10
2	8	17	32	11335	1.83	1.03
2	14	20	37	156487	1.76	1.05
2	15	24	32	4450341	1.79	1.07
3	6	16	12	237	2.25	1.07
3	6	12	10	53	2.25	1.14
3	10	16	11	173	2.25	1.15
3	14	21	11	391485	2.00	1.01
3	16	24	10	1459305	2.10	1.11

4.2. Generating vectors based on the spectral test. The use of the spectral test to analyze the lattice structure of linear congruential generators is described in [31], and tables of good integration lattices are given in [35]. The difference here is that nearly all smaller lattices imbedded in the largest lattice considered must have low discrepancy. In [35], only the full lattice was examined.

The length of the shortest nonzero vector in the dual lattice $L^\perp$ is

$$d_2(\eta, m, s) = \min\{\|k\|_2 : k \neq 0,\ k'(1, \eta, \ldots, \eta^{s-1}) = 0 \mod b^m\}, \tag{4.4}$$

which is related to the discrepancy (3.12) with $r = 2$. This length has the absolute upper bound

$$d_2^*(m, s) = \begin{cases} \gamma_s b^{m/s}, & 1 \leq s \leq 8, \\ \rho_s b^{m/s}, & 8 < s, \end{cases} \tag{4.5}$$

where the constants γ_s and ρ_s depend only on s (see [35] and the references therein). The bound for $s \leq 8$ is the least upper bound for a general s-dimensional lattice with real-valued coordinates and with b^{-m} points per unit of volume. The bound for $s > 8$ is not the least upper bound, but it is still reasonably tight, as our numerical results will show. We define the normalized ℓ_2-spectral test discrepancy as

$$\tilde{G}_2(\eta, m, s) = \frac{d_2^*(m, s)}{d_2(\eta, m, s)}, \tag{4.6}$$

which is larger than 1 and is the inverse of the quantity S_t defined in [35]. (The different notation here is to be consistent with the rest of this article.) The loss function to be minimized is of the form (4.2) with $\tilde{G} = \tilde{G}_2$.

We note that $1/d_2(\eta, m, s)$ can be interpreted as the (Euclidean) distance between the successive hyperplanes that contain all the points of the *primal* lattice L for the family of hyperplanes for which this distance is the largest. The problem of computing a shortest vector in (4.4) can be formulated as a quadratic optimization problem with s integer decision variables, because k can be written as a linear combination of the s vectors of a basis of the dual lattice, with integer coefficients. The decision variables are these coefficients. (See [31] for details.) We solved this problem by using the branch-and-bound algorithm of Fincke and Pohst [10], with a few heuristic modifications to improve the speed. The *worst-case* time complexity of this algorithm is exponential in $d_2(\eta, m, s)$, and polynomial in s for $d_2(\eta, m, s)$ fixed [10]. In practice, it (typically) works nicely even when $d_2(\eta, m, s)$ is large. For example, one can compute $d_2(\eta, 30, s)$ for $b = 2$, an arbitrary η and $s = 2, \ldots, 30$ in less than 1 second on a Pentium-II computer.

4.3. Generating vectors based on the ℓ_1-spectral test. With the ℓ_1 norm, the length of the shortest nonzero vector in $L^\perp$ is

$$d_1(\eta, m, s) = \min\{\|k\|_1 : k \neq 0,\ k'(1, \eta, \ldots, \eta^{s-1}) = 0 \mod b^m\}, \tag{4.7}$$

which is related to the discrepancy (3.12) with $r = 1$. One has the upper bound

$$d_1(\eta, m, s) \leq d_1^*(m, s) \stackrel{\text{def}}{=} (s! b^m)^{1/s},$$

which was established by Marsaglia [38] by applying the general convex body theorem of Minkowski. This suggests the normalized ℓ_1-spectral test quantity:

$$\tilde{G}_3(\eta, m, s) = \frac{d_1^*(m, s)}{d_1(\eta, m, s)}. \tag{4.8}$$

Here, we want to minimize the loss function (4.2) with $\tilde{G} = \tilde{G}_3$.

One can interpret $d_1(\eta, m, s)$ (or $d_1(\eta, m, s) - 1$ in certain cases; see [31]) as the minimal number of hyperplanes that cover all the points of P. We computed $d_1(\eta, m, s)$ via the algorithm of Dieter [6], which works fine for s up to about 10 (independently of m), but becomes very slow for larger s (the time is exponential in s).

The ℓ_1-spectral test quantity in (4.8) has an interpretation similar to that of the quality parameter t for (t, m, s)-nets. Define

$$T(\eta, m, s) = \log_b(\tilde{G}_3(\eta, m, s)) = \frac{m + \log_b(s!)}{s} - \log_b(d_1(\eta, m, s)).$$

Since $\tilde{G}_3 \geq 1$ and $d_1(\eta, m, s) \geq 1$, it follows that $0 \leq T(\eta, m, s) \leq [m + \log_b(s!)]/s$. Thus, the rank-1 lattice defined by η integrates exactly all trigonometric polynomials of wavenumber k when

$$\log(|k_1| + \cdots + |k_s|) \leq \frac{m + \log_b(s!)}{s} - T(\eta, m, s).$$

If one considers $T(\eta, m, s)$ as the quality parameter of the lattice, then this condition is similar to that in (2.5). There, t determines the resolution at which piecewise

constant functions are exactly integrated by a net. Here, $T(\eta, m, s)$ determines the resolution at which trigonometric polynomials are integrated exactly by a lattice.

The discrepancy for the node set of this lattice as defined in (3.12) with $r = 1$ is

$$D_{\mathcal{F},\infty}(P) = d_1(\eta, m, s)^{-\alpha} = (s!)^{-\alpha/s} b^{\alpha T(\eta,m,s)} N^{-\alpha/s}.$$

If one can construct an infinite sequence of digits, η, for which

$$T(\eta, s) = \max_m T(\eta, m, s) < \infty,$$

then the above discrepancy decays like $N^{-\alpha/s}$. Again, the parameter α indicates the assumed smoothness of the integrands.

4.4. Tables of coefficients. We made computer searches to find the best η's, based on minimizing the worst-case loss function

$$G_i(\eta, m_0, m_1, s_1) = \max_{\substack{m_0 \le m \le m_1 \\ 1 \le s \le s_1}} \tilde{G}_i(\eta, m, s) \tag{4.9}$$

for $i = 1, 2, 3$ and selected values of m_0, m_1, and s_1. The bounds m_0, m_1, and s_1 define a range of number of points in the lattice, and maximal number of dimensions, in which we are interested. Selecting different parameters η for different ranges of values of m and s is a convenient compromise between the extreme cases of choosing a different η for each pair (m, s) and choosing the same η for all pairs (m, s). In practice one typically has a general idea of how many points one is going to take. By selecting an η specialized for that range, one can obtain a lattice with a better figure of merit for this particular range.

Table 4.1 gives the optimal η's and the corresponding figures of merit (4.9) for $i = 1, 2, 3$ and certain triples (m_0, m_1, s_1). Because of computational efficiency constraints, for the searches, we limited ourselves to $m_1 \le 20$ for $i = 1$ and to $s_1 \le 10$ for $i = 3$. Then, for the best η that we found, we verified the performance when m_0 was reduced or when m_1 or s_1 was increased and retained the smallest m_0 and largest m_1 and s_1 for which G_i was unchanged. The table also gives the value of $G_{i,\min}(\eta, m_0, m_1, s_1)$, defined by replacing max by min in (4.9). This best-case figure tells us the range of values taken by $\tilde{G}_i(\eta, m, s)$ over the given region. We made exhaustive searches also with $s_1 = 15$ for all the entries where $i = 1$ or 2 in the table, and we obtained the same values of η and $G_i(\eta, m_0, m_1, s_1)$ in all cases.

5. Estimating quadrature error. The advantage of an extensible lattice sequence is that N need not be specified in advance. Therefore, in practice one would estimate the quadrature error for an initial lattice and continue to increase the lattice size until the quadrature error meets the desired tolerance. Although the discrepancy, $D(P)$, is a good measure for the quality of a set P, it cannot be used directly for error estimation. The worst-case error bounds (3.1) are often quite conservative, and there is no easy way to estimate the variation of the integrand. The average case error given, (3.4), is sensitive to how one defines the kernel—multiplying the kernel by a factor of c^2 changes the average case error by a factor of c.

Quadrature error estimates for lattice rules have been investigated by [5, 29, 55]. Two different kinds of quadrature rules and error estimates have been proposed. Both involve estimating the error for $Q(f; P)$ in terms of $Q(f; P)$ and $Q(f; P_l)$, $l = 1, \ldots, M$, where the P_l are node sets of lattices and $P = P_1 \cup \cdots \cup P_M$.

The method proposed in [55] takes P to be composed of 2^s shifted copies of a rank-1 lattice. Rather than considering copy rules, we take P_l to be the node sets of shifted lattices of size b^{m_1} imbedded in the extensible lattice described in section 2.4:

$$P_l = \{\{\phi_b((l-1)b^{m_1}+i)h + \Delta\} : i = 0, \dots, b^{m_1}-1\}, \quad l = 1, \dots, M, \tag{5.1a}$$

$$P = \{\{\phi_b(i)h + \Delta\} : i = 0, \dots, Mb^{m_1}-1\}. \tag{5.1b}$$

Another quadrature rule and error estimate takes the P_l to be independent random shifts of a lattice. This can be done by taking

$$P_l = \{\{\phi_b(i)h + \Delta_l\} : i = 0, \dots, b^{m_1}-1\}, \quad l = 1, \dots, M, \tag{5.2a}$$

$$P = \{\{\phi_b(i)h + \Delta_l\} : i = 0, \dots, b^{m_1}-1,\ l = 1, \dots, M\}, \tag{5.2b}$$

where the Δ_l are independent, uniformly distributed random vectors in $[0,1]^s$.

Note that for both (5.1) and (5.2) the set P can be extended in size as necessary by increasing m_1. The theory behind the error estimates for cases (5.1) and (5.2) are given in the following theorem.

THEOREM 5.1. *Suppose that a quadrature rule $Q(\cdot;P)$ based on some arbitrary P, as given in* (1.1)*, is the average value of the quadrature rules based on the sets $P_1, \dots, P_M$, i.e.,*

$$Q(\cdot;P) = \frac{1}{M}\sum_{l=1}^{M} Q(\cdot;P_l). \tag{5.3}$$

First, consider the case where the integrands are random functions from a sample space $\mathcal{A}$ as described in the paragraph preceding (3.4)*. Then it follows that*

$$E_{f\in\mathcal{A}}[I(f)-Q(f;P)]^2 = \left\{-1 + \frac{1}{M}\sum_{l=1}^{M}\left[\frac{D(P_l)}{D(P)}\right]^2\right\}^{-1} \frac{1}{M}\sum_{l=1}^{M} E_{f\in\mathcal{A}}[Q(f;P_l)-Q(f;P)]^2, \tag{5.4}$$

where D is the discrepancy based on the covariance kernel for $\mathcal{A}$.

Second, consider the case of a fixed integrand, f, but where the P_l are random shifts of a set P_0, i.e.,

$$P = P_1 \cup \cdots \cup P_M, \qquad P_l = \{\{z + \Delta_l\} : z \in P_0\}, \tag{5.5}$$

for independent, uniformly distributed $\Delta_1, \dots, \Delta_M$. Then

$$E_{\Delta_l}\{[I(f)-Q(f;P)]^2\} = \frac{1}{M(M-1)}\sum_{l=1}^{M} E_{\Delta_l}[Q(f;P_l)-Q(f;P)]^2. \tag{5.6}$$

Proof. Assuming that the quadrature rules satisfy (5.3), it follows that the mean square deviation of the $Q(f;P_l)$ from $Q(f;P)$ may be written as

$$\begin{aligned}\frac{1}{M}\sum_{l=1}^{M}[Q(f;P_l)-Q(f;P)]^2 &= \frac{1}{M}\sum_{l=1}^{M}\{[I(f)-Q(f;P)]-[I(f)-Q(f;P_l)]\}^2 \\ &= -[I(f)-Q(f;P)]^2 + \frac{1}{M}\sum_{l=1}^{M}[I(f)-Q(f;P_l)]^2.\end{aligned} \tag{5.7}$$

If the integrand is a random function from a sample space $\mathcal{A}$ with covariance kernel K, then average case error analysis in (3.4) plus (5.7) leads to the following equations:

$$E_{f\in\mathcal{A}}[I(f)-Q(f;P)]^2=[D(P)]^2,$$
$$E_{f\in\mathcal{A}}[I(f)-Q(f;P_l)]^2=[D(P_l)]^2,$$
$$E_{f\in\mathcal{A}}\left\{\frac{1}{M}\sum_{l=1}^{M}[Q(f;P_l)-Q(f;P)]^2\right\}=-[D(P)]^2+\frac{1}{M}\sum_{l=1}^{M}[D(P_l)]^2.$$

The equations above may be rearranged to give (5.4).

The quadrature error for rule $Q(f;P)$ may also be written as

$$\begin{aligned}[I(f)-Q(f;P)]^2&=\left\{\frac{1}{M}\sum_{l=1}^{M}[I(f)-Q(f;P_l)]\right\}^2\\&=\frac{1}{M^2}\sum_{l=1}^{M}[I(f)-Q(f;P_l)]^2\\&\quad+\frac{2}{M^2}\sum_{1\le k<l\le M}[I(f)-Q(f;P_i)][I(f)-Q(f;P_l)].\end{aligned}$$

Substituting the sum of the $[I(f)-Q(f;P_l)]^2$ by (5.7) gives

$$\begin{aligned}(5.8)\quad[I(f)-Q(f;P)]^2&=\frac{1}{M(M-1)}\sum_{l=1}^{M}[Q(f;P_l)-Q(f;P)]^2\\&\quad+\frac{2}{M(M-1)}\sum_{1\le k<l\le M}[I(f)-Q(f;P_k)][I(f)-Q(f;P_l)].\end{aligned}$$

If the P_l are random shifts as in (5.5), then the expected value of the term $[I(f)-Q(f;P_k)][I(f)-Q(f;P_l)]$ vanishes for all $k\neq l$ and taking the expected value of (5.8) yields (5.6). □

Some remarks are in order to explain the assumptions and conclusions of the above theorem. These are given below.

Assumption (5.3), and thus conclusion (5.4), holds for both the cases (5.1) and (5.2) above. In fact, this part of the theorem holds for any imbedded rule or any rule where the P_l all contain the same number of points, and their union is P or multiple copies of P. For example, (5.4) would apply to the case where P is a (t,m,s)-net made up of a union of subnets P_l.

Assumption (5.5), and therefore conclusion (5.6), holds for (5.2). There is a difficulty if one tries to derive a result like (5.6) for an imbedded rule of the form (5.1), where Δ is a random shift. The argument leading to (5.6) assumes that the points in different P_l are uncorrelated, which is not true for (5.1). However, if the extensible lattice is a good one, it is expected that the terms $[I(f)-Q(f;P_k)][I(f)-Q(f;P_l)]$ in (5.8) are on average negative. Under this assumption one may then conclude that the right-hand side of (5.6) is a conservative (too large) upper bound on the expected square quadrature error.

The factor in (5.4) above involving the discrepancies of P and the P_l does not depend strongly on the particular choice of discrepancy but on the asymptotic rate

of decay only. If, for example, $D(P) \approx CN^{-\alpha}$ for some unknown C, but known α, where N is the number of points in P, then

$$E_{f\in\mathcal{A}}[I(f) - Q(f;P)]^2 \approx \frac{1}{M(M^{2\alpha}-1)} \sum_{l=1}^{M} E_{f\in\mathcal{A}}[Q(f;P_l) - Q(f;P)]^2. \tag{5.9}$$

Although conclusions (5.6) and (5.9) are derived under different assumptions, they both suggest error estimates of the form

$$[I(f) - Q(f;P)]^2 \lessapprox \frac{c^2}{M(M^{2\alpha}-1)} \sum_{l=1}^{M} [Q(f;P_l) - Q(f;P)]^2, \tag{5.10}$$

where $\alpha = 1/2$ for (5.6) and α indicates the rate of decay of the discrepancy for (5.9). The factor $c > 1$ depends on how conservative one wishes to be. The Chebyshev inequality implies that the above inequality will hold "at least" $100(1 - c^{-2})\%$ of the time. Error estimate (5.10) leads to the stopping criteria:

$$\frac{c^2}{M(M-1)} \sum_{l=1}^{M} [Q(f;P_l) - Q(f;P)]^2 < \epsilon^2, \tag{5.11}$$

where ϵ is the absolute error tolerance. Note that if the stopping criteria is not met, one would normally increase the *size* of the P_l by increasing m_1, rather than increasing the *number* of the P_l by increasing M.

For some high-dimensional problems the discrepancy of a lattice (or other low discrepancy set) may decay as slowly as the Monte Carlo rate of $\mathrm{O}(N^{-1/2})$ for small N (see [18, 42]). Therefore, even when using (5.9), it may be advisable to make a conservative choice of $\alpha = 1/2$. This choice makes the approach of error estimate (5.9), based on random integrands, equivalent to that of (5.6), based on randomly shifted P_l.

Sloan and Joe [55, Sect. 10.3] suggest an error estimate of the form

$$[I(f) - Q(f;P)]^2 \lessapprox \frac{c^2}{s} \sum_{l=1}^{s} [Q(f;P_l) - Q(f;P)]^2,$$

where P is formed from 2^s copies of a rank-1 lattice, and each $Q(f;P_l)$ is an imbedded rule based on half of the points in P. Although this case does not exactly fit Theorem 5.1, the arguments in the proof can be modified to obtain a result similar to (5.4):

$$E_{f\in\mathcal{A}}[I(f) - Q(f;P)]^2 = \left\{ -1 + \frac{1}{s} \sum_{l=1}^{s} \left[\frac{D(P_l)}{D(P)} \right]^2 \right\}^{-1} \frac{1}{s} \sum_{l=1}^{s} E_{f\in\mathcal{A}}[Q(f;P_l) - Q(f;P)]^2.$$

This would suggest that the error estimation formula of Sloan and Joe is reasonable when $D(P_l) \geq \sqrt{2}D(P)$ on average. The disadvantage of 2^s-copy rules is that they require at least 2^s points, which may be unmanageable for large s.

To summarize, both imbedded lattice rules, (5.1), and independent random shifts of lattices, (5.2), have similar error estimates, (5.10), and stopping criteria, (5.11). The advantage of the independent random shifts approach is that the theory holds

for any integrand, not the average over a space of integrands. One advantage of the imbedded rules approach is that one need generate only a single extensible lattice. Furthermore, the set P for the imbedded rule is the node set of a lattice (if M is a power of b), which is never the case for the independent random shifts approach. Thus, the accuracy of the imbedded rule approach is likely to be better. In the examples in the next section, the imbedded lattice rules based on (5.1) are used.

6. Examples of multidimensional quadrature. Two example problems are chosen to demonstrate the performance of the new rank-1 lattice sequences proposed in section 2.4. The first example is the computation of multivariate normal probabilities, and the second is the evaluation of a multidimensional integral arising in physics problems.

6.1. Multivariate normal probabilities. Consider the following multivariate normal probability:

$$\frac{1}{\sqrt{|\mathbf{\Sigma}|(2\pi)^s}} \int_{a_1}^{b_1} \cdots \int_{a_s}^{b_s} e^{-\frac{1}{2}\theta' \mathbf{\Sigma}^{-1}\theta} \, d\theta, \tag{6.1a}$$

where a and b are known s-dimensional vectors, and $\mathbf{\Sigma}$ is a given $s \times s$ positive definite covariance matrix. Some a_j and/or b_j may be infinite.

Unfortunately, the original form is not well suited for numerical quadrature. Therefore, Alan Genz [12] proposed a transformation of variables that results in an integral over an $(s-1)$-dimensional unit cube. See [12, 13] for the details of the transformation, and see [13, 15] for comparisons of different methods for calculating multivariate normal probabilities.

The particular test problem is one considered by [13, 24] and may be described as follows:

$$a_1 = \cdots = a_s = -\infty, \tag{6.1b}$$

$$b_j \text{ i.i.d. uniformly on } [0, \sqrt{s}], \tag{6.1c}$$

$$\mathbf{\Sigma} \text{ generated randomly according to [13, 39].} \tag{6.1d}$$

Numerical comparisons were made using three types of algorithms:

i. the adaptive algorithm DCHURE [2],
ii. an older Korobov rank-1 lattice rule with a different generating vector for each N and s—this algorithm is a part of NAG and is used in [5, 12]—and
iii. the new rank-1 lattice sequences proposed in section 2.4 with generating vectors given in Table 4.1.

For the second and third algorithms we applied the periodizing transformation $x'_j = |2x_j - 1|$ to the integrand over the unit cube. This appears to increase the accuracy of the lattice rule methods. The computations were carried out in FORTRAN on a Unix work station in double precision. The absolute error tolerance was chosen to be $\epsilon = 10^{-4}$, and this was compared with the actual error E. Since the true value of the integral is unknown for this test problem, the value given by the Korobov algorithm with a tolerance of $\epsilon = 10^{-8}$ was used as the "exact" value for computing the error. For the new rank-1 lattice sequences the stopping criterion (5.11) was used with M between 4 and 7 and $c = 3$.

For each dimension, 50 random test problems were generated and solved by the various quadrature methods. The scaled absolute errors E/ϵ and the computation times in seconds are given in the box and whisker plots of Figure 6.1. The boxes con-

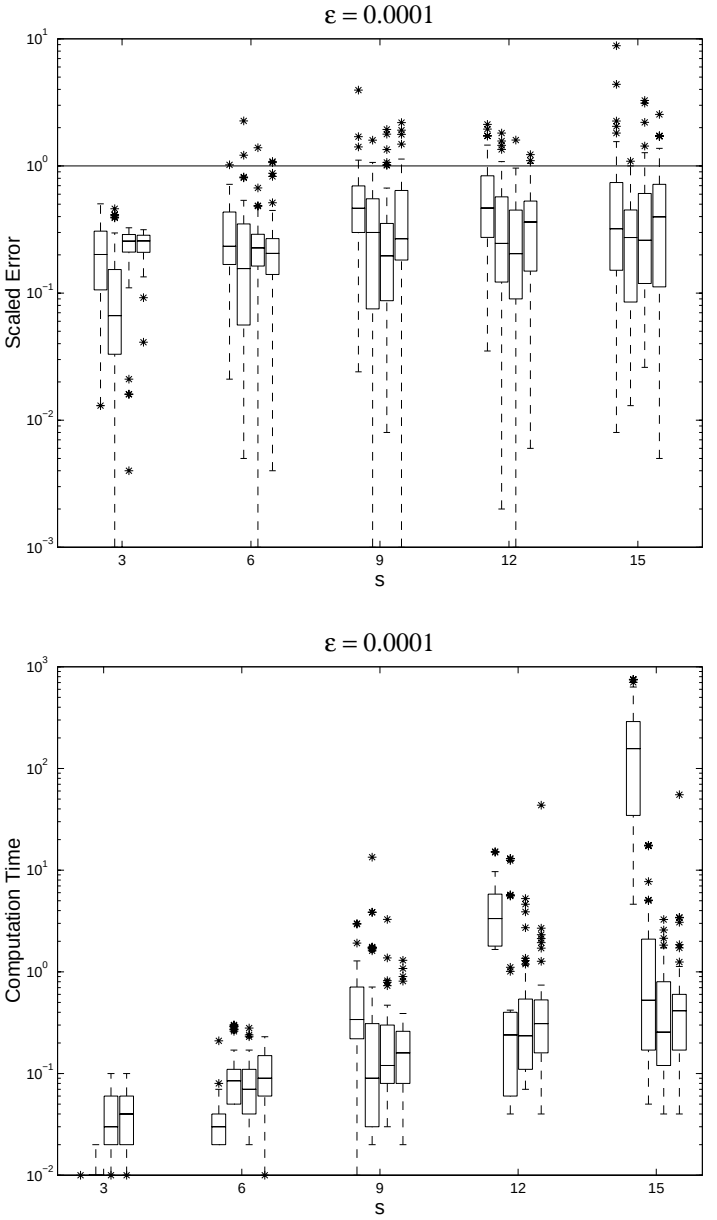


FIG. 6.1. *Box and whisker plots of scaled errors, E/ϵ, and the computation times in seconds for* 50 *randomly chosen test problems* (6.1). *For each dimension s results from left to right correspond to the DCHURE algorithm, Korobov rank-*1 *lattice rules used in NAG, and the new, extensible rank-*1 *lattice rules for* $\eta = 17797$ *and* 1267 *from Table* 4.1.

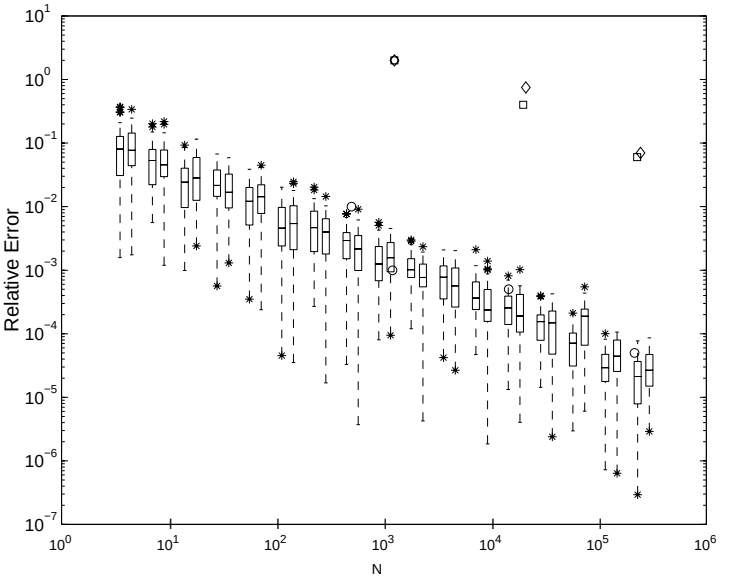


FIG. 6.2. *Plots of the relative error obtained in approximating* (6.2) *for* $s = 25$ *for the McNamee and Stenger* (□)*, Genz and Patterson* (◇)*, generalized Faure* (○) *quadrature rules. The box and whisker plots show the performance of the extensible lattice sequence rules for* $\eta = 17797$ *and* 1267 *from Table* 4.1 *for* $N = 4, 8, \ldots, 2^{17}$.

tain the middle half of the values, and the whiskers give the range of most values except the outliers (denoted by $*$).

Ideally, the scaled error should nearly always be less than one, otherwise the error estimate is not conservative enough. On the other hand, if the scaled error is too small, then the error estimate is too conservative. Figure 6.1 shows that the adaptive rule performs well for smaller dimensions but underestimates the error and is quite slow in higher dimensions. The lattice rules do well even in higher dimensions, and the new rank-1 lattice sequences appear to be faster than the older Korobov-type rule. This is likely due to the fact that the lattice sequences proposed here can reuse the old points when N must be increased.

6.2. A multidimensional integral from physics. Keister [30] considered the following multidimensional integral that has applications in physics:

$$\int_{\mathbf{R}^s} \cos(\|x\|_2) e^{-\|x\|_2^2}\, dx = \pi^{s/2} \int_{[0,1)^s} \cos\left(\sqrt{\sum_{j=1}^{s} \frac{[\Phi^{-1}(y_j)]^2}{2}}\right) dy, \tag{6.2}$$

where Φ denotes the standard Gaussian distribution function. Keister gave an exact formula for the answer and compared the quadrature methods of McNamee and Stenger [40], Genz [11], and Patterson [53] for evaluating this integral. Later, Papageorgiou and Traub [51] applied the generalized Faure sequence from FINDER to this problem.

The results of numerical experiments for the above integral for dimension 25 are shown in Figure 6.2. The exact value of the integral is reported in [51]. To be

consistent with the numerical results reported in [30, 51], we did not perform error estimation but just computed the actual error for each kind of numerical method as a function of N, the number of points. Because $\Phi^{-1}(0) = -\infty$, there is a technical difficulty with using an unshifted lattice rule, so when performing the numerical experiments, the lattice sequences were given random shifts (modulo 1). Box and whisker plots show how well the new rank-1 lattice sequences perform for 50 random shifts.

According to Figure 6.2, the generalized Faure sequence (in base 29) and the lattice sequence perform much better than the other two rules. In some cases the lattice sequences perform better than the generalized Faure sequence.

7. Conclusion. Lattice rules are simpler to code than digital nets. Given the construction in section 2.4, it is now possible to have extensible lattice sequences in the same way that one has (t, s)-sequences. Good generating vectors for these lattice sequences may be found by using the spectral test or minimizing other discrepancy measures, as shown in section 4. The performance of these lattice rules is in many cases comparable to other multidimensional quadrature rules and in some cases superior.

Acknowledgments. Thanks to Alan Genz for making his software for computing multivariate normal distributions available. Thanks also to Joe Traub and Anargyros Papageorgiou for making the FINDER software available.

REFERENCES

[1] M. Abramowitz and I. A. Stegun, eds., *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*, U.S. Government Printing Office, Washington, DC, 1964.

[2] J. Berntsen, T. O. Espelid, and A. Genz, *An adaptive algorithm for the approximate calculation of multiple integrals*, ACM Trans. Math. Software, 17 (1991), pp. 437–451.

[3] R. E. Caflisch, *Monte Carlo and quasi-Monte Carlo methods*, Acta Numer., 7 (1998), pp. 1–49.

[4] R. E. Caflisch, W. Morokoff, and A. Owen, *Valuation of mortgage backed securities using Brownian bridges to reduce effective dimension*, J. Comput. Finance, 1 (1997), pp. 27–46.

[5] R. Cranley and T. N. L. Patterson, *Randomization of number theoretic methods for multiple integration*, SIAM J. Numer. Anal., 13 (1976), pp. 904–914.

[6] U. Dieter, *How to calculate shortest vectors in a lattice*, Math. Comp., 29 (1975), pp. 827–833.

[7] K. Entacher, P. Hellekalek, and P. L'Écuyer, *Quasi-Monte Carlo node sets from linear congruential generators*, in Monte Carlo and Quasi-Monte Carlo Methods 1998, Springer-Verlag, Berlin, 1999, pp. 188–198.

[8] K. T. Fang and Y. Wang, *Number-Theoretic Methods in Statistics*, Chapman and Hall, London, 1994.

[9] H. Faure, *Discrépance de suites associées à un système de numération (en dimension s)*, Acta Arith., 41 (1982), pp. 337–351.

[10] U. Fincke and M. Pohst, *Improved methods for calculating vectors of short length in a lattice, including a complexity analysis*, Math. Comp., 44 (1985), pp. 463–471.

[11] A. C. Genz, *A Lagrange extrapolation algorithm for sequences of approximations to multiple integrals*, SIAM J. Sci. Statist. Comput., 3 (1982), pp. 160–172.

[12] A. Genz, *Numerical computation of multivariate normal probabilities*, J. Comput. Graph. Statist., 1 (1992), pp. 141–150.

[13] A. Genz, *Comparison of methods for the computation of multivariate normal probabilities*, Computing Science and Statistics, 25 (1993), pp. 400–405.

[14] S. Haber, *Parameters for integrating periodic functions of several variables*, Math. Comp., 41 (1983), pp. 115–129.

[15] V. Hajivassiliou, D. McFadden, and P. Ruud, *Simulation of multivariate normal rectangle probabilities and their derivatives: Theoretical and computational results*, J. Econometrics, 72 (1996), pp. 85–134.

[16] P. Hellekalek, *On the assessment of random and quasi-random point sets*, in Random and

Quasi-Random Point Sets, Lecture Notes in Statist. 138, Springer-Verlag, New York, 1998, pp. 49–108.

[17] P. Hellekalek and G. Larcher, eds., *Random and Quasi-Random Point Sets*, Lecture Notes in Statist. 138, Springer-Verlag, New York, 1998.

[18] F. J. Hickernell, *A comparison of random and quasirandom points for multidimensional quadrature*, in Monte Carlo and Quasi-Monte Carlo Methods in Scientific Computing, Lecture Notes in Statist. 106, Springer-Verlag, New York, 1995, pp. 213–227.

[19] F. J. Hickernell, *The mean square discrepancy of randomized nets*, ACM Trans. Model. Comput. Simul., 6 (1996), pp. 274–296.

[20] F. J. Hickernell, *Quadrature error bounds with applications to lattice rules*, SIAM J. Numer. Anal., 33 (1996), pp. 1995–2016; *Erratum*, 34 (1997), pp. 853–866.

[21] F. J. Hickernell, *A generalized discrepancy and quadrature error bound*, Math. Comp., 67 (1998), pp. 299–322.

[22] F. J. Hickernell, *Lattice rules: How well do they measure up?*, in Random and Quasi-Random Point Sets, Lecture Notes in Statist. 138, Springer-Verlag, New York, 1998, pp. 109–166.

[23] F. J. Hickernell, *What affects the accuracy of quasi-Monte Carlo quadrature?*, in Monte Carlo and Quasi-Monte Carlo Methods 1998, Springer-Verlag, Berlin, 1999, pp. 16–55.

[24] F. J. Hickernell and H. S. Hong, *Computing multivariate normal probabilities using rank-1 lattice sequences*, in Proceedings of the Workshop on Scientific Computing, Hong Kong, 1997, G. H. Golub, S. H. Lui, F. T. Luk, and R. J. Plemmons, eds., Springer-Verlag, Singapore, 1997, pp. 209–215.

[25] F. J. Hickernell and H. S. Hong, *The asymptotic efficiency of randomized nets for quadrature*, Math. Comp., 68 (1999), pp. 767–791.

[26] F. J. Hickernell and R. X. Yue, *The mean square discrepancy of scrambled (t, s)-sequences*, SIAM J. Numer. Anal., to appear.

[27] E. Hlawka, *Funktionen von beschränkter Variation in der Theorie der Gleichverteilung*, Ann. Mat. Pura Appl., 54 (1961), pp. 325–333.

[28] L. K. Hua and Y. Wang, *Applications of Number Theory to Numerical Analysis*, Springer-Verlag and Science Press, Berlin-New York, and Beijing, 1981.

[29] S. Joe, *Randomization of lattice rules for numerical multiple integration*, J. Comput. Appl. Math., 31 (1990), pp. 299–304.

[30] B. D. Keister, *Multidimensional quadrature algorithms*, Computers in Physics, 10 (1996), pp. 119–122.

[31] D. E. Knuth, *The Art of Computer Programming. Vol. 2. Seminumerical Algorithms*, 3rd ed., Addison-Wesley, Reading, MA, 1997.

[32] N. M. Korobov, *The approximate computation of multiple integrals*, Dokl. Akad. Nauk SSR, 124 (1959), pp. 1207–1210 (in Russian).

[33] G. Larcher, *On the distribution of digital sequences*, in Monte Carlo and Quasi-Monte Carlo Methods 1996, H. Niederreiter, P. Hellekalek, G. Larcher, and P. Zinterhof, eds., Lecture Notes in Statist. 127, Springer-Verlag, New York, 1998, pp. 109–123.

[34] P. L'Écuyer, *Random number generation*, in Handbook of Simulation, J. Banks, ed., Wiley, 1998, New York, pp. 93–137.

[35] P. L'Écuyer, *Tables of linear congruential generators of different sizes and good lattice structure*, Math. Comp., 68 (1999), pp. 249–260.

[36] C. Lemieux and P. L'Écuyer, *Efficiency improvement by lattice rules for pricing Asian options*, in Proceedings of the 1998 Winter Simulation Conference, Washington, DC, IEEE Press, Piscataway, NJ, 1998, pp. 579–586.

[37] C. Lemieux and P. L'Écuyer, *A comparison of Monte Carlo, lattice rules and other low-discrepancy point sets*, in Monte Carlo and Quasi-Monte Carlo Methods 1998, Springer-Verlag, Berlin, 1999, pp. 326–340.

[38] G. Marsaglia, *Random numbers fall mainly in the planes*, Proc. Natl. Acad. Sci. USA, 60 (1968), pp. 25–28.

[39] G. Marsaglia and I. Olkin, *Generating correlation matrices*, SIAM J. Sci. Statist. Comput., 5 (1984), pp. 470–475.

[40] J. McNamee and F. Stenger, *Construction of fully symmetric numerical integration formulas*, Numer. Math., 10 (1967), pp. 327–344.

[41] W. J. Morokoff, *Generating quasi-random paths for stochastic processes*, SIAM Rev., 40 (1998), pp. 765–788.

[42] W. J. Morokoff and R. E. Caflisch, *Quasi-random sequences and their discrepancies*, SIAM J. Sci. Comput., 15 (1994), pp. 1251–1279.

[43] W. J. Morokoff and R. E. Caflisch, *Quasi-Monte Carlo integration*, J. Comput. Phys., 122 (1995), pp. 218–230.

[44] H. Niederreiter, *Point sets and sequences with small discrepancy*, Monatsh. Math., 104 (1987), pp. 273–337.
[45] H. Niederreiter, *Random Number Generation and Quasi-Monte Carlo Methods*, SIAM, Philadelphia, 1992.
[46] H. Niederreiter and P. J.-S. Shiue, eds., *Monte Carlo and Quasi-Monte Carlo Methods in Scientific Computing*, Lecture Notes in Statist. 106, Springer-Verlag, New York, 1995.
[47] H. Niederreiter and J. Spanier, eds., *Monte Carlo and Quasi-Monte Carlo Methods* 1998, Springer-Verlag, Berlin, 1999.
[48] H. Niederreiter and C. Xing, *Quasirandom points and global function fields*, in Finite Fields and Applications, London Math. Soc. Lecture Note Ser. 233, S. Cohen and H. Niederreiter, eds., Cambridge University Press, Cambridge, UK, 1996, pp. 269–296.
[49] A. Papageorgiou, *private communication*, 1997.
[50] A. Papageorgiou and J. F. Traub, *Beating Monte Carlo*, Risk, 9 (1996), pp. 63–65.
[51] A. Papageorgiou and J. F. Traub, *Faster evaluation of multidimensional integrals*, Computers in Physics, 11 (1997), pp. 574–578.
[52] S. Paskov and J. Traub, *Faster valuation of financial derivatives*, J. Portfolio Management, 22 (1995), pp. 113–120.
[53] T. N. L. Patterson, *The optimum addition of points to quadrature formulae*, Math. Comp., 22 (1968), pp. 847–856.
[54] I. H. Sloan, *Lattice methods for multiple integration*, J. Comput. Appl. Math., 12/13 (1985), pp. 131–143.
[55] I. H. Sloan and S. Joe, *Lattice Methods for Multiple Integration*, The Claredon Press, Oxford University Press, New York, 1994.
[56] I. H. Sloan and P. J. Kachoyan, *Lattice methods for multiple integration: Theory, error analysis and examples*, SIAM J. Numer. Anal., 24 (1987), pp. 116–128.
[57] I. H. Sloan and H. Woźniakowski, *An intractability result for multiple integration*, Math. Comp., 66 (1997), pp. 1119–1124.
[58] I. M. Sobol', *Distribution of points in a cube and the approximate evaluation of integrals*, Comput. Math. Math. Phys., 7 (1967), pp. 86–112.
[59] I. M. Sobol', *Multidimensional Quadrature Formulas and Haar Functions*, Izdat. "Nauka," Moscow, 1969 (in Russian).
[60] J. Spanier, *Quasi-Monte Carlo methods for particle transport problems*, in Monte Carlo and Quasi-Monte Carlo Methods in Scientific Computing, Lecture Notes in Statist. 106, Springer-Verlag, New York, 1995, pp. 121–148.
[61] J. Spanier and E. H. Maize, *Quasi-random methods for estimating integrals using relatively small samples*, SIAM Rev., 36 (1994), pp. 18–44.
[62] T. T. Warnock, *Computational investigations of low discrepancy point sets*, in Applications of Number Theory to Numerical Analysis, S. K. Zaremba, ed., Academic Press, New York, 1972, pp. 319–343.
[63] H. Woźniakowski, *Average case complexity of multivariate integration*, Bull. Amer. Math. Soc., 24 (1991), pp. 185–194.
[64] S. K. Zaremba, *Some applications of multidimensional integration by parts*, Ann. Polon. Math., 21 (1968), pp. 85–96.

SIAM J. SCI. COMPUT.
Vol. 22, No. 4, pp. 1139–1159
© 2000 Society for Industrial and Applied Mathematics

APPROXIMATE PROJECTION METHODS: PART I. INVISCID ANALYSIS*

ANN S. ALMGREN†, JOHN B. BELL†, AND WILLIAM Y. CRUTCHFIELD†

Abstract. The use of approximate projection methods for modeling low Mach number flows avoids many of the numerical complications associated with exact projection methods, but introduces additional design choices in developing a robust algorithm. In this paper we first explore these design choices in the setting of inviscid incompressible flow using several computational examples. We then develop a framework for analyzing the behavior of the different design variations and use that analysis to explain the features observed in the computations. As part of this work we introduce a new variation of the approximate projection algorithm that combines the advantages of several of the previous versions.

Key words. projection method, approximate projection, incompressible flow

AMS subject classifications. 76C99, 65M06, 65C20, 76M20

PII. S1064827599357024

1. Introduction.

The term "projection methods" refers to a class of fractional step algorithms for modeling unsteady low Mach number flows characterized by a divergence constraint on the velocity field. In these methods an intermediate velocity field is constructed by advancing the momentum equations in time without enforcing the constraint. In the next step of the algorithm this field is projected back onto the constraint.

Projection methods have been used for incompressible flow modeling for several decades, but approximate projection methods have appeared only in the past few years. They have been numerically tested on a number of flows in two and three spatial dimensions, but adequate analysis of their properties has been lacking to date. In this paper we provide an analytical basis for understanding the behavior of approximate projection methods and some of the design decisions that accompany their use.

For the purposes of this paper we will focus on inviscid, constant density flows, with numerical examples in two dimensions. In the second paper in this series we will extend the analysis to the modeling of viscous flows. The extension to variable density flows, and adaptive projection methods with subcycling in time, will follow.

In the original projection method developed by Chorin [5] the projection step of the algorithm is specified by defining discrete operators $\mathbf{D}$ and $\mathbf{G}$, approximating divergence and gradient, respectively, which are skew-adjoint; i.e., $\mathbf{D} = -\mathbf{G}^T$. With this definition the discrete projection, $\mathbf{P} = \mathbf{I} - \mathbf{G}(\mathbf{D}\mathbf{G})^{-1}\mathbf{D}$ (with boundary conditions implicitly defined by the flow problem), is a discrete orthogonal projection on the finite-dimensional space of vector fields defined on the mesh. In Chorin's formulation both pressure and velocity are specified at nodes and central differences are used for the definition of $\mathbf{D}$ and $\mathbf{G}$. This results in an expanded five-point stencil for the

*Received by the editors June 14, 1999; accepted for publication (in revised form) April 18, 2000; published electronically October 18, 2000. This work was supported by the Applied Mathematical Sciences Program of the DOE Office of Mathematics, Information, and Computational Sciences, under contract DE-AC03-76SF00098. The U.S. Government retains a nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or allow others to do so, for U.S. Government purposes. Copyright is owned by SIAM to the extent not limited by these rights.
http://www.siam.org/journals/sisc/22-4/35702.html

†MS 50A-1148, Lawrence Berkeley National Laboratory, Berkeley, CA 94720 (ASAlmgren@lbl.gov, JBBell@lbl.gov, WYCrutchfield@lbl.gov).

discrete Laplacian, $L = \mathbf{DG}$, which must be inverted to apply the projection. This expanded stencil produces a local decoupling of the mesh points and leads to a 2^d-dimensional kernel for $\mathbf{G}$ where d is the dimension of the problem. Bell, Colella, and Glaz [3] use a discretization of the projection based on a finite element method due to Fortin [7] which uses pressure defined on cell centers with velocities given at nodes. This approach produces a more compact stencil but also generates a local decoupling of the grid and dim ker $\mathbf{G} > 1$. (For two-dimensional problems dim ker $\mathbf{G} = 2$.) Bell, Colella, and Howell [4] use a fully cell-centered analogue of Chorin's algorithm. This scheme exhibits a local decoupling; however, in the presence of Dirichlet boundary conditions, the cell-centered approximation eliminates the nonconstant elements in ker $\mathbf{G}$. (For periodic problems dim ker $\mathbf{G} = 2^d$ as in the original Chorin algorithm.)

In the absence of boundaries the additional elements in ker $\mathbf{G}$ are not a problem for incompressible flow modeling because the operator $(\mathbf{DG})^{-1}$ is applied to a vector field that lies in the range of $\mathbf{D}$. In the presence of boundaries, however, the nonconstant elements in the kernel introduce additional, artificial compatibility constraints on the boundary of the physical domain. In addition, for more complex low Mach number models, such as low Mach number combustion, a term that is not in the range of the divergence operator is often added to the divergence constraint to represent the expansion or contraction of the fluid due to additional physics. The presence of this additional source term can result in marked oscillations in the solution, as noted by Lai [10] and Lai, Bell, and Colella [11]. An additional difficulty associated with the local decoupling is that the nonstandard discretizations of $\mathbf{DG}$ require specialized solution methods that properly respect the stencil that is used. (See [4, 8] for a discussion of such a procedure.)

Almgren, Bell, and Szymczak [2] first introduced the notion of an approximate projection to circumvent the numerical difficulties with exact discrete projections. Approximate projections are defined by replacing the projection operator $\mathbf{P}$ by an approximation $\widetilde{\mathbf{P}} = \mathbf{I} - \mathbf{G}(L)^{-1}\mathbf{D}$, where L is an approximation to, but not identically, $\mathbf{DG}$. The approximate projection methods discussed here were designed to operate on cell-centered colocated velocities, to avoid any local decoupling of the stencils, to provide a symmetric discretization of the potential flow inherent in inhomogeneous boundary conditions, and to generate a linear system that fits the framework of conventional fast iterative techniques (e.g., multigrid) for second-order elliptic equations. (See [2] for further discussion of the motivation for approximate projections.)

The approximate projection introduced by Almgren, Bell, and Szymczak [2] defines pressure on nodes and uses a finite element formulation with bilinear basis functions to derive a nine-point (in two dimensions) stencil for the Laplacian operator. The finite element derivation gives a characterization of this projection as an exact discrete projection onto an enriched space followed by an L^2 projection onto a subspace corresponding to piecewise constant velocity fields. This derivation provides an analytic characterization of the effect of the "approximateness" and proves stability of the approximation. A standard five-point nodal approximation to the Laplacian can also be used. This approximation can be constructed in a similar manner by specifying pressure to be piecewise linear on the standard triangularization of a square mesh. The analogous three-dimensional seven-point nodal discretization has been successfully used by Almgren et al. in [1], but lacks the finite element derivation.

Lai [10] introduced a cell-centered approximate projection where pressure is defined at cell centers and which uses a standard five-point centered-difference stencil for the Laplacian. The stability of such a scheme was shown in [10]; this approximate

projection has been used by Martin [12] in an adaptive projection algorithm, and a variation of this (with the same L but a higher-order $\mathbf{D}$) by Minion [14] in an adaptive scheme.

Approximate projections have been successfully used in the modeling of more general low Mach number flows such as those arising in combustion (e.g., [17, 18]). However, in spite of these successful applications, approximate projection algorithms are not without their own problems. Adopting an approximate projection approach introduces a number of design choices that are not relevant for an exact discrete projection. When applied to "difficult" problems or to long-time integrations, approximate projection algorithms can produce poor results. Rider discusses some of these design choices for the five-point cell-centered scheme [19] and describes a collection of filtering techniques to deal with artifacts he observes in numerical tests [20].

The goal of this paper is to develop an analytical framework to explain the performance of different variations of approximate projection methods and to explore methods for improving their behavior. In the next section we will give a brief overview of projection methods using an exact projection; in section 3 we introduce several different formulations of approximate projection schemes. In the following two sections we present a temporal analysis of approximate projections, then a spatial analysis of the three specific approximate projections discussed here. In section 6 we discuss issues related to the coupling between the advection scheme and the choice of approximate projection, and in section 7 we present our conclusions and discussion.

2. Overview of exact projection methods. In this paper we will consider constant density, inviscid, incompressible flow, governed by

$$\mathbf{U}_t + (\mathbf{U}\cdot\nabla)\mathbf{U} + \nabla p = 0, \tag{2.1}$$

$$\nabla\cdot\mathbf{U} = 0, \tag{2.2}$$

where (in two dimensions) $\mathbf{U} = (u, v)$ and p represent the velocity and pressure, respectively.

The projection method is a fractional step scheme for solving (2.1)–(2.2). In the advection step we solve (2.1) without enforcing the constraint. In the projection step, we then apply a projection to the intermediate velocity field to enforce the constraint. For the advection step we solve

$$\frac{\mathbf{U}^{*,n+1} - \mathbf{U}^n}{\Delta t} + [(\mathbf{U}\cdot\nabla)\mathbf{U}]^{n+1/2} = -\mathbf{G}p^{n-1/2} \tag{2.3}$$

for the intermediate velocity $\mathbf{U}^{*,n+1}$. The pressure gradient is evaluated at $t^{n-1/2}$ and is treated as a source term. For the computations presented here, the advection term, $\mathbf{N}^{n+1/2} = [(\mathbf{U}\cdot\nabla)\mathbf{U}]^{n+1/2}$, is approximated at time $t^{n+1/2}$ to second-order in space and time using an explicit predictor-corrector scheme described in Appendix A. (We note that the analysis presented later in the paper is applicable when other approaches are used to compute $\mathbf{N}^{n+1/2}$. Experiments with different options in the predictor-corrector scheme presented in section 6 are suggestive of the degree to which the conclusions reached here are applicable to other methods.)

The velocity field $\mathbf{U}^{*,n+1}$ is not, in general, divergence-free. In the projection step of the algorithm the intermediate velocity field can be decomposed into a discrete gradient of a scalar potential and a discretely divergence-free vector field which

correspond, respectively, to the new pressure gradient and the new velocity, i.e.,

$$\mathbf{U}^{n+1} = \mathbf{P}\mathbf{U}^{*,n+1},$$
$$\Delta t\, \mathbf{G}p^{n+1/2} = \Delta t\, \mathbf{G}p^{n-1/2} + (\mathbf{I} - \mathbf{P})\mathbf{U}^{*,n+1}.$$

To apply the projection operator we solve $\mathbf{DG}\phi = \mathbf{DV}$ for a scalar field ϕ, where, in this case, $\mathbf{V} = \mathbf{U}^{*,n+1}$. We then define the divergence-free part of $\mathbf{V}$, in this case $\mathbf{U}^{n+1}$, by $\mathbf{V}_d = \mathbf{V} - \mathbf{G}\phi$. By construction $\mathbf{DV}_d = 0$; thus, $\mathbf{U}^{n+1}$ exactly satisfies the discrete form of divergence constraint.

3. Approximate projections. For an exact projection, $\mathbf{V}$ can take many forms, all of which lead to exactly the same solution (assuming an exact solution of the resulting Poisson equation). Two natural candidates for $\mathbf{V}$ are the velocity itself or the update to velocity. One could also modify each of these by removing the pressure gradient component of the update. Specifically, one might choose any of the following (with the Δt scaling for convenience):

$$(1) \quad \mathbf{V} = \frac{\mathbf{U}^{*,n+1}}{\Delta t},$$
$$(2) \quad \mathbf{V} = \frac{\mathbf{U}^{*,n+1}}{\Delta t} + \mathbf{G}p^{n-1/2},$$
$$(3) \quad \mathbf{V} = \frac{\mathbf{U}^{*,n+1} - \mathbf{U}^{n}}{\Delta t},$$
$$(4) \quad \mathbf{V} = \frac{\mathbf{U}^{*,n+1} - \mathbf{U}^{n}}{\Delta t} + \mathbf{G}p^{n-1/2}.$$

Then, after solving $\mathbf{DG}\phi = \mathbf{DV}$ (or $L\phi = \mathbf{DV}$ for an approximate projection) and setting $\mathbf{V}_d = \mathbf{V} - \mathbf{G}\phi$, the new velocity and pressure would be defined, respectively, by

$$(1) \quad \mathbf{U}^{n+1} = \Delta t\, \mathbf{V}_d, \qquad p^{n+1/2} = p^{n-1/2} + \phi,$$
$$(2) \quad \mathbf{U}^{n+1} = \Delta t\, \mathbf{V}_d, \qquad p^{n+1/2} = \phi,$$
$$(3) \quad \mathbf{U}^{n+1} = \mathbf{U}^{n} + \Delta t\, \mathbf{V}_d, \qquad p^{n+1/2} = p^{n-1/2} + \phi,$$
$$(4) \quad \mathbf{U}^{n+1} = \mathbf{U}^{n} + \Delta t\, \mathbf{V}_d, \qquad p^{n+1/2} = \phi.$$

For approximate projections the choice of $\mathbf{V}$ has a nontrivial effect on the solution. Because the approximate projection operators are second-order accurate approximations to an exact projection, the methods which result from each choice of $\mathbf{V}$ are all second-order accurate for smooth problems, but not identical.

For the sake of clarity, we here introduce some notation for the different spatial and temporal discretizations. We will refer to the approximate projection with the nine-point nodal discretization of L as **N9**, that with the five-point nodal discretization of L as **N5**, and that with the five-point cell-centered discretization of L as **C5**. In addition, for all results we will specify which version of $\mathbf{V}$ is being projected; e.g., **N9_3** will indicate the nine-point nodal scheme, version (3) of $\mathbf{V}$. In the published literature, Almgren et al. [1, 2] have presented results using **N5_3** and **N9_3**; Martin [12] used **C5_2**, and Minion [14] has used **C5_4**. Rider [19] presents extensive testing of **C5_1-4** in conjunction with the development of filters to reduce numerical artifacts.

We also note here that for each of the approximate projection operators $\widetilde{\mathbf{P}}$, there is a corresponding exact discrete projection $\mathbf{P} = \mathbf{I} - \mathbf{G}(\mathbf{DG})^{-1}\mathbf{D}$ defined using the

same **D** and **G** used to define $\widetilde{\mathbf{P}}$. We will refer to the exact nodal projection as **N_EX** and the exact cell-centered projection as **C_EX.**

To demonstrate the second-order accuracy of the different versions of the approximate projections, we present results from a time-periodic test case with diagonally translating vortices. The initial data are

$$u(x,y) = 1 - 2\cos(x)\sin(y),$$
$$v(x,y) = 1 + 2\sin(x)\cos(y)$$

on a square domain, 2π x 2π. The exact solution to the Euler equations with this initial data is

$$u_{ex}(x,y,t) = 1 - 2\cos(x-t)\sin(y-t),$$
$$v_{ex}(x,y,t) = 1 + 2\sin(x-t)\cos(y-t),$$
$$p_{ex}(x,y,t) = -\cos(2(x-t)) - \sin(2(y-t)).$$

For each version of **V** and each approximate projection we present results at three different resolutions: 32^2, 64^2, and 128^2. The L^2 norms of the error in u and p_x are shown in Figure 1. It is clear that each method is second-order accurate; in addition, it is evident that there are differences between the solutions resulting from the different forms of the projections, most noticeably **C5_4**. We will explain these differences in the next two sections.

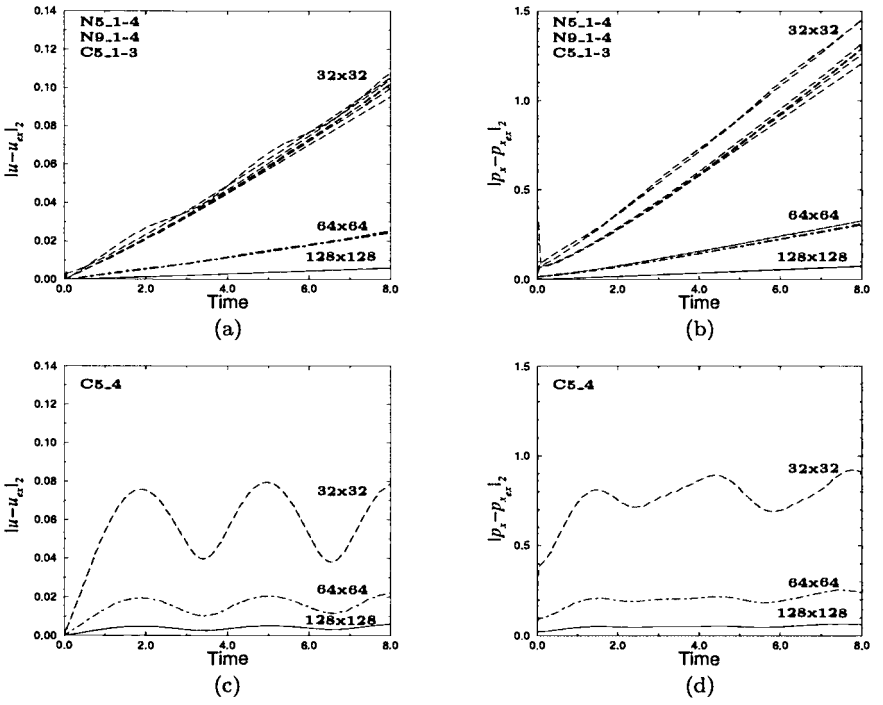


FIG. 1. L^2 *norms of error in* u (a,c) *and* p_x (b,d) *for the translating vortices problem.*

As demonstrated above, the difficulty with using approximate projections is not a question of formal accuracy on smooth problems. Instead, problems tend to appear

as a buildup of "noise" during longer time integrations of more complex problems. To illustrate this behavior we present results from each of the approximate projections run on on a more complex problem, namely the inviscid dynamics of a random initial distribution of vorticity. A key feature of this problem is that it contains substantially more high frequency content than the previous test case. The initial data for this problem are given by specifying a random stream function with spectral characteristics defined by

$$\hat{\psi}(k) = \frac{\omega}{|k|(1 + (|k|/6)^4)},$$

where ω is a normally distributed random variable mapped into the complex domain with the appropriate symmetries so that the inverse Fourier transform of $\hat{\psi}$ is a real function, ψ. We then define $\mathbf{U}^0$ as the discrete curl of ψ. The initial data for this problem are similar to the initial data used to test the decay at high Reynolds number of the two-dimensional Navier–Stokes equations to the sinh-Poisson mean field equations derived by Joyce and Montgomery [9]. This type of problem has been studied extensively by a number of authors. (See the early work by McWilliams [13] and other studies such as Montgomery, Shan, and Matthaeus [15] and the references cited therein.) We note that the key difference between the initial data we have constructed and the data used in the above studies is that in those studies the kinetic energy scales like k^{-3} for large k, but here we are imposing a more rapid decay for large k. This more rapid decay reduces the very high frequency components of the solution so that the computations are less sensitive to the behavior of the advection scheme. (Results similar to those presented here are also obtained with the richer spectrum but the anomalies occur at higher resolution and longer time integrations, which are prohibitive for extensive testing.)

We examine the evolution of the solution from time $t = 0$ to 10, over which time the initial random vorticity essentially coalesces into two smooth patches of counter-rotating vorticity. In Figure 2(a) and 2(e) we show raster plots of the vorticity and p_x at $t = 10$ on a 64^2 grid calculated with **N9_2.** The domain is the unit square with doubly periodic boundary conditions, and all calculations are run with CFL = 0.9 (i.e., $\Delta t = 0.9\ h\ /\ \max(\max_{(i,j)}|u|, \max_{(i,j)}|v|)$, where $\max_{(i,j)}|\cdot|$ is the maximum over all grid cells, and h is the mesh spacing).

Several variations of the approximate projection produce poor results for this problem. In particular, in Figure 2(b)–(d), (f)–(h) we show the vorticity and p_x at $t = 10$ as calculated using **N9_1**, **C5_3**, and **N5_4**. For the **N9_1** case, checkerboarding (high frequency error in the solution) is evident in p_x (Figure 2(f)) and to a lesser extent in the vorticity (Figure 2(b)). Similarly, for **C5_3**, high frequency error is visible, though due to the cell-centered discretization the error appears more as "striping" than checkerboarding in the vorticity (Figure 2(c)). More coherently structured but equally unphysical features are evident in both the vorticity (Figure 2(d)) and pressure gradient (Figure 2(h)) for **N5_4**.

In Figure 3 we plot the numerical divergence of velocity (calculated in each case with the same **D** as the approximate projection used) as a function of time for all 12 cases (**C5_1-4, N9_1-4, N5_1-4**), at resolutions 64^2 and 128^2. While the divergence does decrease with increased resolution, the differences between the variations of the projection persist. (In fact **N5_1** becomes more unstable at higher resolution.) An examination of all 12 cases indicates that for solutions with relatively small divergence, the plots of vorticity and pressure gradient are similar to Figure 2(a) and Figure 2(e), respectively. Similarly, for solutions with relatively larger divergence, the plots of

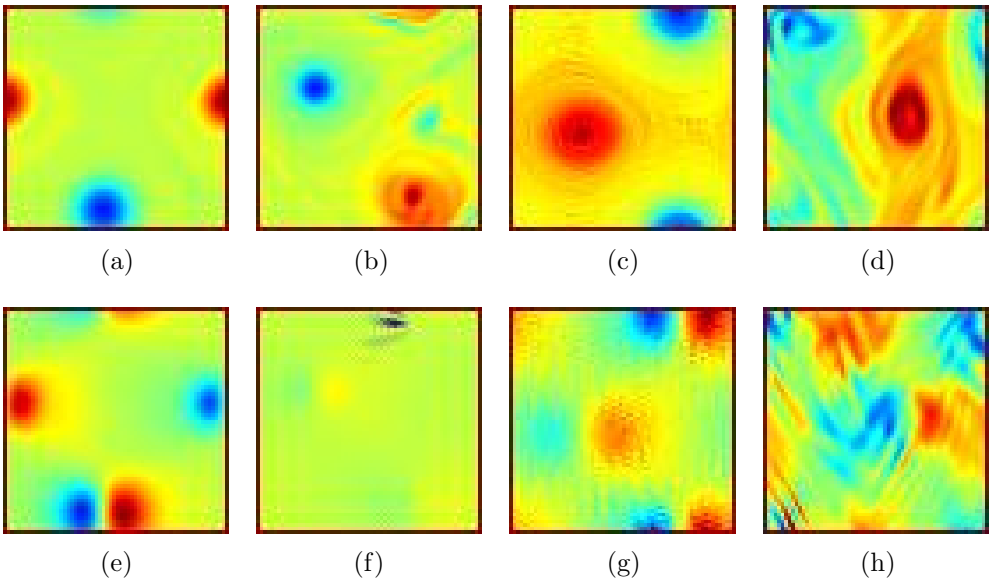


FIG. 2. *Vorticity* (a)–(d) *and* p_x (e)–(h) *at* $t = 10$ *for the random initial vorticity calculation at resolution* 64^2. (a), (e) **N9_2**; (b), (f) **N9_1**; (c), (g) **C5_3**; (d), (h) **N5_4**.

vorticity and pressure gradient more closely resemble Figure 2(b)–(d) and Figure 2(f)–(h), respectively. It is evident that the divergence is a useful indicator of solution quality for problems such as this one for which an exact solution is not known.

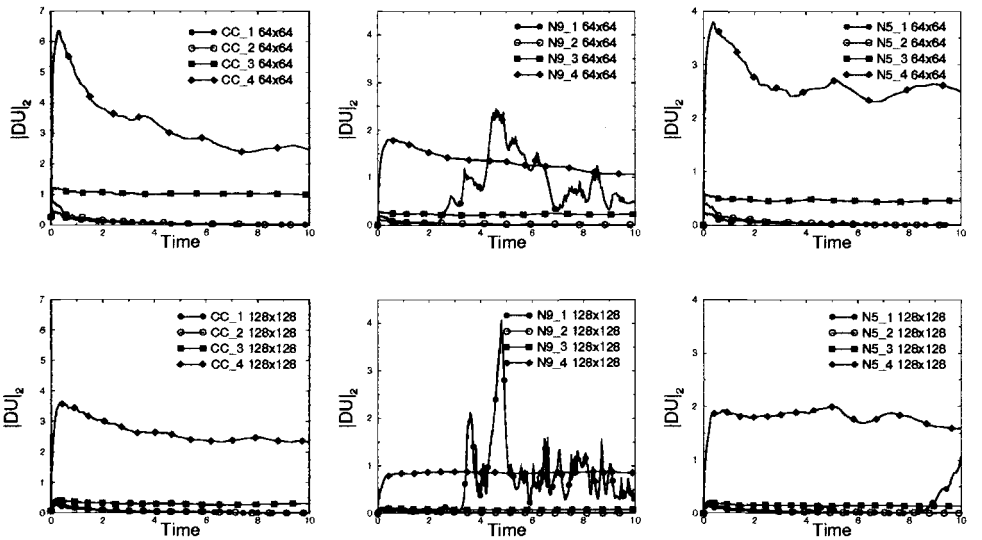


FIG. 3. L^2 *norms of* **DU** *from* $t = 0$ *to* 10 *for the random initial vorticity problem at two different resolutions.*

For the random initial vorticity problem, it appears that the only acceptable formulation of the approximate projection across all three discretizations is version (2). Version (2), however, is not without its drawbacks. As we will see analytically in

the next section, the divergence of the solution found with version (2) is typically larger than when version (1) is used, provided version (1) remains stable. To demonstrate this, we consider a very simple steady problem for which we know the exact solution. The initial data (and steady solution) are $u_{ex} = \sin(y)$, $v_{ex} = \cos(x)$.

To eliminate issues of predictor-projection coupling which will be discussed in section 7, here we use the exact analytic solution to define the advective terms at each time step, i.e., we set $\mathbf{N}_{i,j}^{n+1/2} = (\cos(x_i)\cos(y_j), -\sin(x_i)sin(y_j))$, where $x_i = ih, y_j = jh$ with h the mesh spacing. All calculations are run with CFL = 0.9 and resolution 32^2.

Figure 4 shows the L^2 error in u from time $t = 0$ to 150 for versions (1) and (2) with each approximate projection. The solid curves using version (1) are indistinguishable in this figure from the x-axis; the three remaining curves were computed with version (2). Clearly the error grows steadily for version (2), while remaining essentially zero for version (1).

While the difference between versions (1) and (2) is especially noticeable for a steady problem, analogous behavior is seen in the initial iteration of the algorithm which is used to define the initial pressure field. At the beginning of a calculation, the initial velocity field is projected once to enforce the divergence constraint. Then an iteration is performed to compute $\mathbf{G}p^{-1/2}$. In each step of this iteration a full time step is taken; after the projection step the pressure field is updated, but the velocity is reset to its previous value.

If version (1) or (3) of the approximate projection is used, then in the limit as the number of iterations becomes infinite, the initial pressure converges to the correct value (that which would be found with infinitely many iterations of an exact projection). With versions (2) or (4), however, because these versions are nonincremental in pressure, the pressure does not converge, regardless of the number of iterations.

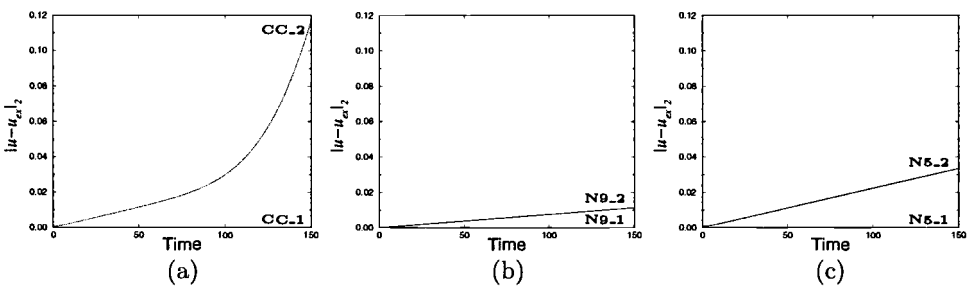


FIG. 4. *L^2 norms of the error in u from $t = 0$ to $t = 150$ from calculations of the steady problem for versions* (a) **CC_1-2**, (b) **N9_1-2**, (c) **N5_1-2**. *Curves using projection version* (1) *overlay the x-axis.*

4. Temporal analysis of approximate projections. The goal of this and the next section is to develop an analytical framework to explain the results of the computations presented in the previous section. The expressions presented by Rider [19] for the divergence of velocity suggest that version (1) should produce the lowest divergence, that versions (2) and (3) should produce similar results, and that version (4) should generate the highest divergence. These predictions hold true in some cases but do not fully explain the computations presented in the previous section or the computations in [19], specifically the lack of robustness of version (1) and the differences in behavior between versions (2) and (3).

In order to understand the behavior of the different variations of approximate projections, in Tables 1 and 2 we express the velocity and pressure gradient after n time steps, $\mathbf{U}^n$ and $\mathbf{G}p^{n-1/2}$, in terms of the initial data, $\mathbf{U}^0$ and $\mathbf{G}p^{-1/2}$, the advective updates at each time step, $\mathbf{N}^{i+1/2}$, and the approximate projection operator, $\widetilde{\mathbf{P}}$. These expressions are derived by substituting the expression for $\mathbf{U}^{*,n+1}$ from (2.3) into the definitions of $\mathbf{U}^{n+1}$ for each version of the approximate projection defined in section 3. Successive substitution provides the desired characterization of each version of the algorithm.

TABLE 1
$\mathbf{U}^n$ *in terms of* $\mathbf{U}^0$, $\mathbf{G}p^{-1/2}$, *and* $\mathbf{N}$. *All sums are from* $i = 1$ *to* n *unless otherwise noted.*

Version	$\mathbf{U}^n =$
Exact	$\mathbf{U}^0 - \sum \mathbf{P}(\Delta t\, \mathbf{N}^{n+1/2-i})$
(1)	$\mathbf{U}^n = \mathbf{Q}_\rho^n \mathbf{U}^0 - \mathbf{Q}_\beta^n(\Delta t\, \mathbf{G}p^{-1/2}) - \sum \mathbf{Q}_\alpha^i(\Delta t\, \mathbf{N}^{n+1/2-i})$
(2)	$\widetilde{\mathbf{P}}^n \mathbf{U}^0 - \sum \widetilde{\mathbf{P}}^i(\Delta t\, \mathbf{N}^{n+1/2-i})$
(3)	$\mathbf{U}^0 - \sum \widetilde{\mathbf{P}}^i(\Delta t\, \mathbf{N}^{n+1/2-i}) - \sum \widetilde{\mathbf{P}}^i(\Delta t\, \mathbf{G}p^{-1/2})$
(4)	$\mathbf{U}^0 - \sum \widetilde{\mathbf{P}}(\Delta t\, \mathbf{N}^{n+1/2-i})$

TABLE 2
$\Delta t\, \mathbf{G}p^{n-1/2}$ *in terms of* $\mathbf{U}^0$, $\mathbf{G}p^{-1/2}$, *and* $\mathbf{N}$. *All sums are from* $i = 1$ *to* n *unless otherwise noted.*

Version	$\Delta t\, \mathbf{G}p^{n-1/2} =$
Exact	$-(\mathbf{I} - \mathbf{P})(\Delta t\, \mathbf{N}^{n-1/2})$
(1)	$-\Delta t \sum \mathbf{Q}_\gamma^i \mathbf{N}^{n+1/2-i} - \Delta t\, \mathbf{Q}_\delta^n \mathbf{G}p^{-1/2}$
(2)	$-(\mathbf{I} - \widetilde{\mathbf{P}})(\Delta t\, \mathbf{N}^{n-1/2})$ $+ (\mathbf{I} - \widetilde{\mathbf{P}})\widetilde{\mathbf{P}}^{n-1}\mathbf{U}^0 - \sum_{i=2}^n (\widetilde{\mathbf{P}}^{i-1} - \widetilde{\mathbf{P}}^i)(\Delta t\, \mathbf{N}^{n+1/2-i}))$
(3)	$\widetilde{\mathbf{P}}^n \mathbf{G}p^{-1/2} - (\mathbf{I} - \widetilde{\mathbf{P}}) \sum \widetilde{\mathbf{P}}^{i-1} \Delta t\, \mathbf{N}^{n+1/2-i}$
(4)	$-(\mathbf{I} - \widetilde{\mathbf{P}})(\Delta t\, \mathbf{N}^{n-1/2})$

The coefficients in the formula for $\mathbf{U}^n$ using version (1) are defined by the recursion relations

$$\mathbf{Q}_\alpha^n = \widetilde{\mathbf{P}}(\mathbf{Q}_\alpha^{n-1}) - (\mathbf{I} - \widetilde{\mathbf{P}})(\mathbf{Q}_\beta^{n-1}), \quad \mathbf{Q}_\beta^n = \widetilde{\mathbf{P}}(\mathbf{Q}_\alpha^{n-1} + \mathbf{Q}_\beta^{n-1}),$$

with starting values $\mathbf{Q}_\alpha^1 = \mathbf{Q}_\beta^1 = \widetilde{\mathbf{P}}$, $\mathbf{Q}_\alpha^2 = 2\widetilde{\mathbf{P}}^2 - \widetilde{\mathbf{P}}$, $\mathbf{Q}_\beta^2 = 2\widetilde{\mathbf{P}}^2$, $\mathbf{Q}_\alpha^3 = 4\widetilde{\mathbf{P}}^3 - 3\widetilde{\mathbf{P}}^2$, and $\mathbf{Q}_\beta^3 = 4\widetilde{\mathbf{P}}^3 - \widetilde{\mathbf{P}}^2$. Note that $\mathbf{Q}_\alpha^n$ and $\mathbf{Q}_\beta^n$ are polynomials in $\widetilde{\mathbf{P}}$ and that the coefficient of $\widetilde{\mathbf{P}}^n$ in each is 2^{n-1}. Note also that the sum of the coefficients in $\mathbf{Q}_\alpha^n$ is one, and the sum of the coefficients in $\mathbf{Q}_\beta^n$ is n.

The coefficients in the formula for $\mathbf{G}p^{n-1/2}$ using version (1) are defined by the recursion relations

$$\mathbf{Q}_\gamma^n = \widetilde{\mathbf{P}}(\mathbf{Q}_\gamma^{n-1}) - (\mathbf{I} - \widetilde{\mathbf{P}})(\mathbf{Q}_\delta^{n-1}), \quad \mathbf{Q}_\delta^n = \widetilde{\mathbf{P}}(\mathbf{Q}_\gamma^{n-1} + \mathbf{Q}_\delta^{n-1}),$$

with starting values $\mathbf{Q}_\delta^1 = -\widetilde{\mathbf{P}}$ and $\mathbf{Q}_\gamma^1 = (\mathbf{I} - \widetilde{\mathbf{P}})$.

It is clear there is a trade-off in choosing any one form of $\mathbf{V}$. In the numerical results of the previous section, version (1) gives the smallest divergence of velocity when it remains stable; however, it suffers from a lack of robustness. Version (4), by contrast, defines the pressure gradient using only the most recent advection terms, as does an exact discrete projection, but produces very poor computational results. Version (2) appears to be robust and reliable but the velocity field, as seen in the steady example, can accumulate the imprint of the pressure gradient from previous times.

To date, approximate projection methods have followed the lead of exact projection methods in using a single projection to define both the new velocity and pressure. Here, however, motivated by the problems with each form, we introduce the concept of separate projections for the velocity and pressure. We propose an approximate projection method in which a version (1) projection is used to define the new velocity field, and a version (4) projection is then used to define the new pressure gradient. We will call this version (5). If the second projection is solved to full precision, this doubles the computational work of the projection step. However, in our numerical tests we have found that because the pressure from the first projection using version (1) is a good first guess for the pressure which would result from version (4), an algebraic reformulation of the second projection allows one to solve for the difference between these two pressures rather than for the new pressure itself. It is then only necessary to approximately solve the second equation (e.g., to require that the iterative solver reduce the residual by only one order of magnitude), resulting in little additional computational work. Note that the scalar being solved for is the difference in pressure due to the difference in formulation, not the pressure update in time. We will use this strategy for the results presented in this paper. This new version (5) has been successfully used to model incompressible axisymmetric flow with swirl in [16].

We comment here that computational cost is not a relevant factor in deciding between versions. The cost of solving $\mathbf{DG}\phi = \mathbf{DV}$ for ϕ, given $\mathbf{D}$ and $\mathbf{G}$, using an iterative solver, is a function of the smoothness of $\mathbf{V}$, the boundary conditions, and the desired reduction in residual. These factors vary between problems, and do not uniquely determine a "cheapest" version. Version (5) requires slightly more computational work than the others because of the need for a second solve, but using the approximation described above, the additional work is typically less than 10% of the total work of the projection step.

The expressions for projection (5) corresponding to those in Tables 1 and 2 are

$$\mathbf{U}^n = \widetilde{\mathbf{P}}^n \mathbf{U}^0 - \widetilde{\mathbf{P}}^n(\Delta t\ \mathbf{G}p^{-1/2}) - \widetilde{\mathbf{P}}(\Delta t\ \mathbf{N}^{n-1/2}) - \sum_{i=1}^{n-1} \widetilde{\mathbf{P}}^i(2\widetilde{\mathbf{P}} - \mathbf{I})(\Delta t\ \mathbf{N}^{n-1/2-i}),$$

$$\mathbf{G}p^{n-1/2} = (\mathbf{I} - \widetilde{\mathbf{P}})(\Delta t\ \mathbf{N}^{n-1/2}).$$

We note here that version (5) is fully second-order accurate for smooth problems; on the translating vortices problem the results are almost indistinguishable from those with version (2). For the random initial vorticity problem, the magnitude of the divergence with version (5) remains low in all cases; it is in fact always lower than that with version (2), typically by 20–40%, and it lacks the instability of version (1). For the simple steady problem, the curves for version (5) are indistinguishable from those for version (1).

Before performing a more detailed analysis of the projection version, we note that the results for each version can be expressed in a general form as

$$\mathbf{U}^n = \rho_n(\widetilde{\mathbf{P}})\mathbf{U}^0 - \Delta t \sum_{i=1}^{n} \alpha_i(\widetilde{\mathbf{P}})\mathbf{N}^{n+1/2-i} - \beta_n(\widetilde{\mathbf{P}})\mathbf{G}p^{-1/2}, \tag{4.1}$$

$$\Delta t\ \mathbf{G}p^{n-1/2} = -\Delta t\ (\mathbf{I}-\widetilde{\mathbf{P}})\mathbf{N}^{n-1/2} - \Delta t \sum_{i=2}^{n} \gamma_i(\widetilde{\mathbf{P}})\mathbf{N}^{n+1/2-i} + \gamma_n(\widetilde{\mathbf{P}})\mathbf{U}^0 - \Delta t\ \delta_n(\widetilde{\mathbf{P}})\mathbf{G}p^{-1/2}, \tag{4.2}$$

with the polynomial coefficients for each version and for an exact projection given in Table 3.

TABLE 3
Coefficients for each version of the approximate projection and for an exact projection.

Version	ρ_n	α_i	β_n	γ_i	δ_n
Exact	1	$\mathbf{P}$	0	0	0
(1)	$\mathbf{Q}_\alpha^n$	$\mathbf{Q}_\alpha^i$	$\mathbf{Q}_\beta^n$	$\mathbf{Q}_\gamma^i$	$\mathbf{Q}_\delta^n$
(2)	$\widetilde{\mathbf{P}}^n$	$\widetilde{\mathbf{P}}^i$	0	$\widetilde{\mathbf{P}}^{i-1} - \widetilde{\mathbf{P}}^i$	0
(3)	1	$\widetilde{\mathbf{P}}^i$	$\sum_{i=1}^n \widetilde{\mathbf{P}}^i$	$\widetilde{\mathbf{P}}^{i-1} - \widetilde{\mathbf{P}}^i$	$\widetilde{\mathbf{P}}^n$
(4)	1	$\widetilde{\mathbf{P}}$	0	0	0
(5)	$\widetilde{\mathbf{P}}^n$	$2\widetilde{\mathbf{P}}^{i+1} - \widetilde{\mathbf{P}}^i$	$\widetilde{\mathbf{P}}^n$	0	0

Performing a complete error analysis for each of these approximate schemes is not tractable. Instead, to characterize the effect of the "approximateness" of the

approximate projection in greater detail, we will use the formulae in (4.1)–(4.2) to compare the solution found using the approximate projection to the solution of the same problem found using the corresponding exact discrete projection $\mathbf{P}$. If we denote by $(\mathbf{U}^n, \mathbf{G}p^{n-1/2})$ the solution with $\widetilde{\mathbf{P}}$ and $(\mathbf{V}^n, \mathbf{G}q^{n-1/2})$ the solution with $\mathbf{P}$, then

$$\mathbf{U}^n - \mathbf{V}^n = (\rho^n(\widetilde{\mathbf{P}}) - \mathbf{P})\mathbf{U}^0 + \Delta t \left(-\sum_{i=1}^{n} \alpha_i(\widetilde{\mathbf{P}})\mathbf{N}_U^{n+1/2-i} + \sum_{i=1}^{n} \mathbf{P}\mathbf{N}_V^{n+1/2-i} - \beta_n(\widetilde{\mathbf{P}})\mathbf{G}p^{-1/2} \right)$$

and

$$\begin{aligned}\Delta t\ (\mathbf{G}p^{n-1/2} - \mathbf{G}q^{n-1/2}) = &-\Delta t\ (\mathbf{I} - \widetilde{\mathbf{P}})\mathbf{N}_U^{n-1/2} + \Delta t\ (\mathbf{I} - \mathbf{P})\mathbf{N}_V^{n-1/2} \\ &-\Delta t\ \sum_{i=2}^{n} \gamma_i(\widetilde{\mathbf{P}})\mathbf{N}_U^{n+1/2-i} + \gamma_n(\widetilde{\mathbf{P}})\mathbf{U}^0 - \Delta t\ \delta_n(\widetilde{\mathbf{P}})\mathbf{G}p^{-1/2},\end{aligned}$$

where $\mathbf{N}_U$ and $\mathbf{N}_V$ denote evaluation of the advective terms with $(\mathbf{U}^n, \mathbf{G}p^{n-1/2})$ and $(\mathbf{V}^n, \mathbf{G}q^{n-1/2})$, respectively.

If we rewrite these expressions, adding and subtracting terms involving $\mathbf{P}\mathbf{N}_U^{n+1/2-i}$ to the equations, we obtain

$$\begin{aligned}\mathbf{U}^n - \mathbf{V}^n = &(\rho^n(\widetilde{\mathbf{P}}) - \mathbf{P})\mathbf{U}^0 - \Delta t\ \sum_{i=1}^{n} (\alpha_i(\widetilde{\mathbf{P}}) - \mathbf{P})\mathbf{N}_U^{n+1/2-i} \\ &-\Delta t\ \sum_{i=1}^{n} \mathbf{P}(\mathbf{N}_U^{n+1/2-i} - \mathbf{N}_V^{n+1/2-i}) - \Delta t\ \beta_n(\widetilde{\mathbf{P}})\mathbf{G}p^{-1/2}\end{aligned}$$

and

$$\begin{aligned}\Delta t\ (\mathbf{G}p^{n-1/2} - \mathbf{G}q^{n-1/2}) = &-\Delta t\ (\mathbf{P} - \widetilde{\mathbf{P}})\mathbf{N}_U^{n-1/2} - \Delta t\ (\mathbf{I} - \mathbf{P})(\mathbf{N}_U^{n-1/2} - \mathbf{N}_V^{n-1/2}) \\ &-\Delta t\ \sum_{i=2}^{n} \gamma_i(\widetilde{\mathbf{P}})\mathbf{N}_U^{n+1/2-i} + \gamma_n(\widetilde{\mathbf{P}})\mathbf{U}^0 - \Delta t\ \delta_n(\widetilde{\mathbf{P}})\mathbf{G}p^{-1/2}.\end{aligned}$$

If we now take norms and use the fact that $||\mathbf{P}|| = 1$ and $||\mathbf{I} - \mathbf{P}|| = 1$, we obtain

$$\begin{aligned}||\mathbf{U}^n - \mathbf{V}^n|| \le &||(\rho^n(\widetilde{\mathbf{P}}) - \mathbf{P})\mathbf{U}^0|| + \Delta t\ \sum_{i=1}^{n} ||(\alpha_i(\widetilde{\mathbf{P}}) - \mathbf{P})\mathbf{N}_U^{n+1/2-i}|| \\ &+\Delta t\ \sum_{i=1}^{n} ||\mathbf{N}_U^{n+1/2-i} - \mathbf{N}_V^{n+1/2-i}|| + \Delta t\ ||\beta_n(\widetilde{\mathbf{P}})\mathbf{G}p^{-1/2}||\end{aligned} \tag{4.3}$$

and

$$\begin{aligned}\Delta t\ ||\mathbf{G}p^{n-1/2} - \mathbf{G}q^{n-1/2}|| \le &\Delta t\ ||(\mathbf{P} - \widetilde{\mathbf{P}})\mathbf{N}_U^{n-1/2}|| + \Delta t\ ||\mathbf{N}_U^{n-1/2} - \mathbf{N}_V^{n-1/2}|| \\ &+\Delta t\ \sum_{i=2}^{n} ||\gamma_i(\widetilde{\mathbf{P}})|N_U^{n+1/2-i}|| + ||\gamma_n(\widetilde{\mathbf{P}})|U^0|| + \Delta t\ ||\delta_n(\widetilde{\mathbf{P}})\mathbf{G}p^{-1/2}||.\end{aligned} \tag{4.4}$$

For the methods considered here $\mathbf{N}_U$ depends on $\mathbf{U}$ and $\mathbf{G}p$ smoothly, likewise $\mathbf{N}_V$ depends on $\mathbf{V}$ and $\mathbf{G}q$ smoothly, so we can assume $||\mathbf{N}_U - \mathbf{N}_V||$ terms are small and can be absorbed into the left-hand side of the equation with the introduction of a

constant multiplier C on the right-hand side. With that assumption, summing (4.3) and (4.4) over n yields

$$\sum_{n=1}^{N}\{||\mathbf{U}^n - \mathbf{V}^n|| + \Delta t\,||\mathbf{G}p^{n-1/2} - \mathbf{G}q^{n-1/2}||\} \le C\sum_{n=1}^{N}\Bigg\{||\rho^n(\widetilde{\mathbf{P}}) - \mathbf{P})\mathbf{U}^0||$$

$$+\Delta t\,\sum_{i=1}^{n}||(\alpha_i(\widetilde{\mathbf{P}}) - \mathbf{P})\mathbf{N}_U^{n+1/2-i}|| + \Delta t\,||\beta_n(\widetilde{\mathbf{P}})\mathbf{G}p^{-1/2}|| + ||\gamma_n(\widetilde{\mathbf{P}})\mathbf{U}^0|| \tag{4.5}$$

$$+\Delta t\,||\delta_n(\widetilde{\mathbf{P}})\mathbf{G}p^{-1/2}|| + \Delta t\,||(\mathbf{P} - \widetilde{\mathbf{P}})\mathbf{N}_U^{n-1/2}|| + \Delta t\,\sum_{i=2}^{n}||\gamma_i(\widetilde{\mathbf{P}})\mathbf{N}_U^{n+1/2-i}||\Bigg\}.$$

From (4.5) we can exactly identify the polynomial expressions which contribute to the error due to the approximateness of the projection; these terms will be examined further in the next section.

Before examining the spatial structure of the different approximate projection schemes, we note that the derivation presented here also provides a characterization of the divergence of the solution of the approximate projection solution. If we apply $\mathbf{D}$ to (4.1) we obtain

$$\mathbf{D}\mathbf{U}^n = \mathbf{D}\rho^n(\widetilde{\mathbf{P}})\mathbf{U}^0 - \Delta t\,\sum_{i=1}^{n}\mathbf{D}\alpha_i(\widetilde{\mathbf{P}})\mathbf{N}_U^{n+1/2-i} - \Delta t\,\mathbf{D}\beta_n(\widetilde{\mathbf{P}})\mathbf{G}p^{-1/2}.$$

Thus, the structure of the spatial operators described below also provides a characterization of the divergence. Furthermore, since $\widetilde{\mathbf{P}}$ and $\mathbf{P}$ do not change discretely divergence-free fields, the error terms in the estimate (4.5) are discrete gradients. This explains the correspondence between the quality of the computational results and the behavior of the discrete divergence of the solution.

5. Spatial analysis of the projection. The temporal analysis of the previous section provides a basic framework for understanding the behavior of approximate projection schemes. The details of this behavior depend on the details of the spatial structure of the approximate projections. In this section we will analyze that behavior for the **N5, N9**, and **C5** approximate projections. In particular, we will look at how well the polynomials of $\widetilde{\mathbf{P}}$ that characterize the approximate projections approximate the corresponding exact discrete projection.

Before looking at the behavior of the approximate projections, however, we first need to characterize the structure of the corresponding exact discrete projection, $\mathbf{I} - \mathbf{G}(\mathbf{D}\mathbf{G})^{-1}\mathbf{D}$. As noted earlier, the difficulties associated with using these exact discrete projections arise from the (locally) decoupled stencils associated with $\mathbf{D}\mathbf{G}$. This local decoupling is related to the presence of elements in the kernel of $\mathbf{G}$ other than constants. In two dimensions, for the nodal scheme dim ker $\mathbf{G} = 2$ and for the cell-centered scheme dim ker $\mathbf{G} = 4$. For each of these methods one of these elements corresponds to constants and the other modes are spurious. For the nodal scheme this additional mode corresponds to the Nyquist frequency in both directions. This mode is also in ker $\mathbf{G}$ for the cell-centered scheme, as well as vertical and horizontal "stripe" modes corresponding to the Nyquist frequency in one component and constant in the other. These modes can be seen in Figure 2(c), (g).

To characterize the projections we will look at their Fourier behavior. After the Fourier transform, the projection has the form of a 2×2 matrix for each wave number. As noted above, discretely divergence-free vector fields are unchanged by

both the exact discrete and the approximate projections, thus both the exact discrete and approximate versions share an eigenvalue of 1 with a common eigenvector. To characterize the behavior we can then focus on the other eigenvalue, λ_{min}. For $\mathbf{P}$, $\lambda_{min} = 0$ except for wave numbers corresponding to elements in ker $\mathbf{G}$. In Figure 5 we show λ_{min} for the **N9, N5**, and **C5** approximate projection operators, $\widetilde{\mathbf{P}}$. The plots of λ_{min} for **N_EX** would have a single spike to value 1 at the center, corresponding to the checkerboard mode; the plots for **C_EX** would have spikes at the center and the midpoints of each side, corresponding to the checkerboard and "stripe" modes. In Figures 5–9 the center of each plot corresponds to the Nyquist frequency in both directions with constants at the corner. (The peak at constants has been suppressed.)

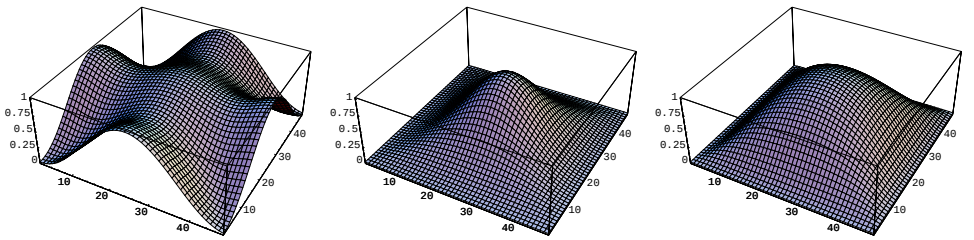


FIG. 5. *Smallest eigenvalue λ_{min} of the approximate projections* (a) **C5**, (b) **N9**, *and* (c) **N5**.

In each case we see that λ_{min} for the approximate projection operator has a smooth peak near the elements of ker $\mathbf{G}$ for the corresponding exact discrete projection. Thus, because of additional elements in ker $\mathbf{G}$ the cell-centered discretization broadens the range over which the approximation is a poor one. We also note that the eigenvalue structure of the five-point nodal projection is similar to that of the nine-point nodal only with a slightly wider distribution. A key property of each discretization is that $\widetilde{\mathbf{P}}^n \to \mathbf{P}$ as $n \to \infty$.

From the characterization of the approximate projection in Figure 5 we can already explain the poor performance of versions (3) and (4) of the approximate projection method. In version (4), $\alpha_i(\widetilde{\mathbf{P}}) = \widetilde{\mathbf{P}}$ and so, recalling (4.1), we see that at each time step $\mathbf{U}$ accumulates a portion of the discrete gradient component of $\mathbf{N}$, resulting in an error that builds over time. By contrast, for version (2), $\alpha_i(\widetilde{\mathbf{P}}) = \widetilde{\mathbf{P}}^i$, so that the contributions from any fixed time converge to the correct contribution as n increases, because $\widetilde{\mathbf{P}}^n \to \mathbf{P}$.

The difficulty with version (3), on the other hand, is caused by a residual imprint of pressure as characterized by the β_n. Recall that for version (3), $\beta_n = \sum_i \widetilde{\mathbf{P}}^i$. For each eigenvalue we can write

$$\widetilde{\mathbf{P}}(\mathbf{G}\phi(\xi)) = \lambda(\xi)\mathbf{G}\phi(\xi), \quad 0 < \lambda < 1;$$

then for version (3)

$$\beta_n(\widetilde{\mathbf{P}})\mathbf{G}(\xi) \equiv (\widetilde{\mathbf{P}} + \widetilde{\mathbf{P}}^2 + \cdots + \widetilde{\mathbf{P}}^n)\mathbf{G}\phi(\xi) \to \frac{\lambda}{\lambda - 1}\mathbf{G}\phi(\xi),$$

so that the pressure gradient terms in $\mathbf{U}$ that are annihilated by the exact discrete projection asymptote to a nonnegligible error in $\mathbf{U}$ with an approximate projection. By contrast, we note that this type of term is absent for version (2) (i.e., $\beta_n = 0$) which is otherwise similar in structure to version (3).

To characterize the other versions of the approximate projection we look at the α_i in more detail. For each of the other three versions ((1), (2), and (5)) of the approximate projection, the lowest power of $\widetilde{\mathbf{P}}$ in α_i increases with i. Furthermore, the coefficients in α_i sum to 1. Therefore, for each version $\alpha_i(\widetilde{\mathbf{P}}) \to \mathbf{P}$ as $i \to \infty$. To obtain a more detailed comparison we need to look at the behavior of the α_i for small i.

For each version $\alpha_1(\widetilde{\mathbf{P}}) = \widetilde{\mathbf{P}}$ which is described above and illustrated in Figure 5. For the next two terms in the expressions, for version (1), $\alpha_2 = 2\widetilde{\mathbf{P}}^2 - \widetilde{\mathbf{P}}$, $\alpha_3 = 4\widetilde{\mathbf{P}}^3 - 3\widetilde{\mathbf{P}}^2$; for version (2), $\alpha_2 = \widetilde{\mathbf{P}}^2, \alpha_3 = \widetilde{\mathbf{P}}^3$; and for version (5), $\alpha_2 = 2\widetilde{\mathbf{P}}^2 - \widetilde{\mathbf{P}}, \alpha_3 = 2\widetilde{\mathbf{P}}^3 - \widetilde{\mathbf{P}}^2$. In Figures 6 and 7 we plot α_2 and α_3, respectively, for **C5_1**, **C5_2**, and **C5_5**, and **N9_1**, **N9_2**, and **N9_5**. (The **N5** results are similar to the **N9** results, only somewhat wider.)

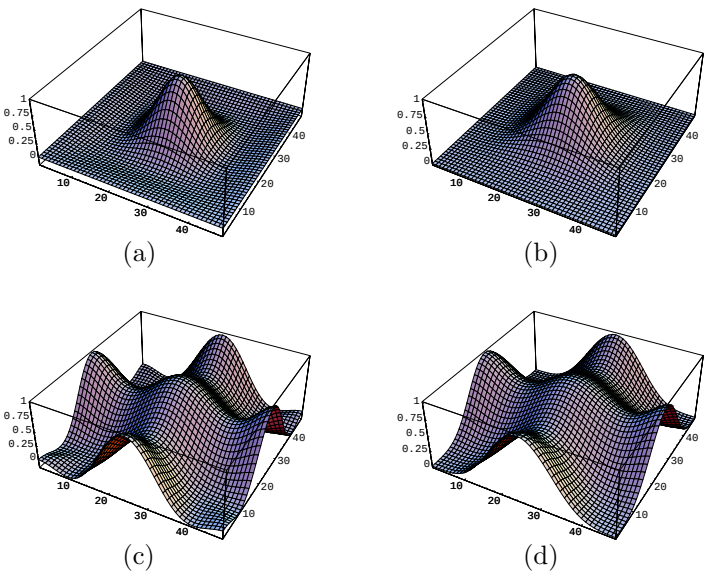


FIG. 6. α_2 *for* (a) **C5_1, C5_5**, (b) **C5_2**, (c) **N9_1, N9_5**, (d) **N9_2**.

For α_2, we note that for versions (1) and (5) there is a narrower peak than for version (2) near the frequencies that are in ker $\mathbf{G}$, with some small negative values at intermediate frequencies. Overall it is smaller in an L^2 integral sense. A similar pattern is observed for the α_3's. For version (2), the peak is fairly broad but the values are between 0 and 1. For version (5), the peak is narrow and there is a small region where the values are slightly negative ($\lambda_{min} > -.04$). For version (1), the peak near the ker $\mathbf{G}$ elements is sharpest but there are fairly large negative values ($\lambda_{min} = -.25$). Although version (1) is the smallest in an integral sense, the negative values can introduce perturbations into the solution which may be a contributing factor to the nonrobustness of version (1).

There are two other potential sources for the behavior of version (1). As with version (3), version (1) contains a residual contribution from the pressure gradient (i.e., $\beta_n \neq 0$). For version (2) there is no such term; for version (5) this term is $\widetilde{\mathbf{P}}^n$ which goes to zero on the range of $\mathbf{G}$ as $n \to \infty$. For version (1), this behavior is characterized by the behavior of the $\mathbf{Q}_\beta$'s defined in section 5. In Figure 8 we

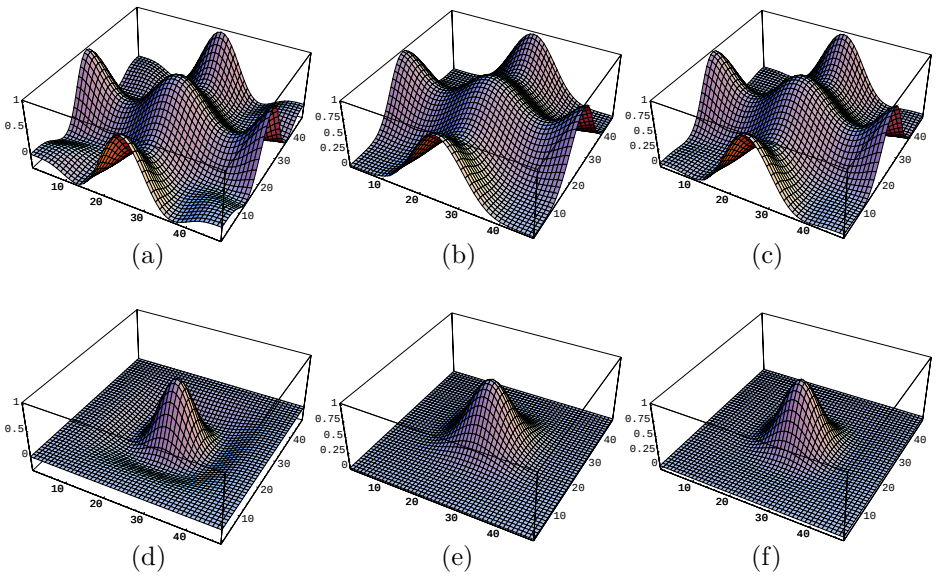


FIG. 7. α_3 *for* (a) **C5_1**, (b) **C5_2**, (c) **C5_5**, (d) **N9_1**, (e) **N9_2**, (f) **N9_5**.

plot the functions β_2, β_3, and β_4 for **N9_1**. Unlike the operators for version (3) which asymptotically converge to nonzero values, the operators described here do not converge to zero for any fixed frequency except those near elements in ker G. However, the peak value near elements in ker $\mathbf{G}$ is n for the β_n for version (1), just as in version (3) but with a narrower peak. This growth in residual influence of $\mathbf{G}p^{-1/2}$ also has the potential for contribution to the nonrobust behavior of version (1).

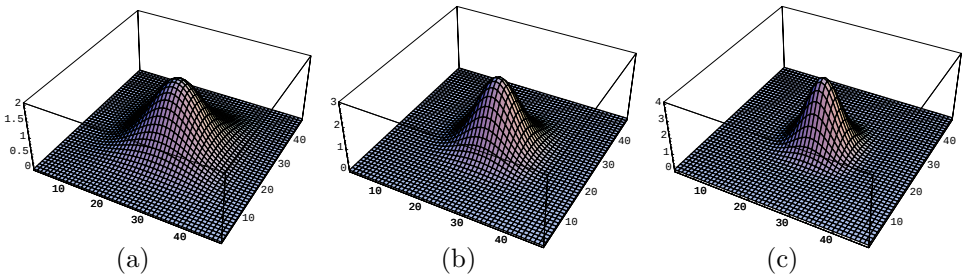


FIG. 8. (a) β_2, (b) β_3, *and* (c) β_4 *for the* **N9_1** *approximate projection.*

The final issue regarding the spatial structure associated with the approximate projection versions concerns the error in $\mathbf{G}p^{n-1/2}$ associated with the $\mathbf{N}$'s (recall equation (4.2)). For all versions the leading error term looks like $\mathbf{P} - \widetilde{\mathbf{P}}$, which we have already examined. For versions (4) and (5) there are no other terms. In Figure 9 we show γ_2, γ_3, and γ_4 for **N9_1**. The functions γ_i for **N9_2** and **N9_3** are similar to those shown but have smaller overall magnitudes (0.2 v. 0.4 for γ_2, 0.15 v. 0.3 for γ_3, and 0.1 v. 0.3 for γ_4) and, unlike for version (1), remain positive at all frequencies.

To summarize the findings, we first note that the unacceptable performance of versions (3) and (4) can be related to an imprint of the old pressure gradients and a failure to damp the gradient components of the advection terms, respectively. We have

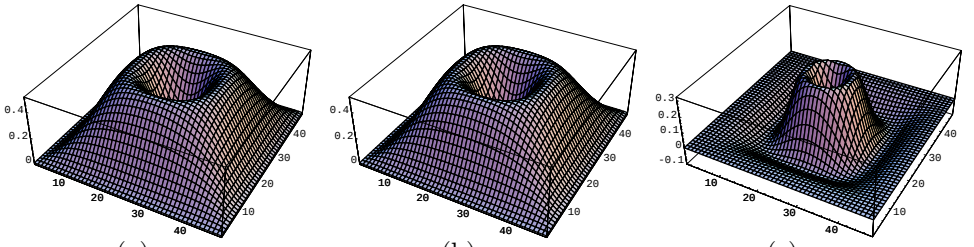


FIG. 9. (a) γ_2, (b) γ_3, *and* (c) γ_4 *for the* **N9_1** *approximate projection.*

identified three features of version (1) which may contribute to its lack of robustness. The first is the oscillatory nature of the α_i; however, this is a relatively small effect and probably plays a negligible role in determining the overall behavior. The other two features, a growing imprint of the old pressure gradient on $\mathbf{U}$ given by the β_i's combined with the errors introduced into $\mathbf{G}p$ from the advection terms specified by the γ_i's, provide a mechanism for feeding back oscillatory behavior into the solution.

6. Advection-projection coupling. As we can see from the numerical results and the analysis, the nature of the difficulties for versions (3) and (4) differs from that of version (1). This is evident as well in further numerical testing. Here we perform two additional numerical experiments with the random initial vorticity problem. In the first we halve the time step in each calculation by reducing the CFL number from 0.9 to 0.45 to examine the effect on the solution. In the top row of Figure 10, for resolution 64^2, we see that the results from versions (3) and (4) are essentially unchanged but the results from version (1) are markedly improved (i.e., lower divergence) relative to the results shown in Figure 3. Version (1) has also been rerun for resolution 128^2, with the same improvement.

Our second experiment is to modify the predictor so that the coupling between the pressure gradient $\mathbf{G}p^{n-1/2}$ and the advective update term $\mathbf{N}^{n+1/2}$ is reduced, by MAC-projecting the advected velocities as well as the advection velocities in the predictor step. This modification is described in more detail in Appendix A. The results for the same cases as above are shown in the bottom row of Figure 10; here the results of versions (3) and (4) are worsened (in the sense that the divergence is larger) but the divergence from version (1) is again noticeably lower, again compared to Figure 3. This experiment was run with CFL = 0.9.

The results in this section suggest that the poor results obtained with versions (3) and (4), and the acceptable results of versions (2) and (5), are independent of the choice of advection scheme. By contrast, the behavior of version (1) for inviscid flow is intimately coupled to the details of the particular advection method.

7. Conclusions and future work. When adopting an approximate projection formulation in which the projection operator does not exactly enforce a discrete divergence condition on the velocity field, there are several options for choosing which vector field to project. In this paper we have explored these options computationally and developed an analytical framework that explains the computational results. The results show that projecting the velocity field rather than the velocity increment is necessary to produce acceptable results. We also find that projecting $\frac{\mathbf{U}^{*,n+1}}{\Delta t}$ (version (1)) can produce a lower divergence in the computed solution than projecting $\frac{\mathbf{U}^{*,n+1}}{\Delta t} + \mathbf{G}p^{n-1/2}$ (version (2)); however, it is not as robust as version (2) which always

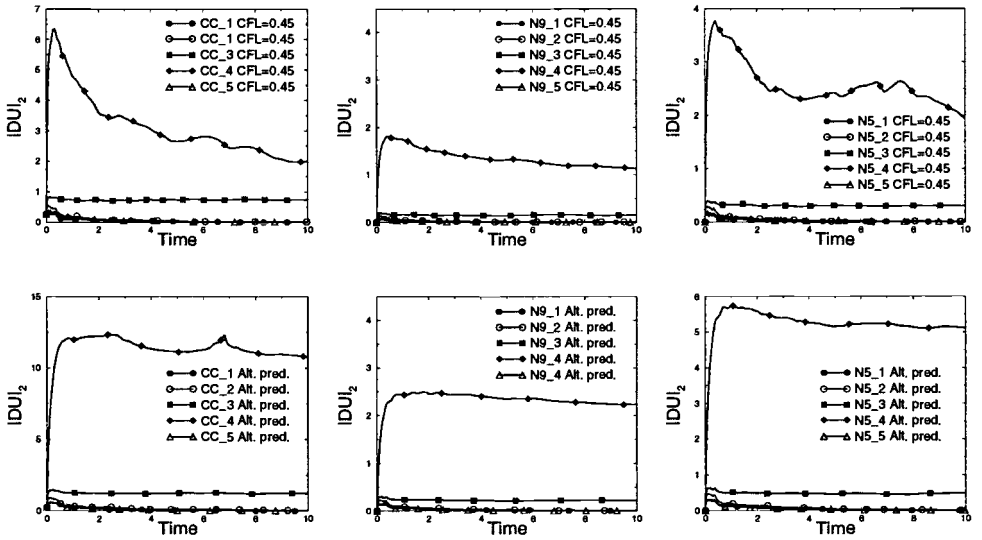


FIG. 10. *L^2 norms of* $\mathbf{DU}$ *from* $t = 0$ *to* 10 *for the random initial vorticity problem with CFL* $= 0.45$ *(top row), or the alternate predictor (bottom row).*

produces acceptable results.

We have also introduced a new approximate projection formulation based on projecting $\frac{\mathbf{U}^{*,n+1}}{\Delta t}$ to update $\mathbf{U}^{n+1}$ while projecting $\frac{\mathbf{U}^{*,n+1}-\mathbf{U}^n}{\Delta t}+\mathbf{G}p^{n-1/2}$ as an iterative correction to update p. The new formulation produces results similar to version (1) when version (1) is successful and exhibits the same robustness as version (2).

In Part II we will present the extension of the analysis to viscous flows, in the context of a Crank–Nicolson discretization of the viscous terms. This discretization requires the application of the inverse diffusion operator before the projection step. As a consequence, the expressions for $\mathbf{U}^n$ and $\mathbf{G}p^{n-1/2}$, presented in Tables 1 and 2 of this paper, and the analysis of these expressions, become much more complicated. One interesting consequence is that while minor variations in the advection step can be exploited to obtain better results with version (1) in the case of inviscid flow, when viscosity is introduced the poor behavior of version (1) cannot be salvaged.

Further issues and future work will include the extension of the analysis for variable density flows, adaptive mesh calculations with subcycling in time, and low Mach number flows with nonzero divergence constraints.

Appendix A. The predictor/corrector step. In this section we first describe the construction of the advective update $\mathbf{N}^{n+1/2}$ used in our original tests; then we discuss the modifications as discussed in section 6.

In the predictor we first extrapolate the normal velocities to cell faces at $t^{n+1/2}$. For face $(i+1/2, j)$ this gives

$$\widetilde{u}^{L,n+1/2}_{i+1/2,j} = u^n_{i,j} + \left(\frac{\Delta x}{2} - u^n_{i,j}\frac{\Delta t}{2}\right)(u_x^{n,lim})_{i,j} + \frac{\Delta t}{2}(-(\widehat{vu_y})_{i,j} - (G_x p)^{n-1/2}_{i,j})$$

extrapolated from (i, j), and

$$\widetilde{u}^{R,n+1/2}_{i+1/2,j} = u^n_{i+1,j} - \left(\frac{\Delta x}{2} + u^n_{i+1,j}\frac{\Delta t}{2}\right)(u_x^{n,lim})_{i+1,j} + \frac{\Delta t}{2}(-(\widehat{vu_y})_{i+1,j} - (G_x p)^{n-1/2}_{i+1,j})$$

extrapolated from $(i+1,j)$. Here the first derivatives normal to the face (in this case $u_x^{n,lim}$) are evaluated using a monotonicity-limited fourth-order slope approximation [6] (except in the translating vortices problem, where unlimited fourth-order slopes are used), and the transverse derivative terms ($\widehat{vu_y}$ in this case) are evaluated exactly as in [1]. $\mathbf{G} = (G_x, G_y)$ is a discretization of the gradient operator. For a nodal pressure field,

$$(G_x p)_{i,j} = \frac{1}{4\Delta x}(p_{i+1/2,j+1/2} + p_{i+1/2,j-1/2} - p_{i-1/2,j+1/2} - p_{i-1/2,j-1/2}),$$
$$(G_y p)_{i,j} = \frac{1}{4\Delta y}(p_{i+1/2,j+1/2} + p_{i-1/2,j+1/2} - p_{i+1/2,j-1/2} - p_{i-1/2,j-1/2}).$$

For a cell-centered pressure field,

$$(G_x p)_{i,j} = \frac{1}{2\Delta x}(p_{i+1,j} - p_{i-1,j}),$$
$$(G_y p)_{i,j} = \frac{1}{2\Delta x}(p_{i,j+1} - p_{i,j-1}).$$

Analogous formulae are used to predict values for $\widetilde{u}^{L/R,n+1/2}_{i-1/2,j}$, $\widetilde{v}^{F/B,n+1/2}_{i,j+1/2}$, and $\widetilde{v}^{F/B,n+1/2}_{i,j-1/2}$.

The normal velocity at each face is then determined by an upwinding procedure based on the states predicted from the cell centers on either side.

$$\widetilde{u}^{n+1/2}_{i+1/2,j} = \begin{cases} \widetilde{u}^{L,n+1/2} & \text{if } \widetilde{u}^{L,n+1/2} > 0 \text{ and } \widetilde{u}^{L,n+1/2} + \widetilde{u}^{R,n+1/2} > 0, \\ 0 & \text{if } \widetilde{u}^{L,n+1/2} \le 0, \widetilde{u}^{R,n+1/2} \ge 0 \text{ or } \widetilde{u}^{L,n+1/2} + \widetilde{u}^{R,n+1/2} = 0, \\ \widetilde{u}^{R,n+1/2} & \text{if } \widetilde{u}^{R,n+1/2} < 0 \text{ and } \widetilde{u}^{L,n+1/2} + \widetilde{u}^{R,n+1/2} < 0. \end{cases}$$

We follow a similar procedure to construct $\widetilde{v}^{n+1/2}_{i,j+1/2}$.

In order to enforce the divergence-free condition on these edge-based velocities we now impose the MAC projection (see [4]). The equation

$$\mathbf{D}^{MAC}(\mathbf{G}^{MAC}\phi^{MAC}) = \mathbf{D}^{MAC}(\widetilde{U}^{n+1/2})$$

is solved for ϕ^{MAC}, with homogeneous Neumann boundary conditions on all physical boundaries except for outflow, where ϕ^{MAC} is set to zero to enforce the "no tangential acceleration" criterion. Here

$$\mathbf{D}^{MAC}(\widetilde{U}^{n+1/2}) = \frac{\widetilde{u}^{n+1/2}_{i+1/2,j} - \widetilde{u}^{n+1/2}_{i-1/2,j}}{\Delta x} + \frac{\widetilde{v}^{n+1/2}_{i,j+1/2} - \widetilde{v}^{n+1/2}_{i,j-1/2}}{\Delta y}$$

and $\mathbf{G}^{MAC} = -(\mathbf{D}^{MAC})^T$ so that

$$(G_x^{MAC}\phi^{MAC})_{i+1/2,j} = \frac{(\phi^{MAC}_{i+1,j} - \phi^{MAC}_{i,j})}{\Delta x}, \quad (G_y^{MAC}\phi^{MAC})_{i,j+1/2} = \frac{(\phi^{MAC}_{i,j+1} - \phi^{MAC}_{i,j})}{\Delta x}.$$

The advection velocity U^{ADV} is then defined by

$$u^{ADV}_{i+1/2,j} = \widetilde{u}^{n+1/2}_{i+1/2,j} - (G_x^{MAC}\phi^{MAC}),$$
$$v^{ADV}_{i,j+1/2} = \widetilde{v}^{n+1/2}_{i,j+1/2} - (G_y^{MAC}\phi^{MAC}).$$

At this point the tracing step from cell centers to edges is performed for both the normal and tangential velocity components exactly as above, except that the upwinding now depends on the MAC-projected advection velocity, i.e.,

$$\widetilde{U}_{i+1/2,j} = \begin{cases} \widetilde{U}^L & \text{if } u^{ADV}_{i+1/2,j} > 0, \\ 1/2(\widetilde{U}^L + \widetilde{U}^R) & \text{if } u^{ADV}_{i+1/2,j} = 0, \\ \widetilde{U}^R & \text{if } u^{ADV}_{i+1/2,j} < 0. \end{cases}$$

For the original advective term, we now define

$$\mathbf{N}_{i,j} = \frac{(u^{ADV}_{i+1/2,j}\widetilde{U}_{i+1/2,j} - u^{ADV}_{i-1/2,j}\widetilde{U}_{i-1/2,j})}{\Delta x} + \frac{(v^{ADV}_{i,j+1/2}\widetilde{U}_{i,j+1/2} - v^{ADV}_{i,j-1/2}\widetilde{U}_{i,j-1/2})}{\Delta y}.$$

The modification of the predictor as described in section 6 is the replacement of $\widetilde{U}$ by $\widetilde{\widetilde{U}} = (\widetilde{\widetilde{u}}, \widetilde{\widetilde{v}})$ in the above expression, where

$$\begin{aligned} \widetilde{\widetilde{u}}_{i+1/2,j} &= u^{ADV}_{i,j+1/2}, \\ \widetilde{\widetilde{u}}_{i,j+1/2} &= \widetilde{u}_{i,j+1/2} - \frac{1}{4\Delta x}(\phi^{MAC}_{i+1,j} + \phi^{MAC}_{i+1,j-1} - \phi^{MAC}_{i-1,j} - \phi^{MAC}_{i-1,j-1}), \\ \widetilde{\widetilde{v}}_{i+1/2,j} &= \widetilde{v}_{i+1/2,j} - \frac{1}{4\Delta x}(\phi^{MAC}_{i,j+1} + \phi^{MAC}_{i-1,j+1} - \phi^{MAC}_{i,j-1} - \phi^{MAC}_{i-1,j-1}), \\ \widetilde{\widetilde{v}}_{i,j+1/2} &= v^{ADV}_{i+1/2,j}. \end{aligned}$$

Then

$$\mathbf{N}_{i,j} = \frac{(u^{ADV}_{i+1/2,j}\widetilde{\widetilde{U}}_{i+1/2,j} - u^{ADV}_{i-1/2,j}\widetilde{\widetilde{U}}_{i-1/2,j})}{\Delta x} + \frac{(v^{ADV}_{i,j+1/2}\widetilde{\widetilde{U}}_{i,j+1/2} - v^{ADV}_{i,j-1/2}\widetilde{\widetilde{U}}_{i,j-1/2})}{\Delta y}.$$

Appendix B. Approximate projection operators in three dimensions. In this appendix we provide a characterization of the extension of the three different spatial approximate projections considered above to three space dimensions. We will refer to the seven-point cell-centered discretization as **C7**, the 21-point nodal discretization as **N21**, and the seven-point nodal discretization as **N7.** As before, the approximate projections do not alter discretely divergence-free vector fields. In three dimensions this means that the discrete projection operator is a 3×3 matrix for each frequency with two eigenvalues of 1. The other eigenvalue, λ_{min}, represents the degree to which gradients are damped by the projection. For **C7** there are 8 elements in ker **G** while for **N7** and **N21** there is a one-dimensional family of elements. As a further characterization of these schemes, we can expand the smallest eigenvalue about the constant mode to obtain the behavior of the different approximations for low frequencies. In particular, for **C7** we obtain

$$\lambda_{min}(k_1, k_2, k_3) = \frac{(k_1^4 + k_1^4 + k4^2)}{2(k_1^2 + k_2^2 + k_3^2)},$$

for **N21** we obtain

$$\lambda_{min}(k_1, k_2, k_3) = \frac{(k_1^2k_2^2 + k_1^2k_3^2 + k_2^2k_3^2)}{6(k_1^2 + k_2^2 + k_3^2)},$$

and for **N7** we obtain

$$\lambda_{min}(k_1, k_2, k_3) = \frac{(k_1^2 k_2^2 + k_1^2 k_3^2 + k_2^2 k_3^2)}{2(k_1^2 + k_2^2 + k_3^2)}.$$

REFERENCES

[1] A. S. Almgren, J. B. Bell, P. Colella, L. H. Howell, and M. Welcome, *A conservative adaptive projection method for the variable density incompressible Navier-Stokes equations*, J. Comput. Phys., 142 (1998), pp. 1–46.

[2] A. S. Almgren, J. B. Bell, and W. G. Szymczak, *A numerical method for the incompressible Navier–Stokes equations based on an approximate projection*, SIAM J. Sci. Comput., 17 (1996), pp. 358–369.

[3] J. B. Bell, P. Colella, and H. M. Glaz, *A second-order projection method for the incompressible Navier–Stokes equations*, in the AIAA 8th Computational Fluid Dynamics Conference, Honolulu, 1987, pp. 789–794.

[4] J. B. Bell, P. Colella, and L. H. Howell, *An efficient second-order projection method for viscous incompressible flow*, in the AIAA 10th Computational Fluid Dynamics Conference, Honolulu, 1991, pp. 360–367.

[5] A. J. Chorin, *Numerical solution of the Navier-Stokes equations*, Math. Comp., 22 (1968), pp. 745–762.

[6] P. Colella, *A direct Eulerian MUSCL scheme for gas dynamics*, SIAM J. Comput., 6 (1985), pp. 104–117.

[7] M. Fortin, *Numerical solutions of the steady state Navier-Stokes equations*, in Numerical Methods in Fluid Dynamics, 1972, AGARD-LS-48.

[8] L. H. Howell and J. B. Bell, *An adaptive mesh projection method for viscous incompressible flow*, SIAM J. Sci. Comput., 18 (1997), pp. 996–1013.

[9] G. Joyce and D. Montgomery, *Negative temperature states for a two-dimensional guiding center plasma*, J. Plasma Phys., 10 (1973), p. 107.

[10] M. F. Lai, *A Projection Method for Reacting Flow in the Zero Mach Number Limit*, Ph.D. thesis, University of California at Berkeley, 1993.

[11] M. F. Lai, J. B. Bell, and P. Colella, *A projection method for combustion in the zero Mach number limit*, in the AIAA 11th Computational Fluid Dynamics Conference, Orlando, FL, 1993, pp. 776–783.

[12] D. F. Martin and P. Colella, *A cell-centered adaptive projection method for the incompressible Euler equations*, J. Comp. Phys., 163 (2000), pp. 271–312.

[13] J. C. McWilliams, *The emergence of isolated coherent vortices in turbulent flow*, J. Fluid Mech., 146 (1984), pp. 21–43.

[14] M. L. Minion, *A projection method for locally refined grids*, J. Comput. Phys., 127 (1996), pp. 158–178.

[15] D. Montgomery, X. Shan, and W. H. Matthaeus, *Navier-Stokes relaxation to sinh-Poisson states at finite Reynolds numbers*, Phys. Fluids A, 5 (1993), pp. 2207–2216.

[16] D. S. Nolan, A. S. Almgren, and J. B. Bell, *High Reynolds Number Simulations of Axisymmetric Tornado-Like Vortices with Adaptive Mesh Refinement*, LBNL Report LBNL-42860, Lawrence Berkeley National Laboratory, Berkeley, CA, 1999.

[17] R. B. Pember, A. S. Almgren, J. B. Bell, P. Colella, L. H. Howell, and M. Lai, *A higher-order projection method for the simulation of unsteady turbulent nonpremixed combustion in an industrial burner*, in Transport Phenomena in Combustion, S. H. Chan, ed., Taylor and Francis, Washington, DC, 1996, pp. 1200–1211.

[18] R. B. Pember, L. H. Howell, J. B. Bell, P. Colella, W. Y. Crutchfield, W. A. Fiveland, and J. P. Jessee, *An adaptive projection method for unsteady, low-Mach number combustion*, Comb. Sci. Tech., 140 (1998), pp. 123–168.

[19] W. J. Rider, *Approximate Projection Methods for Incompressible Flow: Implementation, Variants and Robustness*, Tech. report LA-UR-94-2000, Los Alamos National Laboratory, Los Alamos, NM, 1994.

[20] W. J. Rider, *Filtering non-solenoidal modes in numerical solutions of incompressible flows*, Internat. J. Numer. Methods Fluids, 28 (1998), pp. 789–814.

SIAM J. SCI. COMPUT.
Vol. 22, No. 4, pp. 1160–1183
© 2000 Society for Industrial and Applied Mathematics

A LEVINSON–GALERKIN ALGORITHM FOR REGULARIZED TRIGONOMETRIC APPROXIMATION*

THOMAS STROHMER†

Abstract. Trigonometric polynomials are widely used for the approximation of a smooth function from a set of nonuniformly spaced samples. If the samples are perturbed by noise, a good choice for the polynomial degree of the trigonometric approximation becomes an essential issue to avoid overfitting and underfitting of the data. Standard methods for trigonometric least squares approximation assume that the degree for the approximating polynomial is known a priori, which is usually not the case in practice. We derive a multilevel algorithm that recursively adapts to the least squares solution of suitable degree. We analyze under which conditions this multilevel approach yields the optimal solution. The proposed algorithm computes the solution in at most $\mathcal{O}(rM + M^2)$ operations (M being the polynomial degree of the approximation and r being the number of samples) by solving a family of nested Toeplitz systems. It is shown how the presented method can be extended to multivariate trigonometric approximation. We demonstrate the performance of the algorithm by applying it in echocardiography to the recovery of the boundary of the left ventricle of the heart.

Key words. trigonometric approximation, Toeplitz matrix, Levinson algorithm, multilevel method

AMS subject classifications. 65T10, 42A10, 65D10, 65F10

PII. S1064827597329254

1. Introduction.

The necessity of recovering a function from a finite set of nonuniformly spaced measurements arises in areas as diverse as digital signal processing, geophysics, spectroscopy, and medical imaging. The measurements $\{s_j\}_{j=1}^r$ are often distorted by several kinds of errors. Hence a complete reconstruction of the function from the perturbed data $s_j^\varepsilon = s_j + \nu_j$ is not possible. Often the function to be reconstructed is smooth, in which case a trigonometric polynomial of relatively low degree (compared with the possibly huge number of samples) can provide a good approximation to the function. This trigonometric approximation may be found by solving the least squares problem

$$\min_{p \in \boldsymbol{P}_M} \sum_{j=1}^{r} |p(x_j) - s_j^\varepsilon|^2 w_j, \tag{1.1}$$

where $w_j > 0$ are weights and $\boldsymbol{P}_M$ is the space of trigonometric polynomials of degree less than or equal to M.

Many efficient algorithms have been developed to solve (1.1); e.g., see the articles [22, 7, 25, 11, 10]. But surprisingly little attention has been paid to the problem of how to control the smoothness of the approximation in order to avoid overfitting and underfitting of the data. An adaptation of the smoothness of the approximation can be achieved, for instance, by providing a suitable upper bound for the degree M of the space $\boldsymbol{P}_M$ in (1.1). In most of the aforementioned algorithms a necessary requirement to get useful results in applications is that a good a priori guess of the degree of the

*Received by the editors October 27, 1997; accepted for publication (in revised form) May 15, 2000; published electronically October 18, 2000.
http://www.siam.org/journals/sisc/22-4/32925.html
†Department of Mathematics, University of California, Davis, CA 95616-8633 (strohmer@math.ucdavis.edu). The author was supported by project S7001-MAT, Schrödinger fellowship J01388-MAT of the Austrian Science Foundation FWF, and NSF DMS grant 9973373.

trigonometric approximation is available. However, a priori it is not clear what is a suitable degree for the solution, in terms of how to choose a reasonable degree M when solving (1.1). Determining M by "trial and error" is certainly not a satisfactory alternative.

It is the goal of this paper to derive an efficient algorithm that computes the trigonometric approximation that provides the "optimal" balance between fitting the given data and preserving smoothness of the solution. Here optimality is meant in the sense that the solution has minimal degree among all trigonometric polynomials that satisfy a certain least squares criterion. The algorithm recursively adapts to the least squares approximation of optimal degree by solving a family of nested Toeplitz systems in at most $\mathcal{O}(Mr + M^2)$ operations.

If the data $\{s_j^\varepsilon\}_{j=1}^r$ (i) were unperturbed and (ii) stem from sampling a trigonometric polynomial (with degree less than $r/2$), then the solution of (1.1) would automatically have the appropriate degree, since the original function could be completely recovered in this case. However, the assumptions (i) and (ii) are rarely met in applications, and controlling the smoothness of the solution becomes essential to avoid overfitting and underfitting of the data. If we choose the upper bound for the degree in (1.1) too large, the solution will almost always take on the maximal possible degree; hence it will be too wiggly and pick up too much noise (overfit); see also Figure 1.1(a)–(b). In the extreme case $2M + 1 = r$ we will get an interpolating polynomial, mostly with strong oscillations and far away from approximating the function between the given samples. On the other hand, if we choose M too small, then the approximation will be very smooth but will poorly fit the given data (underfit). Figure 1.1(c) illustrates this behavior. The "regularized" trigonometric approximation obtained by the algorithm proposed in this paper—which we will refer to as the *Levinson–Galerkin algorithm*—is shown in Figure 1.1(d).

The paper is organized as follows. In sections 2 and 4 we present the main results, including the Levinson–Galerkin algorithm and a theoretical analysis that clarifies under which conditions this algorithm provides optimal results. In section 3 we show how properly chosen weights can be used as a simple but efficient tool to precondition the least squares problem. Some aspects of extending the algorithm to multivariate trigonometric polynomials are discussed in section 5. In section 6 we apply the proposed algorithm to a problem in echocardiography.

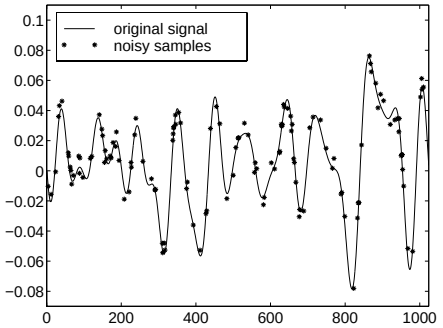
Before we proceed we introduce some notation and conventions. The inner product is denoted by $\langle \cdot, \cdot \rangle$, and the conjugate transpose of a matrix A by A^*. The space of trigonometric polynomial of degree equal to or less than M is defined as

$$\boldsymbol{P}_M = \left\{ p : p(x) = \sum_{k=-M}^{M} c_k e^{2\pi i k x} \right\}. \tag{1.2}$$

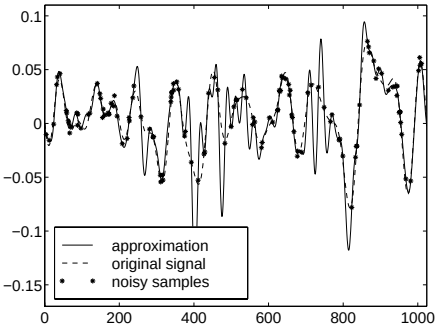
The norm of $p(x) = \sum_{k=-M}^{M} c_k e^{2\pi i k x} \in \boldsymbol{P}_M$ is given by

$$\|p\| = \left(\int_0^1 |p(x)|^2 \, dx \right)^{\frac{1}{2}} = \left(\sum_{k=-M}^{M} |c_k|^2 \right)^{\frac{1}{2}} = \|c\|, \tag{1.3}$$

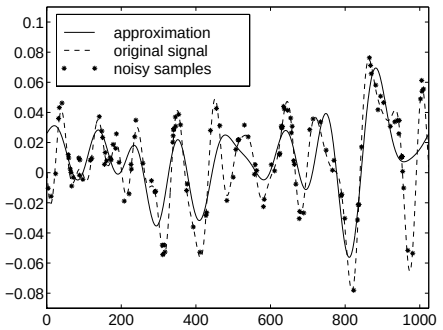
where $c = \{c_k\}_{k=-M}^{M}$. In some applications it is advantageous to deal with complex-valued polynomials (see also section 6); hence we do not restrict ourselves to the case of real-valued trigonometric approximation.



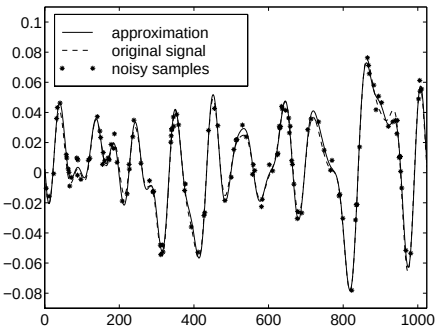
(a) Original signal and perturbed samples



(b) Least squares approximation using a too large upper bound for the degree of the polynomial (overfit)



(c) Least squares approximation using a too small upper bound for the degree of the polynomial (underfit)



(d) Regularized approximation by proposed Levinson–Galerkin algorithm

FIG. 1.1. *Controlling the smoothness of the solution is essential for trigonometric approximation from perturbed data in order to avoid overfitting and underfitting of the data. The proposed Levinson–Galerkin algorithm automatically adapts to the least squares solution of optimal degree.*

For $a = [a_{-M}, a_{-M+1}, \ldots, a_{M-1}, a_M] \in \mathbb{C}^{2M+1}$ we define the orthogonal projections P_N by

$$P_N a = [0, \ldots, 0, a_{-N}, a_{-N+1}, \ldots, a_{N-1}, a_N, 0, \ldots, 0] \tag{1.4}$$

for $N = 1, 2, \ldots, M$ and identify the image of P_N with the $2N+1$-dimensional space $\mathbb{C}^{2N+1}$.

Let p^M and p^N be trigonometric polynomials of degree M and N, respectively, with coefficient vectors $c^M \in \mathbb{C}^{2M+1}$, $c^N \in \mathbb{C}^{2N+1}$. If $N < M$, then we can always interpret p^N as a polynomial of degree M by adding an appropriate number of zero coefficients, and by doing so we are embedding the vector c^N into a zero-padded vector of length $2M+1$. We will henceforth tacitly assume that such an embedding has been made, when we compute expressions such as $\|c^M - c^N\|$.

2. Multilevel least squares approximation. A standard method in numerical analysis to find the optimal balance between fitting the given data and preserving smoothness of the solution is to introduce a regularization parameter. The best value of this regularization parameter is then determined, for instance, by generalized cross validation [15] or via the L-curve [20]. Here we understand regularization not as a way to stabilize ill-conditioned problems, but in a broader context as a means of finding the best compromise between fitting a given set of data and preserving smoothness of the solution. As we will see, in our case it is not necessary to introduce an additional parameter, since we can regularize the smoothness of the solution by varying the parameter M of the space $\boldsymbol{P}_M$ in which we are searching for the solution of (1.1).

For the derivation of the algorithm we consider first the following situation. Assume $p_*(x) = \sum_{k=-N_*}^{N_*} (c_*)_k e^{2\pi i k x} \in \boldsymbol{P}_{N_*}$, and let $s^\varepsilon = \{s_j^\varepsilon\}_{j=1}^r, 2M+1 \le r$ with $s_j^\varepsilon = s_j + \nu_j = p_*(x_j) + \nu_j$ be given noisy samples satisfying

$$\|s^\varepsilon - s\|^2 \le \varepsilon \|s^\varepsilon\|^2. \tag{2.1}$$

For convenience we assume that r, the number of samples, is odd.

The aim is to approximate p_* from the data $\{s_j^\varepsilon\}_{j=1}^r$. Let us first assume that we already know that we are searching for our least squares solution in the space $\boldsymbol{P}_{N_*}$. In this case the coefficient vector of the polynomial that solves (1.1) is the least squares solution of

$$W V_{N_*} c = W s^\varepsilon\,, \tag{2.2}$$

where V_{N_*} is an $r \times (2N_*+1)$ Vandermonde matrix with entries

$$(V_M)_{j,k} = e^{2\pi i k x_j}, \quad j = 1, \dots, r, \quad k = -N_*, \dots, N_*, \tag{2.3}$$

and $W = \operatorname{diag}(\{\sqrt{w_j}\})$.

We will discuss the role and specific choice of the weights in more detail in section 3. To reduce the notational burden we absorb the weight matrix W in the Vandermonde matrix and in the sampling values. Thus for given degree, M, say, we consider the linear system of equations

$$V_M c = s^\varepsilon, \tag{2.4}$$

where V_M is now the $r \times (2M+1)$ "weighted Vandermonde" matrix. We will denote the least squares solution of (2.4) by $c^{(M)} = \{c_k^{(M)}\}_{k=-M}^M$, and the corresponding polynomial is $p^{(M)}(x) = \sum_{k=-M}^M c_k^{(M)} e^{2\pi i k x}$.

Since in general we do not know the optimal degree or *level* M of the space $\boldsymbol{P}_M$ in which we should solve the least squares problem, the situation becomes considerably more complicated. If we want to solve (1.1) under the information (2.1) without knowing the degree of the polynomial, one may argue that we have to accept any trigonometric polynomial $p(x) = \sum_{k=-N}^N c_k e^{2\pi i k x}$ with $\|V_N c - s^\varepsilon\| \le \varepsilon \|s^\varepsilon\|$ as an approximate solution to p_*, since it is compatible with the only knowledge we have on the data.

In general there may be infinitely many such polynomials, which raises the questions of how to find a polynomial p that yields a small approximation error $\|p_* - p\|$ and at the same time can be computed efficiently.

2.1. A multilevel algorithm and an efficient stopping criterion. The heuristic considerations above suggest the following approach.

ALGORITHM 1. *Set $N = 0$ and solve $V_0 c^{(0)} = s^\varepsilon$. If $c^{(0)}$ satisfies the condition $\|V_0 c^{(0)} - s^\varepsilon\| \le \varepsilon \|s^\varepsilon\|$, take $c^{(0)}$ as the solution. Otherwise set $N = N + 1$ and solve*

$$V_N c^{(N)} = s^\varepsilon \,, \tag{2.5}$$

until $c^{(N)}$ satisfies for the first time the stopping criterion

$$\|V_N c^{(N)} - s^\varepsilon\| \le \varepsilon \|s^\varepsilon\| \tag{2.6}$$

at some level $N = N_0$. Set $c^{(N_0)} = c^{(N)}$. The approximation to p_* is then $p^{(N_0)}(x) = \sum_{k=-M}^{M} c_k^{(N_0)} e^{2\pi i k x}$.

The stopping criterion (2.6) is well defined, since it is definitely satisfied for $N = (r-1)/2$, in which case the left side of (2.6) equals 0. Thus Algorithm 1 selects among all least squares solutions $p^{(N)}$ with degree $N = 0, \dots, (r-1)/2$, that polynomial with minimal degree N_0.

Algorithm 1 and stopping criterion (2.6) can be justified by the following theoretical considerations.

One readily verifies that the matrices $V_N, N = 0, \dots, (r-1)/2$, satisfy the following relations:

(i) There exists a left inverse V_N^+ such that

$$V_N^+ V_N = I_N \quad \text{with } V_N^+ = (V_N^* V_N)^{-1} V_N^*, \tag{2.7}$$

where I_N is the identity matrix on $\mathbb{C}^{2N+1}$.

(ii) Let $a \in \mathbb{C}^{2M+1}$ be the coefficient vector of some $p \in \boldsymbol{P}_M$. Then

$$V_N a = V_M a \qquad \text{for all } N > M,\ a \in \mathbb{C}^{2M+1}. \tag{2.8}$$

In (ii) we have made use of the fact that the coefficient vector a can be interpreted as a coefficient vector of a polynomial of degree N by extending it to a vector of length $2N+1$ via zero padding. The matrix–vector multiplication $V_M a$ and (2.8) should be understood in this sense.

LEMMA 2.1. *If $N \ge N_*$, then $c^{(N)}$ satisfies $\|V_N c^{(N)} - s^\varepsilon\| \le \varepsilon \|s^\varepsilon\|$; hence the stopping criterion* (2.6) *always becomes active at some level $N_0 \le N_*$.*

Proof. Note that $V_N V_N^+$ is the orthogonal projection into range(V_N) and $s \in$ range$(V_{N_*}) \subseteq$ range(V_N) for $N_* \le N$; hence $V_N V_N^+ s = s$. Therefore

$$\|V_N c^{(N)} - s^\varepsilon\|^2 = \|V_N V_N^+ (s+\nu) - (s+\nu)\|^2 = \|\nu - V_N V_N^+ \nu\|^2 \tag{2.9}$$

$$= \|\nu\|^2 - \|V_N V_N^+ \nu\|^2 \le \varepsilon^2 \|s^\varepsilon\|^2, \tag{2.10}$$

where we have used condition (2.1) in the last step. It follows from (2.10) that Algorithm 1 terminates at some level $N_0 \le N_*$. ∎

The following lemma shows that from the viewpoint of numerical stability it is advisable to keep the level N of the space $\boldsymbol{P}_N$ in which we search for our solution as small as possible.

LEMMA 2.2. cond$(V_N^* V_N) \ge$ cond$(V_M^* V_M)$ *for $N \ge M$.*

Proof. Since

$$P_M (V_N^* V_N) P_M = V_M^* V_M \qquad \text{for } M \le N,$$

Cauchy's interlace theorem [16] implies that $\operatorname{cond}(V_N^* V_N) \geq \operatorname{cond}(V_M^* V_M)$ for $N \geq M$. □

In what follows we demonstrate that the fact that Algorithm 1 terminates at some level $\leq N_*$ is a desired property in many cases. We show that stopping criterion (2.6) is even optimum in a number of cases.

Let us first consider two special cases: (i) noisefree samples and (ii) uniformly spaced samples.

2.1.1. Noisefree samples. Any reasonable stopping criterion has to satisfy the following lemma.

LEMMA 2.3. *For noisefree data the stopping criterion* (2.6) *yields the exact solution.*

Proof. One readily verifies that Algorithm 1 terminates at level N_*. Hence for $N = N_*$,

$$\|p^{(N)} - p_*\| = \|c^{(N)} - c_*\| = \|V_N^+ s^\varepsilon - c_*\| = \|V_N^+ V_N c_* - c_*\| = 0, \tag{2.11}$$

since $V_N^+ V_N c_* = c_*$ for $N \geq N_*$. □

Lemmas 2.2 and 2.3 together show that stopping criterion (2.6) yields the optimum solution for noisefree data while providing maximum numerical stability.

2.1.2. Uniformly spaced samples. If the sampling points $x_j, j = 1, \ldots, r$, are uniformly spaced and we choose $w_j = 1/r$ as weights, then a simple calculation shows that V_N is unitary on $\mathbb{C}^{2N+1}$, i.e., $V_N^* V_N = I_N$ for $N = 0, 1, \ldots, (r-1)/2$.

In this case

$$\|c_* - c^{(N)}\| = \|c_* - V_N^* s^\varepsilon\| = \|c_* - V_N^* V_{N_*} c_* - V_N^* \nu\|. \tag{2.12}$$

$N \geq N_*$ implies $V_N^* V_{N_*} = I_N$ and hence

$$\|c_* - c^{(N)}\| = \|V_N^* \nu\|. \tag{2.13}$$

Note that

$$\|V_N^* \nu\| = \langle V_N^* \nu, V_N^* \nu \rangle = \langle V_N V_N^* \nu, \nu \rangle = \|V_N V_N^* \nu\|, \tag{2.14}$$

since $V_N V_N^*$ is an orthogonal projection. Equation (2.14) yields

$$\|V_M^* \nu\| \leq \|V_N^* \nu\| \qquad \text{for } M \leq N. \tag{2.15}$$

Consequently

$$\|c_* - c^{(M)}\| \leq \|c_* - c^{(N)}\| \qquad \text{if } N_* \leq M \leq N. \tag{2.16}$$

Thus for uniformly spaced samples any stopping criterion should terminate Algorithm 1 at the latest at $N = N_*$. Under a mild condition on the coefficients c_* we can show that the proposed stopping criterion provides the optimal solution among all least squares solutions.

PROPOSITION 2.4. *Assume that the samples are regularly spaced. Then the solution* $p^{(N_0)}$ *computed via Algorithm* 1 *satisfies*

$$\|p_* - p^{(N_0)}\| \leq \|p_* - p^{(N)}\| \qquad \textit{for all } N \geq N_*. \tag{2.17}$$

If, furthermore, p_* satisfies

$$\|(I_{N_*} - P_N)c_*\| \geq \|(I_{N_*} - P_N)V_{N_*}^*\nu\|, \tag{2.18}$$

then

$$\|p_* - p^{(N_0)}\| \leq \|p_* - p^{(N)}\| \qquad \textit{for all } N. \tag{2.19}$$

Condition (2.18) is satisfied, e.g., if all coefficients of p_* are larger than the relative noise level, i.e., $|c_k| \geq \varepsilon\|s^\varepsilon\|$.

Proof. Lemma 2.1 yields that $N_0 \leq N_*$; thus (2.16) implies (2.17).

To prove assertion (2.19) we have only to show that

$$\|c_* - c^{(N_0)}\| \leq \|c_* - c^{(N)}\| \qquad \text{for all } N < N_*.$$

For $N < N_*$ note that $(c_* - V_N^* V_{N_*} c_*)$ is orthogonal to $V_N^*\nu$, since

$$\langle c_*, V_N^*\nu\rangle = \langle V_{N_*}^* V_{N_*} c_*, V_N^*\nu\rangle \tag{2.20}$$

$$= \langle V_{N_*} c_*, V_{N_*}^* V_N^*\nu\rangle = \langle V_{N_*} c_*, V_N V_N^*\nu\rangle; \tag{2.21}$$

hence

$$\langle c_* - V_N^* V_{N_*} c_*, V_N^*\nu\rangle = 0.$$

Therefore

$$\|c_* - c^{(N)}\|^2 = \|c_* - V_N^+ V_{N_*} c_* + V_N^*\nu\|^2 = \|c_* - V_N^+ V_{N_*} c_*\|^2 + \|V_N^*\nu\|^2 .$$

In order to prove $\|c_* - c^{(N_0)}\| \leq \|c_* - c^{(N)}\|$ for all $N < N_*$ we need to verify $\|c_* - V_N^+ V_{N_*} c_*\|^2 + \|V_N^*\nu\|^2 \geq \|V_{N_*}^*\nu\|^2$. Since

$$\|c_* - V_N^+ V_{N_*} c_*\|^2 = \sum_{|k|=N+1}^{N_*} |(c_*)_k|^2 = \|(I_N - P_N)c_*\|^2$$

and

$$\|(V_{N_*})^*\nu\|^2 - \|V_N^*\nu\|^2 = \sum_{|k|=N+1}^{N_*} |(V_{N_*}^*\nu)_k|^2 = \|(I_N - P_N)V_{N_*}^*\nu\|^2,$$

the result follows now from the assumption (2.18). ∎

Remark. Proposition 2.4 shows that the least squares polynomial that gives the best approximation to p_* is not necessarily of degree N_*.

2.1.3. Noisy nonuniform samples. For noisy nonuniformly spaced data we observe that

$$\|p_* - p^{(N)}\| = \|c_* - c^{(N)}\| \leq \|c_* - V_N^+ V_{N_*} c_*\| + \|V_N^+\nu\|,$$

and for $N \geq N_*$,

$$\|p_* - p^{(N)}\| \leq \|V_N^+\nu\|, \tag{2.22}$$

since $\|c_* - V_N^+ V_{N_*} c_*\| = 0$ in this case.

If V_N is not unitary then $\|V_N^+ \nu\|$ is not necessarily monotonically increasing with increasing level N. One can argue heuristically that since $\|V_N^+\|$ is increasing with increasing level N due to Lemma 2.2, we may fairly assume that $\|V_N^+ \nu\|$ will also increase (although not strictly monotonically). Also from the viewpoint of numerical stability it is reasonable to keep the level N small, since by Lemma 2.2 we know that $\mathrm{cond}(V_N^* V_N) \geq \mathrm{cond}(V_M^* V_M)$ for $N \geq M$. This together with (2.22) suggests that we choose a stopping criterion that terminates at or before level N_*, which is guaranteed for stopping criterion (2.6) by Lemma 2.1.

We can conclude that the stopping criterion will provide excellent results if the noise level ε is small or if the condition number of $V_N^* V_N$ is small (which implies that V_N is approximately unitary). In order to verify the latter it is useful to have estimates for the condition number of $V_N^* V_N$. We will address this issue in Proposition 3.1.

2.2. A Toeplitz system and trigonometric approximation. Instead of directly solving $V_M c^{(M)} = s^\varepsilon$ it is more efficient in our case to consider the normal equations

$$V_M^* V_M^* c^{(M)} = V_M^* s^\varepsilon. \tag{2.23}$$

The reason is that from a numerical point of view the structural properties of the matrix $V_M^* V_M$ are much more attractive than those of V_M, which in turn leads to faster numerical algorithms; see also section 4.

Set $T_M = V_M^* V_M$; then a simple calculation shows that the entries of the hermitian matrix T_M are

$$(T_M)_{k,l} = \sum_{j=1}^{r} w_j e^{2\pi i (k-l) x_j}, \qquad k, l = -M, \ldots, M. \tag{2.24}$$

T_M is a Toeplitz matrix, since the entries $(T_M)_{k,l}$ depend only on the difference $k-l$. Obviously T_M is invertible if $2M+1 \leq r$.

The following result is just a reformulation of (2.23) together with relation (2.8), but since it plays a key role in section 4 it is helpful to state it in detail (cf. also [18]).

THEOREM 2.5. *Given the sampling points $0 \leq x_1 < \cdots < x_r < 1$, the samples $\{s_j^\varepsilon\}_{j=1}^N$, positive weights $\{w_j\}_{j=1}^r$, and the degree M with $2M+1 \leq r$. The polynomial $p^{(M)} \in \boldsymbol{P}_M$ that solves* (1.1) *is given by*

$$p^{(M)}(x) = \sum_{k=-M}^{M} c_m^{(M)} e^{2\pi i k x} \in \boldsymbol{P}_M, \tag{2.25}$$

where its coefficients $c_k^{(M)}$ satisfy

$$T_M c^{(M)} = b^{(M)} \ \in \mathbb{C}^{(2M+1)^2}, \tag{2.26}$$

with

$$b_k^{(M)} = \sum_{j=1}^{r} s_j^\varepsilon w_j e^{2\pi i k x_j} \qquad \text{for } |k| \leq M, \tag{2.27}$$

and T_M as defined in (2.24).

3. Weights as simple preconditioner. Vandermonde matrices are known to be ill conditioned if the nodes x_j are clustered [13]. To improve the stability of systems (2.4) and (2.26) we can use the weights as a simple diagonal preconditioner. This leads to the problem of how to choose the weights w_j.

We propose to use the size of the area of the Voronoi region [23] associated with the sampling point x_j as weight w_j. In one dimension this reduces to

$$w_j = \frac{x_{j+1} - x_{j-1}}{2}\,. \tag{3.1}$$

This choice is motivated by the following observations.

In this section we let V_N denote the Vandermonde matrix defined in (2.3) without weights. Let $p \in \boldsymbol{P}_M$ with coefficient vector c. Since

$$\langle T_M c, c\rangle = \langle WV_M c, WV_M c\rangle = \|WV_M c\|^2 = \sum_{j=1}^{r} |p(x_j)|^2 w_j, \tag{3.2}$$

the inequality

$$C_1\|c\|^2 \le \sum_{j=1}^{r} |p(x_j)|^2 w_j \le C_2\|c\|^2 \tag{3.3}$$

holds for all $c \in \mathbb{C}^{2M+1}$ with constants $C_1 = \lambda_{\min}$ and $C_2 = \lambda_{\max}$, where $\lambda_{\min}$ and $\lambda_{\max}$ denote the minimal and maximal eigenvalues, respectively, of T_M.

(i) The lower bound of T_M is mainly determined by the large gaps in the sampling set. Suppose there is a large gap in the sampling set and denote the interval corresponding to this gap by Γ (hence $x_j \notin \Gamma$). Choose a trigonometric polynomial $p \in \boldsymbol{P}_M$, which, like the prolate spheroidal functions, concentrates most of its energy in the interval Γ. Then the sampling values of p will not pick up any information about the main concentration of the polynomial energy. Consequently, if we use no weights (or set $w_j = 1$), we get

$$\sum_{j\notin\Gamma} |p(x_j)|^2 \ll \|p\|^2 = \|c\|^2.$$

For such a sampling set the lower frame bound C_1 in the inequality (3.3) must be small. Generically, large gaps and the ensuing lack of information always result in bad condition numbers. This problem cannot be fixed by preconditioning.

(ii) On the other hand, we can choose a trigonometric polynomial that is mainly concentrated in the region where the sampling points are located. In this case the same local information is counted and added several times. Thus

$$\sum_{j\notin\Gamma} |p(x_j)|^2 \gg \|p\|^2 = \|c\|^2$$

and the upper constant C_2 in (3.3) will be large. Yet, as mentioned in (i), a cluster will not contribute much to the lower bound and to the uniqueness of the problem. In this case the condition number is large, because too much local information is given in certain areas of the polynomial.

Problem (ii) can be addressed by introducing properly chosen weights. The idea is to compensate for the local variation of the sampling density by using weights in

inequality (3.3). Suppose that $0 \le x_1 \le x_2 \le \cdots \le x_r < 1$ is a sampling set in $[0,1]$. Then a natural choice for the weights is $w_j = (x_{j+1} - x_{j-1})/2$. Thus if many samples are clustered near a point x_j, then the weight w_j is small. If x_j is the only sampling point in a large neighborhood, then the corresponding weight is large. This choice has not only been confirmed by extensive numerical experiments [11] but also by the following optimization approach.

A standard approach for the construction of preconditioners for a matrix A is the following. One attempts to find the matrix P in a given class $\mathcal{M}$ of matrices (e.g., the class of all circulant matrices or the class of all diagonal matrices) that solves

$$\min_{P \in \mathcal{M}} \|I - PA\|_F \,, \tag{3.4}$$

where $\|.\|_F$ denotes the Frobenius norm.

In our setting this translates into the optimization problem

$$\min_{W \in \mathcal{D}} \|I - (WV)^* WV\|_F \,, \tag{3.5}$$

where $\mathcal{D}$ is the class of all $r \times r$ diagonal matrices and I is the $(2M+1) \times (2M+1)$ identity matrix.

Note that we require that $w_j > 0$, whereas (3.5) could in principle yield weights that violate this condition. However, since we will make use of (3.5) in our actual algorithm, we are somewhat sloppy here.

An alternative approach is to consider the solution of

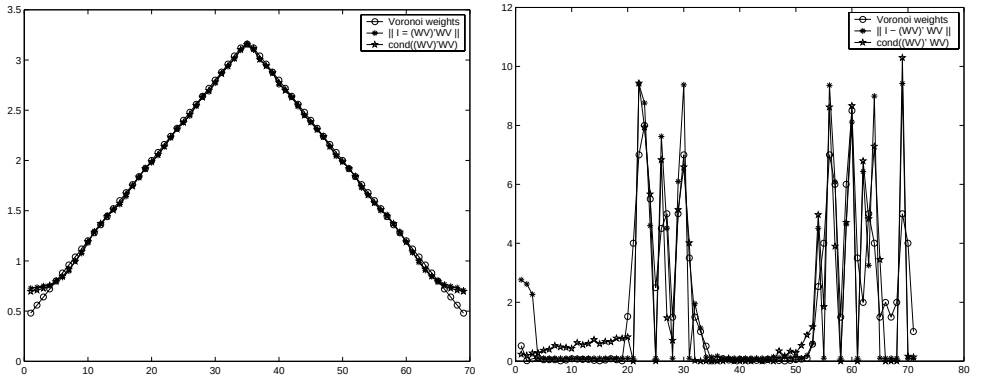
$$\begin{gathered} \min\{\mathrm{cond}[(WV)^* WV]\} \\ \text{subject to } W \in \mathcal{D}. \end{gathered} \tag{3.6}$$

This optimization problem can be transformed to a general eigenvalue problem (see [4]), which can be solved by convex optimization algorithms.

In the simple case of regular sampling it is easy to check that the solution of both optimization problems is given by $W = \mathrm{diag}(\{\sqrt{w_j}\})$ with $w_j = (x_{j+1} - x_{j-1})/2 = 1/r$. However, in the more interesting case of nonuniform sampling neither problem (3.5) nor (3.6) in general has an analytic solution. Thus, using these approaches for the actual construction of a preconditioner would be ridiculous, since the computational costs to solve these optimization problems are considerably larger than solving the trigonometric approximation problem. Nevertheless, solving (3.5) and (3.6) numerically for a variety of different examples is useful to gain insight into the type of weights obtained by these approaches.

The numerical results confirm the choice of the Voronoi-type weights defined in (3.1). Sampling points in densely sampled areas are assigned a small weight, whereas sampling points in sparse sampled regions are assigned a large weight. Two typical comparisons of the weights obtained via optimization and the Voronoi weights are illustrated in Figure 3.1. In the first case we consider a sampling set with high density at the endpoints and strongly decreasing density toward the center. The weights obtained by solving (3.5) and (3.6) are almost identical and are very close to the Voronoi weights, as can be seen in Figure 3.1(a). The difference at the endpoints is probably due to boundary effects.

In the second example we consider a random sampling set with several areas with high sampling density and relatively few samples between these clusters. Again all three approaches give weights that show a similar behavior; see Figure 3.1(b). The



(a) (b)

FIG. 3.1. *Comparison of weights obtained by different approaches.*

condition number of the nonweighted Toeplitz matrix in this example is 33, compared with the significantly smaller condition number 3.3 when using Voronoi-type weights. Using the weights obtained via (3.5) gives $\mathrm{cond}(T_M) = 3.1$, and for the weights resulting from (3.6) we get $\mathrm{cond}(T_M) = 2.9$, which is only a slight improvement compared with the Voronoi-type weights.

Obtaining good estimates for the condition number of a Toeplitz matrix is a difficult problem. It is gratifying that by using the weights defined in (3.1) it is possible to get an upper bound for the condition number.

PROPOSITION 3.1 (Gröchenig [18]). *Assume that the sampling set* $\{x_j\}_{j=1}^r$ *satisfies*

$$\max(x_{j+1} - x_j) := \gamma < \frac{1}{2M}$$

and set $w_j = (x_{j+1} - x_{j-1})/2$. *Then the condition number of the Toeplitz matrix* T_M *defined in* (2.24) *is bounded by*

$$\kappa(T_M) \leq \left(\frac{1+\gamma}{1-\gamma}\right)^2. \tag{3.7}$$

4. A Levinson–Galerkin algorithm for trigonometric approximation. The method described in Algorithm 1 can be seen as a Galerkin-type approach, since we try to determine an approximation by searching for a solution in a finite-dimensional space spanned by orthogonal polynomials, and by increasing the dimension of the space we increase the resolution of our approximation by adding more and more details.

When we use Levinson's algorithm [16] to solve (2.26) for $M = 0, 1, \ldots, N_0$ the total computational effort would be of $\mathcal{O}(N_0^3)$, since the solution of each system $T_M c^{(M)} = b^{(M)}$ requires $\mathcal{O}(M^2)$ operations. Using one of the fast Toeplitz algorithms [2, 5] reduces this effort to $\mathcal{O}(kM \log M)$ for each level M, where k is the number of iterations, thus leading to a total of $\mathcal{O}(kN_0^2 \log N_0)$ operations. In this section we show that the systems $T_M c^{(M)} = b^{(M)}, M = 0, 1, \ldots, N_0$ can be solved in

$\mathcal{O}(N_0^2)$ operations and the total effort (including the calculation of the entries of T_M and the evaluation of the stopping criterion (2.6)) for computing $p^{(N_0)}$ is $\mathcal{O}(rN_0+N_0^2)$ operations.

The following observation is crucial for the derivation of the proposed Levinson–Galerkin algorithm.

LEMMA 4.1. *For fixed degree M and $M+1$ let $T_M, b^{(M)}$ and $T_{M+1}, b^{(M+1)}$ be the Toeplitz matrices and right-hand sides as defined in* (2.24) *and* (2.27)*, respectively. Then T_M and $b^{(M)}$ are embedded in T_{M+1} and $b^{(M+1)}$ in the following way:*

$$T_{M+1} = \begin{bmatrix} t_0 & \dots & \bar{t}_{2(M+1)} \\ \vdots & \boxed{T_M} & \vdots \\ t_{2(M+1)} & \dots & t_0 \end{bmatrix}, \quad b^{(M+1)} = \begin{bmatrix} b_{-(M+1)} \\ b^{(M)} \\ b_{M+1} \end{bmatrix}. \tag{4.1}$$

Proof. Equation (4.1) follows immediately from the definition of T_M and $b^{(M)}$ and (2.8). ∎

Unfortunately the solutions $c^{(M)}$ and $c^{(M+1)}$ of the systems $T_M c^{(M)} = b^{(M)}$ and $T_{M+1}c^{(M+1)} = b^{(M+1)}$ are not related in such a simple manner. But we can exploit the nested structure of the family $\{T_M\}_{M=1}^{N_0}$ by solving the systems $T_M c^{(M)} = b^{(M)}$ recursively via a modified Levinson algorithm. The standard Levinson algorithm cannot be applied directly, since it only addresses Toeplitz systems, where the principal leading submatrix and the principal leading subvector of the right-hand side stay unchanged during the recursion, which is not the case here. For T_{M+1} it does not matter if we enlarge T_M by appending new entries below or above, whereas the right-hand side $b^{(M)}$ cannot be rearranged in such a way that the principal leading subvector of the right-hand side will be changed if we switch from $b^{(M)}$ at level M to $b^{(M+1)}$ at level $M+1$.

To adapt Levinson's algorithm to our situation, we have to split up the change from the system $T_M c^{(M)} = b^{(M)}$ at level M to the system $T_{M+1}c^{(M+1)} = b^{(M+1)}$ at level $M+1$ into two separate steps. Instead of indexing the matrix T_M and the vectors $b^{(M)}, c^{(M)}$ by the degree M, it is therefore advantageous to index them according to their dimension. For clarity of presentation we reserve the subscript (M) for the degree of the polynomial and its coefficient vector, respectively, and use the subscript (ℓ) when we refer to the dimension of the corresponding coefficient vector in $\mathbb{C}^\ell$. Thus for even ℓ, $b^{(\ell)} = [b_{-\frac{\ell}{2}+1}, \dots, b_{\frac{\ell}{2}}]^T \in \mathbb{C}^\ell$, and for odd ℓ we set $b^{(\ell)} = [b_{-\frac{\ell-1}{2}}, \dots, b_{\frac{\ell-1}{2}}]^T$ (whence $b^{(1)} = b_0$), analogously for $c^{(\ell)}$. Further, it is useful in what follows to denote $t^{(\ell)} = [t_1, \dots, t_\ell]^T$. Then the Toeplitz matrix T_ℓ of size $\ell \times \ell$ is generated by the vector $[t_0, (t^{(\ell-1)})^T]^T$ with $t_k = \sum_{j=1}^r w_j e^{2\pi i k x_j}$ according to (2.24).

Assume we have already solved the system $T_M c^{(M)} = b^{(M)}$ at level M (with $\ell = 2M+1$) and now we want to switch to the next level $M+1$. As we have agreed, we do this in two steps. In the first step ($\ell \to \ell+1$) the Toeplitz system can be written as

$$\begin{bmatrix} T_\ell & E_\ell \overline{t^{(\ell)}} \\ (t^{(\ell)})^T E_\ell & t_0 \end{bmatrix} \begin{bmatrix} v^{(\ell)} \\ v_{\frac{\ell+1}{2}} \end{bmatrix} = \begin{bmatrix} b^{(\ell)} \\ b_{\frac{\ell+1}{2}} \end{bmatrix}, \tag{4.2}$$

where E_ℓ is the rotated identity matrix on $\mathbb{C}^\ell$, i.e.,

$$E_\ell = \begin{bmatrix} 0 & & 1 \\ & \iddots & \\ 1 & & 0 \end{bmatrix}.$$

System (4.2) can be solved recursively by the standard Levinson algorithm [21, 16]. To be more precise, we assume that we have already solved the system $T_\ell c^{(\ell)} = b^{(\ell)}$ for $\ell = 2M+1$ and assume further that the solution of the ℓth-order *Yule–Walker system* $T_\ell y^{(\ell)} = -t^{(\ell)}$ is available. Then the solution of (4.2) can be computed recursively by

$$\begin{aligned} v_{\frac{\ell+1}{2}} &= (b_{\frac{\ell+1}{2}} - [t^{(\ell)}]^T E_\ell c^{(\ell)})/\beta_\ell, \\ v^{(\ell)} &= c^{(\ell)} + v_{\frac{\ell+1}{2}} E_\ell \overline{y^{(\ell)}}, \end{aligned}$$

where

$$\begin{aligned} \beta_\ell &= t_0 + [t^{(\ell)}]^T \overline{y^{(\ell)}} = (1 - \alpha_{\ell-1}\overline{\alpha_{\ell-1}})\beta_{\ell-1}, \\ \alpha_\ell &= -(t_{\ell+1} + [t^{(\ell)}]^T E_\ell y^{(\ell)})/\beta_\ell, \\ z^{(\ell)} &= y^{(\ell)} + \alpha_\ell E_\ell \overline{y^{(\ell)}}, \\ y^{(\ell+1)} &= \begin{bmatrix} z^{(\ell)} \\ \alpha_\ell \end{bmatrix}. \end{aligned}$$

Now we can proceed to the second step ($\ell + 1 \to \ell + 2 = 2(M+1)+1$), where the Toeplitz system can be expressed as

$$\begin{bmatrix} t_0 & (t^{(\ell+1)})^* \\ t^{(\ell+1)} & T_{\ell+1} \end{bmatrix} \begin{bmatrix} v_{-\frac{\ell+1}{2}} \\ v^{(\ell+1)} \end{bmatrix} = \begin{bmatrix} b_{-\frac{\ell+1}{2}} \\ b^{(\ell+1)} \end{bmatrix} \tag{4.3}$$

with $c^{(\ell+2)} = [v_{-\frac{\ell+1}{2}}, (v^{(\ell+1)})^T]^T = c^{(M+1)}$. Observe that (4.3) cannot be transformed to a system of the form (4.2) by simple permutations, i.e., just by interchanging rows and columns. Since we have already solved the systems $T_{\ell+1}c^{(\ell+1)} = b^{(\ell+1)}$ and $T_{\ell+1}y^{(\ell+1)} = -t^{(\ell+1)}$ we can write

$$v^{(\ell+1)} = (T_{\ell+1})^{-1}(b^{(\ell+1)} - t^{(\ell+1)}v_{-\frac{\ell+1}{2}}) = c^{(\ell+1)} + v_{-\frac{\ell+1}{2}} y^{(\ell+1)}$$

and

$$\begin{aligned} v_{-\frac{\ell+1}{2}} =& (b_{-\frac{\ell+1}{2}} - [t^{(\ell+1)}]^* v^{(\ell+1)})/t_0 \\ =& (b_{-\frac{\ell+1}{2}} - [t^{(\ell+1)}]^* c^{(\ell+1)} - [t^{(\ell+1)}]^* v_{-\frac{\ell+1}{2}} y^{(\ell+1)})/t_0 \\ =& (b_{-\frac{\ell+1}{2}} - [t^{(\ell+1)}]^* c^{(\ell+1)})/\beta_{\ell+1}, \end{aligned}$$

where we have used in the last step that $T_\ell = [T_\ell]^*$, which implies that $(t^{(\ell+1)})^* y^{(\ell+1)}$ is real and therefore $t_0 + (t^{(\ell+1)})^* y^{(\ell+1)} = t_0 + (t^{(\ell+1)})^T \overline{y^{(\ell+1)}} = \beta_{\ell+1}$.

Note that at each level M we have to check if the stopping criterion (2.6) is satisfied. The evaluation of the expression

$$\sum_{j=1}^{r} |p^{(M)}(x_j) - s_j^\varepsilon|^2 w_j \tag{4.4}$$

can be considerably simplified and by avoiding the evaluation of $p^{(M)}$ at the nonuniformly spaced points x_j we can reduce the computational effort from $\mathcal{O}(Mr)$ to $\mathcal{O}(M)$ operations.

To do this we define the subspace $\mathcal{R} = \left\{\{p(x_j)\}_{j=1}^r : p \in \boldsymbol{P}_M\right\} \subseteq \mathbb{C}^r$ with the weighted inner product $\langle y, z\rangle_{\mathcal{R}} = \sum_{j=1}^r y_j \bar{z}_j w_j$ for $y, z \in \mathbb{C}^r$. The solution of the least

squares problem (1.1) is the orthogonal projection of the vector $\{s_j^\varepsilon\}_{j=1}^r \in \mathbb{C}^r$ onto $\mathcal{R}$ and therefore must satisfy

$$\langle \{p^{(M)}(x_j)\} - \{s_j^\varepsilon\}, \{p^{(M)}(x_j)\}\rangle_{\mathcal{R}} = \sum_{j=1}^r (p^{(M)}(x_j) - s_j^\varepsilon)\overline{p^{(M)}(x_j)} w_j = 0,$$

which implies

$$\langle \{p^{(M)}(x_j)\}, \{s_j^\varepsilon\}\rangle_{\mathcal{R}} = \langle \{p^{(M)}(x_j)\}, \{p^{(M)}(x_j)\}\rangle_{\mathcal{R}}\,. \tag{4.5}$$

Since

$$\begin{aligned}\sum_{j=1}^r |s_j^\varepsilon - p^{(M)}(x_j)|^2 w_j &= \sum_{j=1}^r |s_j^\varepsilon|^2 w_j - 2\,\mathrm{Re}\,\langle s^\varepsilon, \{p^{(M)}(x_j)\}\rangle_{\mathcal{R}} + \sum_{j=1}^r |p^{(M)}(x_j)|^2 w_j \\ &= \sum_{j=1}^r |s_j^\varepsilon|^2 w_j - \sum_{j=1}^r |p^{(M)}(x_j)|^2 w_j\end{aligned}$$

by (4.5), and because

$$\begin{aligned}\sum_{j=1}^r |p^{(M)}(x_j)|^2 w_j &= \sum_{j=1}^r w_j \left(\sum_{m=-M}^M c_m^{(M)} e^{2\pi i m x_j}\right)\left(\sum_{n=-M}^M \overline{c_n^{(M)}} e^{2\pi i n x_j}\right) \\ &= \sum_{m=-M}^M \sum_{n=-M}^M c_m^{(M)}\overline{c_n^{(M)}} \left(\sum_{j=1}^r w_j e^{2\pi i (m-n) x_j}\right) \\ &= \langle T_M c^{(M)}, c^{(M)}\rangle = \langle b^{(M)}, c^{(M)}\rangle\,,\end{aligned} \tag{4.6}$$

it follows that

$$\sum_{j=1}^r |s_j^\varepsilon - p^{(M)}(x_j)|^2 w_j = \sum_{j=1}^r |s_j^\varepsilon|^2 w_j - \langle b^{(M)}, c^{(M)}\rangle\,. \tag{4.7}$$

Since $\sum_{j=1}^r |s_j^\varepsilon|^2 w_j$ has to be computed only once at the beginning of the algorithm, the evaluation of (4.4) can be carried out in $\mathcal{O}(M)$ operations.

Summing up we have arrived at the following algorithm to compute $p^{(N_0)}$.

ALGORITHM 2 (Levinson–Galerkin algorithm for trigonometric polynomials). *Let the sampling points $\{x_j\}_{j=1}^r$, sampling values $\{s_j^\varepsilon\}_{j=1}^r$, weights $w_j > 0$, and the data error estimate ε be given. Then the trigonometric polynomial $p^{(N_0)}$ determined in Algorithm 1 can be computed in $\mathcal{O}(rN_0 + N_0^2)$ operations by the following algorithm:*

Initialize: $t_0 = \sum_{j=1}^r w_j, t_1 = \sum_{j=1}^r w_j e^{2\pi i x_j}, b_0 = \sum_{j=1}^r s_j^\varepsilon w_j, \sigma = \sum_{j=1}^r |s_j^\varepsilon|^2 w_j$, $y^{(1)} = -t_1/t_0, c^{(1)} = b_0/t_0, \beta_0 = t_0, \alpha_0 = -t_1/t_0, \varepsilon_1 = (\sigma - b_0^2/t_0)/\sigma, \ell = 1.$

while $\varepsilon_\ell > \varepsilon$

$\beta_\ell = (1 - \alpha_{\ell-1}\overline{\alpha_{\ell-1}})\beta_{\ell-1}$

if $\ell \equiv 1 \mod 2$

$$b_{\frac{\ell+1}{2}} = \sum_{j=1}^r s_j^\varepsilon w_j e^{\pi i(\ell+1)x_j}$$

$$v_{\frac{\ell+1}{2}} = \frac{b_{\frac{\ell+1}{2}} - \langle E_\ell \overline{c^{(\ell)}}, t^{(\ell)}\rangle}{\beta_\ell}$$

$$v^{(\ell)} = c^{(\ell)} + v_{\frac{\ell+1}{2}} E_\ell \overline{y^{(\ell)}}$$

$$c^{(\ell+1)} = \begin{bmatrix} v^{(\ell)} \\ v_{\frac{\ell+1}{2}} \end{bmatrix}$$

$$b^{(\ell+1)} = \begin{bmatrix} b^{(\ell)} \\ b_{\frac{\ell+1}{2}} \end{bmatrix}$$

elseif $\ell \equiv 0 \mod 2$

$$b_{-\frac{\ell}{2}} = \sum_{j=1}^r s_j^\varepsilon w_j e^{-\pi i \ell x_j}$$

$$v_{-\frac{\ell}{2}} = \frac{b_{-\frac{\ell}{2}} - \langle \overline{c^{(\ell)}}, t^{(\ell)}\rangle}{\beta_\ell}$$

$$v^{(\ell)} = c^{(\ell)} + v_{-\frac{\ell}{2}} y^{(\ell)}$$

$$c^{(\ell+1)} = \begin{bmatrix} v_{-\frac{\ell}{2}} \\ v^{(\ell)} \end{bmatrix}$$

$$b^{(\ell+1)} = \begin{bmatrix} b_{-\frac{\ell}{2}} \\ b^{(\ell)} \end{bmatrix}$$

$$\varepsilon_{\ell+1} = |\sigma - \langle b^{(\ell+1)}, c^{(\ell+1)}\rangle|/\sigma$$

end

$$t^{(\ell+1)} = \sum_{j=1}^r w_j e^{2\pi i(\ell+1)x_j}$$

$$\alpha_\ell = -\frac{t^{(\ell+1)} + \langle E_\ell \overline{y^{(\ell)}}, t^{(\ell)}\rangle}{\beta_\ell}$$

$$z^{(\ell)} = y^{(\ell)} + \alpha_\ell E_\ell \overline{y^{(\ell)}}$$

$$y^{(\ell+1)} = \begin{bmatrix} z^{(\ell)} \\ \alpha_\ell \end{bmatrix}$$

$$t^{(\ell+1)} = \begin{bmatrix} t^{(\ell)} \\ t_{\ell+1} \end{bmatrix}$$

$\ell = \ell + 1$

end

$N_0 = \ell/2$

$$p^{(N_0)}(x) = \sum_{k=-N_0}^{N_0} c_k^{(N_0)} e^{2\pi i k x}$$

Remark. Usually one evaluates the final approximation on regularly spaced gridpoints; hence the last step of the algorithm can be realized by a fast Fourier transform (FFT). The most costly steps are the computation of the entries of $t^{(\ell)}$ and $b^{(\ell)}$. According to Corollary 1 in [11] the entries of T_M and $b^{(M)}$ can also be computed via FFT by embedding the x_j into a regular grid (since the x_j can be stored only in finite precision). In this case one automatically gets all entries $t_0, \ldots, t_r$ at once. However, this trick is useful only if the number of points of the regular grid is of the same magnitude as the number of sampling points. Alternatively one may use the numerical attractive formulas of Dutt and Rokhlin [9] or Beylkin [3] for a fast evaluation of trigonometric sums at unequally spaced nodes.

Algorithm 2 can be simplified for real-valued data; this modification is left to the reader.

Fast Vandermonde solvers require $\mathcal{O}(Mr)$ operations for the solution of $V_M c^{(M)} = s^{\varepsilon}$; cf. [25]. It is not clear, however, if these algorithms can utilize the nested structure of the sequence of matrices $\{V_M\}_M$ in order to give rise to an efficient implementation of Algorithm 1. Moreover, it is an open problem if the Vandermonde solvers can be extended to multivariate trigonometric approximation. We will see in the next section that the extension of Algorithm 2 to higher dimensions is straightforward.

5. Multivariate trigonometric approximation. An advantage of the proposed approach, besides its numerical efficiency, is the fact that it can be easily extended to multivariate trigonometric approximation. In this section we briefly discuss some results for the 2-D case.

We define the space of 2-D trigonometric polynomials $\boldsymbol{P}_M^2$ by

$$\boldsymbol{P}_M^2 = \left\{ p : p(x,y) = \sum_{j,k=-M}^{M} c_{j,k} e^{2\pi i (jx+ky)} \right\}. \tag{5.1}$$

To reduce the notational burden, we have assumed in (5.1) that p has degree equal to M in each coordinate; the extension to polynomials with different degree in each coordinate is straightforward.

For an arbitrary sampling set $\{(x_j, y_j)\}_{j=1}^r \in [0,1)^2$ and given degree M the system matrix according to the 2-D version of Theorem 2.5 is [28]

$$(T_M)_{k,l} = \sum_{j=1}^{r} w_j e^{2\pi i (k-l)(x_j+y_j)}, \qquad k,l = 0, \ldots, 2M. \tag{5.2}$$

One can easily verify that T_M is a hermitian block Toeplitz matrix with $2M+1$ different Toeplitz blocks of size $(2M+1) \times 2M+1$; cf. [28]. For a given sampling set let T_M be the block Toeplitz matrix for degree M and T_{M+1} the block Toeplitz matrix for degree $M+1$. There is a similar relationship between T_M and T_{M+1} as in the 1-D case. More precisely, denote the Toeplitz blocks of T_M and T_{M+1} by $(B_M)_k, k = 0, \ldots, 2M$, and $(B_{M+1})_k, k = 0, \ldots, 2M+2$, respectively. Then one readily verifies the following embedding:

$$T_{M+1} = \begin{bmatrix} (B_{M+1})_0 & \cdots & (B_{M+1})_{2(M+1)}^* \\ \vdots & \boxed{T_M} & \vdots \\ (B_{M+1})_{2(M+1)} & \cdots & (B_{M+1})_0 \end{bmatrix}, \tag{5.3}$$

$$B_{M+1} = \begin{bmatrix} t_0 & \dots & \bar{t}_{2(M+1)} \\ \vdots & \boxed{B_M} & \vdots \\ t_{2(M+1)} & \dots & t_0 \end{bmatrix}. \tag{5.4}$$

In [1] Levinson's algorithm has been extended to general block Toeplitz systems. With this extension and relation (5.3) at hand, we can easily generalize Algorithm 2 to 2-D (and along the same lines to multivariate) trigonometric approximation.

The analysis of the stopping criterion (2.6) in section 2 can be applied line by line to the 2-D (actually to the n-D) setting. The only difficulty arises in the search for simple criteria for the invertibility of the block Toeplitz matrix T_M. The condition

$$(2M+1)^d \leq r$$

is necessary in dimension $d > 1$, but is no longer sufficient, since the fundamental theorem of algebra does not hold in dimensions larger than one. In [19] Gröchenig has derived estimates for the condition number of T_M in higher dimensions. In two dimensions these estimates can be stated as follows.

Let $D_\delta(a,b) = \{(x,y) \in \mathbb{R}^2 : (x-a)^2 + (y-b)^2 < \delta^2\}$ be the disc of radius δ centered at (a,b). We say that a set $\{(x_j, y_j), j = 1, \dots, r\}$ is δ-dense in $[0,1] \times [0,1]$, if $\bigcup_{j=1}^r D_\delta(x_j, y_j) \supseteq [0,1] \times [0,1]$. In other words, the distance of a given sample (x_j, y_j) to its nearest neighbor $(x_k, y_k), k \neq j$ is at most 2δ.

Analogously to section 3 we choose the size of the Voronoi region V_j associated with x_j as weight w_j in the computation of the block Toeplitz matrix T_M in (5.2). Suppose that the sampling set $\{(x_j, y_j), j = 1, \dots, r\} \subseteq [0,1] \times [0,1]$ is δ-dense and

$$\delta < \frac{\log 2}{4\pi M}. \tag{5.5}$$

Gröchenig [19] has shown that under these conditions

$$\operatorname{cond}(T_M) \leq \frac{4}{(2 - e^{4\pi M \delta})^2}. \tag{5.6}$$

In particular, for arbitrary δ-dense sampling sets, the block Toeplitz matrix T_M is invertible and the 2-D version of Algorithm 2 is applicable.

5.1. Line-type nonuniform sampling in two dimensions. In the following we consider a special case of trigonometric approximation in two dimensions. This case arises when a function is irregularly sampled along lines. A typical example is illustrated in Figure 5.1. Such sampling patterns are encountered, for instance, in geophysics and medical imaging; see also section 6.2.

COROLLARY 5.1. *Let $p \in \boldsymbol{P}_M^2$ and let $\{x_j, y_{j,k}\}, j = 1, \dots, r, k = 1, \dots, r_j$, be a sampling set in $[0,1)^2$ such that*

$$A_1 \|p\|^2 \leq \sum_{k=1}^{r_j} |p(y_{j,k})|^2 \leq B_1 \|p\|^2, \qquad A_1, B_1 > 0, \tag{5.7}$$

for every $p \in \boldsymbol{P}_M$ and for all j. Further assume that $\{x_j\}_{j \in \mathbb{Z}}$ is a sampling set such that

$$A_2 \|p\|^2 \leq \sum_{j=1}^{r} |p(x_j)|^2 \leq B_2 \|p\|^2, \qquad A_2, B_2 > 0, \tag{5.8}$$

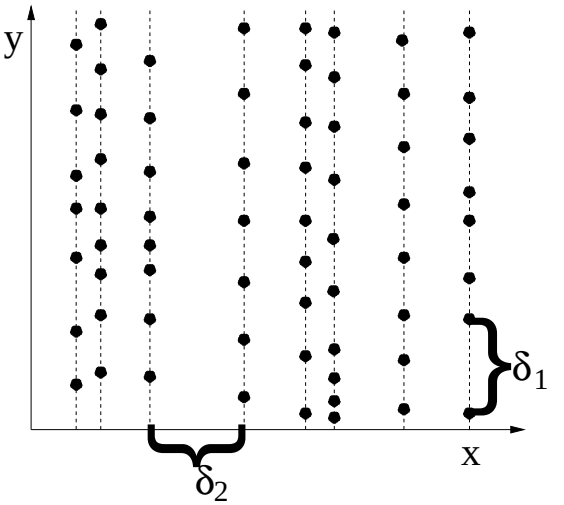


FIG. 5.1. *Line-type nonuniform sampling set.*

for every $p \in \boldsymbol{P}_M$. *Then*

$$A_1 A_2 \|p\|^2 \le \sum_{j=1}^{r} \sum_{k=1}^{r_j} |p(x_j, y_{j,k})|^2 \le B_1 B_2 \|p\|^2 \tag{5.9}$$

for every $p \in \boldsymbol{P}_M^2$.

If $\{x_j\}$ *and* $\{y_{j,,k}\}$ *are sampling sets with* $\sup_k (x_{k+1} - x_k) = \delta_2 < \frac{1}{2M}$ *and* $\sup_{j,k}(y_{j,k+1} - y_{j,k}) = \delta_1 < \frac{1}{2M}$, *then* $A_l = (1-\delta_l)^2, B_l = (1+\delta_l)^2, l = 1, 2$, *and the condition number of the block Toeplitz matrix* T_M *is bounded by*

$$\kappa(T_M) \le \frac{(1+\delta_1)^2(1+\delta_2)^2}{(1-\delta_1)^2(1-\delta_2)^2}\,. \tag{5.10}$$

Proof. Let x be fixed. Then $y \to p(x, y) \in \boldsymbol{P}_M$ and hence for all j,

$$A_1 \int_0^1 |p(x_j, y)|^2\, dy \le \sum_{k=1}^{r_j} |p(x_j, y_{j,k})|^2 \le B_1 \int_0^1 |p(x_j, y)|^2\, dy \tag{5.11}$$

by assumption (5.7). It follows that

$$A_1 \sum_{j=1}^{r} \int_0^1 |p(x_j, y)|^2\, dy \le \sum_{j,k} |p(x_j, y_{j,k})|^2 \le B_1 \sum_{j=1}^{r} \int_0^1 |p(x_j, y)|^2\, dy. \tag{5.12}$$

Now let y be fixed. Then $x \to p(x, y) \in \boldsymbol{P}_M$ and

$$A_2 \int_0^1 |p(x, y)|^2\, dx \le \sum_{j=1}^{r} |p(x_j, y)|^2 \le B_2 \int_0^1 |p(x, y)|^2\, dx\,. \tag{5.13}$$

Since

$$\sum_{j=1}^{r} \int_0^1 |p(x_j, y)|^2\, dy = \int_0^1 \sum_{j=1}^{r} |p(x_j, y)|^2\, dy\,, \tag{5.14}$$

assertion (5.9) follows by combining (5.12) and (5.13) with (5.14). The estimate of the constants A_l, B_l and of the condition number of the block Toeplitz matrix T_M follows from Theorem 2.5. □

The proof of Corollary 5.1 is due to Gröchenig [17]. Corollary 5.1 not only guarantees that $p \in \boldsymbol{P}_M^2$ can be recovered from its samples $p(x_j, y_{j,k})$, it provides more. An immediate consequence is that it can be reconstructed by an efficient algorithm relying on a successive application of Algorithm 2 and the Gohberg–Semencul representation of the inverse of a Toeplitz matrix. See section 6.2 for more details and an application in medical imaging.

6. Curve and surface approximation by trigonometric polynomials. Trigonometric polynomials can be used to model the boundary or the surface of smooth objects. Let us consider a 2-D object, obtained, e.g., by a planar cross section from a 3-D object and assume that the boundary of this 2-D object is a closed curve in $\mathbb{R}^2$. We denote this curve by f and parameterize it by $f(u) = (x_u, y_u)$, where x_u and y_u are the coordinates of f at "time" u in the x- and y-direction, respectively. Obviously we can interpret f as a 1-D continuous, complex, and periodic function, where x_u represents the real part and y_u represents the imaginary part of $f(u)$. It follows from the theorem of Weierstrass (and from the theorem of Stone–Weierstrass [26] for higher dimensions) that a continuous periodic function can be approximated uniformly by trigonometric polynomials. If f is smooth, we can fairly assume that trigonometric polynomials of low degree provide an approximation of sufficient precision.

Assume that we know only some arbitrary, perturbed points $s_j = (x_{u_j}, y_{u_j}) = f(u_j) + \delta_j, j = 1, \dots, r$ of f, and we want to recover f from these points. By a slight abuse of notation we interpret s_j as a complex number and write

$$s_j = x_j + iy_j\,. \tag{6.1}$$

We relate the curve parameter u to the boundary points s_j by computing the distance between two successive points s_{j-1}, s_j via

$$u_1 = 0, \tag{6.2}$$

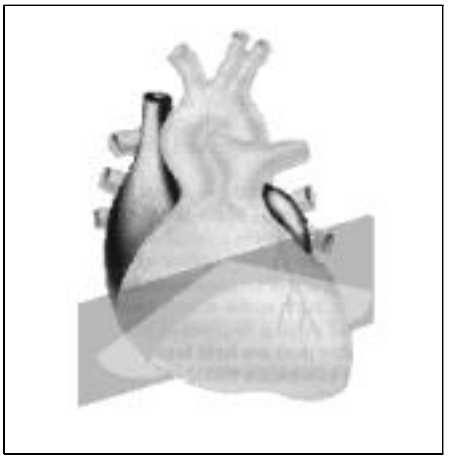
$$u_j = u_{j-1} + d_j, \tag{6.3}$$

$$d_j = \sqrt{(x_j - x_{j-1})^2 + (y_j - y_{j-1})^2} \tag{6.4}$$

for $j = 2, \dots, r$. Via the normalization $t_j = t_j/L$ with $L = u_r + d_N$ we force all sampling points to be in $[0, 1)$. Other choices for d_j in (6.1) can be found in [8] in conjunction with curve approximation using splines.

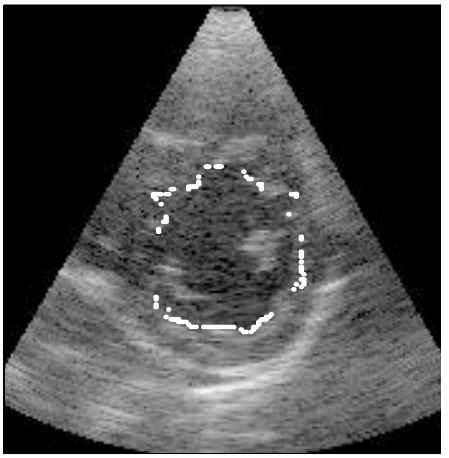
Having carried out the transformations (6.1)–(6.4), we can solve the problem of recovering the curve f from its perturbed points s_j by Algorithm 2.

6.1. Object boundary recovery in echocardiography. Trigonometric polynomials are certainly not suitable to model the shape of arbitrary objects. However, they are often useful in cases where an underlying (stationary) physical process implies smoothness conditions of the object. Typical examples arise in medical imaging, for instance in clinical cardiac studies, where the evaluation of cardiac function using parameters of left ventricular contractibility is an important constituent of an echocardiographic examination [30]. These parameters are derived using boundary tracing of endocardial borders of the left ventricle (LV). The extraction of the boundary of the LV comprises two steps, once the ultrasound image of a cross section of the LV is given; see Figure 6.1(a)–(d). First, an edge detection is applied to the ultrasound

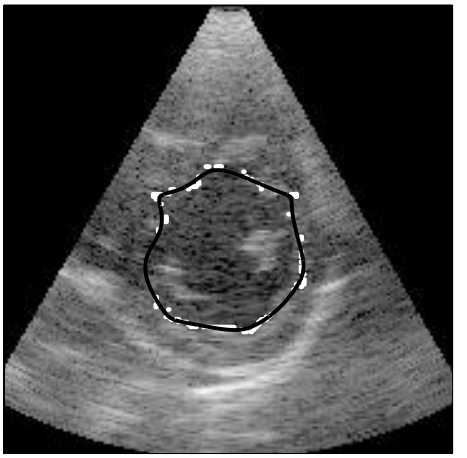


(a) 2-D echocardiography

(b) Cross section of LV



(c) Detected boundary points

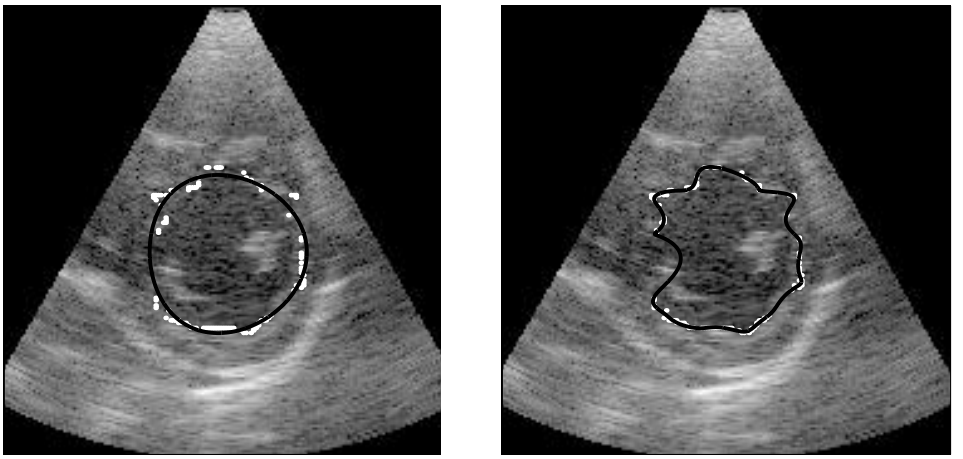


(d) Recovered boundary of LV computed by Algorithm 2

FIG. 6.1. *The recovery of the boundary of the left ventricle (LV) from* 2*-D ultrasound images is a basic step in echocardiography to extract relevant parameters of cardiac function.*

image to detect the boundary of the LV; cf. Figure 6.1(c). However, this procedure may be hampered by the presence of interfering biological structures (such as papillar muscles), the unevenness of boundary contrast, and various kinds of noise [29]. Thus edge detection often provides only a set of nonuniformly spaced, perturbed boundary points rather than a connected boundary. Therefore a second step is required to recover the original boundary from the detected edgepoints; cf. Figure 6.1(d). Since the shape of the LV is definitely smooth, trigonometric polynomials are particularly well suited to model its boundary.

After having transformed the detected boundary points as described in (6.1)–(6.4)



(a) Underfitted solution (b) Overfitted solution

FIG. 6.2. *The approximation in the left image results from using a too small polynomial degree, the approximation in the right image from a too large degree for the trigonometric approximation.*

we can use Algorithm 2 to recover the boundary. The noise level δ depends on the technical equipment being used; it can be determined from experimental experience. Figure 6.2(a)–(b) demonstrates the importance of determining a proper degree for the approximating polynomial. The approximation displayed in Figure 6.2(a) has been computed by solving (1.1), where M has been chosen too small; obviously we have underfitted the data. The overfitted approximation obtained by solving (1.1) using a too large M is shown in Figure 6.2(b). The approximation shown in Figure 6.1(d) has been computed by Algorithm 2; it provides the optimal balance between fitting the data and smoothness of the solution.

6.2. Boundary recovery from a sequence of images. In cardiac clinical studies one is more interested in the behavior of the LV over a period of time rather than in a single "snapshot." Thus for a fixed cross section we are given a sequence of ultrasound images (usually regularly spaced in time) describing the variation of the shape of the LV with time. One cycle from diastole (the state of maximal contraction of the LV), passing systole (the state of maximal expansion) to the next diastole consists typically of about 30 image frames. Since the behavior of the LV is (at least for a short period of time) almost periodic, one can model the varying shape of a fixed cross section of the LV as a distorted 2-D torus, which in turn can be interpreted as a 2-D trigonometric polynomial. Clearly, we have to use a different degree for the time coordinate τ and for the spatial coordinate u.

Due to interfering biological structures and other distortions it sometimes happens that some of the image frames cannot be used to extract any reliable boundary information. Thus we have to approximate these missing boundaries from the information of the other image frames. To be more precise, assume that an echocardiographic examination provides a sequence of ultrasound images I_τ taken at timepoints $\tau = 0, 1, \ldots, T-1$, where T is approximately the length of one diastolic cycle (the timepoints could also be nonuniformly spaced). Assume that some of the images I_τ

provide no useful information, so that we can only detect boundary points $\{s_{j,k}\}_{k=1}^{r_j}$ from the images I_{τ_j}, where $\{\tau_j\}_{j=1}^{r}$ is a subset of $0, 1, \ldots, T-1$. In order to get a complete description of the LV for the time interval $[0, T]$, we have not only to approximate the boundaries f_j from each I_j, but we also have to recover the boundaries corresponding to the missing images. In other words we look for a 2-D trigonometric polynomial $p_* \in \boldsymbol{P}_M^2$ of appropriate degree M that satisfies $p(\tau_j, u_{j,k}) \equiv (x_{j,k}, y_{j,k})$, where the parameter u is related to $s_{j,k} = x_{j,k} + iy_{j,k}$ by formulas (6.2)–(6.4). This approximation can be computed by the 2-D version of Algorithm 2, as indicated in the beginning of section 5.

Under certain conditions we can use the 1-D version of Algorithm 2 instead of its 2-D version. As long as the assumptions of Corollary 5.1 are satisfied, we can compute $p_* \in \boldsymbol{P}_M^2$ by a successive application of Algorithm 2. We first approximate the boundaries f_j for each j separately from its samples $\{s_{j,k}\}_{k=1}^{r_j}$, which yields j different polynomials $p^{(M_j)} \in \boldsymbol{P}_{M_j}$. Having done this, the next step is to recover the missing boundaries at those timepoints where no information is available. We proceed by approximating successively the missing information "line by line." We choose $u = 0$, say, and approximate the missing information from the samples $p^{(M_j)}(u)$ taken at the time points $\tau_j, j = 1, \ldots, r$.

Note that the Toeplitz matrices of the systems $(T_M)_u c_u^{(M)} = b_u^{(M)}$ coincide for all u, since the sampling geometry is constant along the u-coordinate (because we have recovered all samples at each τ_j). Thus, we have to solve multiple Toeplitz systems with the same system matrix but a different right-hand side. It is well known that this can be done efficiently by exploiting the Gohberg–Semencul representation of the inverse of the Toeplitz matrix [14]. In our context this reads as follows. We solve

$$(T_M)_u c_u^{(M)} = b_u^{(M)} \tag{6.5}$$

for one u by Algorithm 2. We can solve now all other systems efficiently by establishing $(T_M)^{-1}$ in the Gohberg–Semencul form

$$(T_M)^{-1} = \left([L^{(M)}]^* L^{(M)} - U^{(M)} [U^{(M)}]^* \right) / z_0, \tag{6.6}$$

where $L^{(M)}$ is a lower triangular Toeplitz matrix with $z = [z_0, z_1, \ldots, z_{2M}]^T$ as its first column and $U^{(M)}$ is an upper triangular Toeplitz matrix with $[0, z_1, \ldots, z_{2M}]^T$ as its last column (z being the first column of $(T_M)^{-1}$). The matrix–vector multiplications to compute $c_u^{(M)} = (T_M)_u^{-1} b_u^{(M)}$ can now be carried out quickly using the FFT by embedding $L^{(M)}$ and $U^{(M)}$ into circulant matrices.

7. Miscellaneous remarks. For sampling sets with large gaps it can happen that the system $T_M c^{(M)} = b^{(M)}$ becomes ill conditioned with increasing degree M and therefore Algorithm 2 may become unstable [6]. In this case one can use a different, more robust, approach, which comes at higher computational cost [27]. We solve the system $T_M c^{(M)} = b^{(M)}$ iteratively, e.g., by the conjugate gradient method until a certain stopping criterion is satisfied at iteration k, say, yielding the solution $c_k^{(M)}$. We use this solution as an initial guess at the next level $M+1$ by setting $c_0^{(M+1)} = [0\ (c_k^{(M)})^T\ 0]^T$. The crucial point in this procedure is to find a stopping criterion that guarantees convergence of the iterates; see [27, 24] for more details.

The computation of the entries of the Toeplitz matrix in section 6 involves the nodes u_j, which in this particular case depend on the (perturbed) samples s_j. Therefore not only the right-hand side $b^{(M)}$, but also T_M is subject to perturbations. Hence,

in principle one might use the concept of total least squares (see [12]) instead of a least squares approach. A detailed discussion of this modification is beyond the scope of this paper.

Acknowledgments. The major part of this work was completed during my stay at Stanford University. I want to thank Prof. David Donoho and the Department of Statistics for their hospitality.

REFERENCES

[1] H. Akaike, *Block Toeplitz matrix inversion*, SIAM J. Appl. Math., 24 (1973), pp. 234–241.
[2] G. S. Ammar and W. B. Gragg, *Superfast solution of real positive definite Toeplitz systems*, SIAM J. Matrix Anal. Appl., 9 (1988), pp. 61–76.
[3] G. Beylkin, *On the fast Fourier transform of functions with singularities*, Appl. Comput. Harmon. Anal., 2 (1995), pp. 363–381.
[4] S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan, *Linear Matrix Inequalities in System and Control Theory*, SIAM, Philadelphia, PA, 1994.
[5] R. H. Chan and M. K. Ng, *Conjugate gradient methods for Toeplitz systems*, SIAM Rev., 38 (1996), pp. 427–482.
[6] G. Cybenko, *The numerical stability of the Levinson-Durbin algorithm for Toeplitz systems of equations*, SIAM J. Sci. Statist. Comput., 1 (1980), pp. 303–320.
[7] C. Demeure, *Fast QR factorization of Vandermonde matrices*, Linear Algebra Appl., 122–124 (1989), pp. 165–194.
[8] P. Dierckx, *Curve and Surface Fitting with Splines*, Monographs on Numerical Analysis, Oxford University Press, Oxford, UK, 1993.
[9] A. Dutt and V. Rokhlin, *Fast Fourier transforms for nonequispaced data*, SIAM J. Sci. Comput., 14 (1993), pp. 1368–1393.
[10] H. Fassbender, *On numerical methods for discrete least-squares approximation by trigonometric polynomials*, Math. Comp., 66 (1997), pp. 719–741.
[11] H. G. Feichtinger, K. Gröchenig, and T. Strohmer, *Efficient numerical methods in non-uniform sampling theory*, Numer. Math., 69 (1995), pp. 423–440.
[12] R. D. Fierro, G. H. Golub, P. C. Hansen, and D. P. O'Leary, *Regularization by truncated total least squares*, SIAM J. Sci. Comput., 18 (1997), pp. 1223–1241.
[13] W. Gautschi, *How (un)stable are Vandermonde systems?*, in Asymptotic and Computational Analysis, Dekker, New York, 1990, pp. 193–210.
[14] I. Gohberg and A. Semencul, *On the inversion of finite Toeplitz matrices and their continuous analogs*, Mat. Issled., 2 (1972), pp. 201–233.
[15] G. Golub, M. Heath, and G. Wahba, *Generalized cross-validation as a method for choosing a good ridge parameter*, Technometrics, 21 (1979), pp. 215–223.
[16] G. Golub and C. van Loan, *Matrix Computations*, 3rd ed., Johns Hopkins, Baltimore, MD, 1996.
[17] K. Gröchenig, *personal communication*, 1997.
[18] K. Gröchenig, *A discrete theory of irregular sampling*, Linear Algebra Appl., 193 (1993), pp. 129–150.
[19] K. Gröchenig, *Finite and infinite-dimensional models for non-uniform sampling*, in Proceedings of the International Workshop on Sampling Theory and Applications, Aveiro, Portugal, 1997, pp. 285–290.
[20] P. C. Hansen, *Analysis of discrete ill-posed problems by means of the L-curve*, SIAM Rev., 34 (1992), pp. 561–580.
[21] N. Levinson, *The Wiener RMS (root-mean square) error criterion in filter design and prediction*, J. Math. Phys., 25 (1947), pp. 261–278.
[22] A. Newbery, *Trigonometric interpolation and curve-fitting*, Math. Comp., 24 (1970), pp. 869–876.
[23] A. Okabe, B. Boots, and K. Sugihara, *Spatial tessellations: Concepts and Applications of Voronoĭ Diagrams*, John Wiley & Sons Ltd., Chichester, UK, 1992.
[24] M. Rauth and T. Strohmer, *Smooth approximation of potential fields from noisy scattered data*, Geophysics, 63 (1998), pp. 85–94.
[25] L. Reichel, G. Ammar, and W. Gragg, *Discrete least squares approximation by trigonometric polynomials*, Math. Comp., 57 (1991), pp. 273–289.
[26] W. Rudin, *Fourier Analysis on Groups*, Wiley Interscience, New York, 1976.

[27] O. Scherzer and T. Strohmer, *A multi-level algorithm for the solution of moment problems*, Numer. Funct. Anal. Optim., 19 (1998), pp. 353–375.

[28] T. Strohmer, *Computationally attractive reconstruction of band-limited images from irregular samples*, IEEE Trans. Image Proc., 6 (1997), pp. 540–548.

[29] M. Suessner, M. Budil, T. Strohmer, M. Greher, G. Porenta, and T. Binder, *Contour detection using artificial neuronal network presegmention*, in Proceedings of the Symposium on Computers in Cardiology, Vienna, 1995, pp. 737–740.

[30] D. Wilson, E. Geiser, and J. Li, *Feature extraction in* 2*-dimensional short-axis echocardiographic images*, J. Math. Imaging Vision, 3 (1993), pp. 285–298.

SIAM J. SCI. COMPUT.
Vol. 22, No. 4, pp. 1184–1198
© 2000 Society for Industrial and Applied Mathematics

POSITIVITY-PRESERVING SCHEMES IN MULTIDIMENSIONS*

AMBADY SURESH†

Abstract. The performance of the MUSCL–Hancock upwind scheme is examined on problems of two-dimensional advection. It is found that when the advection direction is skewed relative to the mesh, most discrete total variation (TVD) limiters give rise to large spurious oscillations near discontinuities. The cause of these oscillations is traced to reconstructions that are not bounded by neighboring cell averages. It is proved that if the reconstruction in each cell is bounded by the cell averages of first order neighbors, then the MUSCL–Hancock scheme is positivity preserving. A simple limiter that achieves such bounded reconstructions is presented next along with a variant that is uniformly second order accurate. Numerical experiments show that the new schemes are accurate and efficient and compare favorably with other schemes.

Key words. advection, positivity preserving, upwind

AMS subject classifications. 65M06, 76D05

PII. S1064827599360443

1. Introduction.

Upwind methods [1], [2], [3], [4] for solving partial differential equations in general and the Euler equations of fluid flow in particular have gained popularity in the last decade due to their ability to capture discontinuities without the spurious oscillations incurred by other methods. Since their inception, upwind methods have been incorporated into a number of academic as well as commercial flow solvers.

Most upwind methods are first developed for the one-dimensional advection equation, where theoretical properties of the numerical scheme, such as the decrease of the discrete total variation (TVD), or monotonicity preservation, can be proved. The extension of these methods to multidimensions is typically carried out on a dimension-by-dimension basis where the one-dimensional algorithm is applied separately in each dimension. Although not much can be said about their nonoscillatory properties, these schemes have been quite successful in resolving complex patterns of interacting shocks and smooth waves [5].

For two-dimensional advection, where exact solutions are known, these schemes do not perform so well. In fact, this paper is motivated by the poor performance of one such scheme on these problems. The base scheme we choose is the MUSCL–Hancock scheme [6] which uses a midpoint rule in time and a linear reconstruction in each cell. The linear reconstruction is obtained by TVD limiters, like Superbee, Minmod, etc. acting on one-dimensional slivers of data. We find that when the advection direction is skewed relative to the mesh, all limiters except for the most dissipative Minmod limiter give rise to large spurious overshoots and undershoots near discontinuities. Indeed, some of these solutions are no better than the base scheme without any limiting at all (see Figures 1 and 2).

The reasons for these oscillations are traced to reconstructions that are not bounded by the initial data and lead us to another class of schemes proposed in recent years, namely, positivity-preserving (PP) schemes [7], [8], [9], [10], [11], [12].

*Received by the editors August 26, 1999; accepted for publication (in revised form) March 17, 2000; published electronically October 18, 2000.

http://www.siam.org/journals/sisc/22-4/36044.html

†Dynacs Engineering, Inc., NASA Glenn Research Center, Cleveland, OH 44135 (ambady.suresh@grc.nasa.gov).

For positive initial data, these schemes have the property that the solution at any future time is also positive. These schemes are also called positive definite, shape preserving, and bounded schemes in the literature. We show that if the reconstruction is bounded by the data, then the MUSCL–Hancock scheme can be proved to be PP. An efficient limiter that achieves such bounded reconstructions is presented next along with a fix for uniform second order accuracy. Finally, we present some numerical experiments to assess the performance of PP schemes and compare them to a few existing schemes.

2. TVD schemes on two-dimensional advection problems. Let us consider the two-dimensional advection equation given by

$$u_t + au_x + bu_y = 0 \tag{2.1}$$

The numerical scheme we use is the MUSCL–Hancock scheme, which employs a midpoint rule in time and linear reconstructions in each cell, a combination widely used in many commercial codes for the Euler equations. We assume a uniform grid of spacing Δx and Δy and $a \geq 0$ and $b \geq 0$. If we denote by $u_{i,j}$ the approximations to the cell averages at time level n, the scheme can be written as

$$u_{i,j}^{n+1} = u_{i,j}^n - \lambda_x(u_{i+1/2,j}^{n+1/2} - u_{i-1/2,j}^{n+1/2}) - \lambda_y(u_{i,j+1/2}^{n+1/2} - u_{i,j-1/2}^{n+1/2}), \tag{2.2}$$

where $\lambda_x = a\Delta t/\Delta x$, $\lambda_y = b\Delta t/\Delta y$. The interface values are obtained by a Taylor series expansion inside the upwind cell, i.e.,

$$u_{i+1/2,j}^{n+1/2} = u_{i,j}^n + \frac{1}{2}\Delta x\,(u_x)_{i,j} + \frac{1}{2}\Delta t\,(u_t)_{i,j}. \tag{2.3}$$

Substituting for u_t from (2.1) and denoting the normalized values of $(u_x)_{i,j}$ and $(u_y)_{i,j}$ by $S_{i,j}^x$ and $S_{i,j}^y$, the interface values can be derived as

$$\begin{aligned} u_{i+1/2,j}^{n+1/2} &= u_{i,j}^n + \tfrac{1}{2}(1-\lambda_x)S_{i,j}^x - \tfrac{1}{2}\lambda_y S_{i,j}^y, \\ u_{i,j+1/2}^{n+1/2} &= u_{i,j}^n + \tfrac{1}{2}(1-\lambda_y)S_{i,j}^y - \tfrac{1}{2}\lambda_x S_{i,j}^x. \end{aligned} \tag{2.4}$$

If $S_{i,j}^x$ and $S_{i,j}^y$ are given by their central difference estimates, i.e.,

$$\begin{aligned} S_{i,j}^x &= \tfrac{1}{2}(u_{i+1,j} - u_{i-1,j}), \\ S_{i,j}^y &= \tfrac{1}{2}(u_{i,j+1} - u_{i,j-1}), \end{aligned} \tag{2.5}$$

then (2.2) is second order accurate in time and space and stable in the region $\lambda_x \geq 0$, $\lambda_y \geq 0$, $\lambda_x + \lambda_y \leq 1$. Thus, the time step can be defined in terms of a CFL number $\sigma = \lambda_x + \lambda_y$ by

$$\Delta t = \sigma/(a/\Delta x + b/\Delta y). \tag{2.6}$$

The slopes $S_{i,j}^x$ and $S_{i,j}^y$ are obtained by one-dimensional TVD limiters acting on slivers of one-dimensional data. For the x direction, if we define

$$\begin{aligned} s_+ &= (u_{i+1,j} - u_{i,j}), \\ s_- &= (u_{i,j} - u_{i-1,j}), \end{aligned} \tag{2.7}$$

then the Minmod, Average, and Superbee limiters can be written as

$$
\begin{aligned}
S^x_{i,j} &= \tfrac{1}{2}\,[\mathrm{sgn}(s_+) + \mathrm{sgn}(s_-)]\,\mathrm{Min}(|s_-|, |s_+|),\\
S^x_{i,j} &= \tfrac{1}{2}\,[\mathrm{sgn}(s_+) + \mathrm{sgn}(s_-)]\,\mathrm{Min}(|s_+ + s_-|/2,\, 2|s_-|,\, 2|s_+|),\\
S^x_{i,j} &= \tfrac{1}{2}\,[\mathrm{sgn}(s_+) + \mathrm{sgn}(s_-)]\,\mathrm{Min}(\mathrm{Max}(|s_+|, |s_-|),\, 2|s_-|,\, 2|s_+|).
\end{aligned}
\tag{2.8}
$$

Note that in one dimension, all these schemes give updates bounded by the initial data; i.e., it can be shown that (see the appendix) if $b = 0$ and $a > 0$, then

$$
u^{n+1}_{i,j} \in [\mathrm{Min}(u^n_{i,j}, u^n_{i-1,j}), \mathrm{Max}(u^n_{i,j}, u^n_{i-1,j})]
\tag{2.9}
$$

so that if $u^n_{i,j} > 0$, for all i, j, then so is $u^{n+1}_{i,j}$. Such schemes are called PP schemes. However, this property does not carry over to two dimensions as we shall see below.

The results of a typical calculation with these schemes on a grid of 120×120 cells over the domain of $[-1, 1] \times [-1, 1]$, with $a = 1.0$, $b = 0.1$, $\sigma = 0.8$ at the final time $t = 20$ (1650 time steps), are shown in Figures 1 and 2. The initial condition is

$$
\begin{aligned}
u(x, y, 0) &= 1 \quad \text{for } (x^2 + y^2)^{1/2} < 0.4\\
&= 0 \quad \text{otherwise.}
\end{aligned}
\tag{2.10}
$$

Figure 1 actually shows the numerical solutions obtained on the line $y = -0.258$, while Figure 2 shows solutions on the line $y = 0.392$. As can be seen, both the Superbee and the Average limiters show large overshoots and undershoots. The Minmod limiter gives acceptable but diffused results.

Our numerical experiments over a wide range of initial conditions, propagation directions, and Courant numbers indicate that most TVD limiters give oscillatory solutions when the convection angle is not aligned with either axis or the diagonal. The worst oscillations are observed with the Superbee limiter while none are observed with the Minmod limiter.

3. PP schemes in two dimensions. To understand why these oscillations occur it is instructive to rewrite (2.2) and (2.4) as

$$
\begin{aligned}
u^{n+1}_{i,j} &= (1 - \lambda_x - \lambda_y)(u_{i,j} - \tfrac{1}{2}\lambda_x\, S^x_{i,j} - \tfrac{1}{2}\lambda_y\, S^y_{i,j})\\
&+\lambda_x(u_{i-1,j} + \tfrac{1}{2}(1 - \lambda_x)\, S^x_{i-1,j} - \tfrac{1}{2}\lambda_y\, S^y_{i-1,j})\\
&+\lambda_y(u_{i,j-1} - \tfrac{1}{2}\lambda_x\, S^x_{i,j-1} + \tfrac{1}{2}(1 - \lambda_y)\, S^y_{i,j-1})
\end{aligned}
\tag{3.1}
$$

If the reconstruction assumed in each cell is written in normalized coordinates as

$$
R_{i,j}(\xi, \eta) = u_{i,j} + S^x_{i,j}\xi + S^y_{i,j}\eta,
\tag{3.2}
$$

with $|\xi|, |\eta| \le 1/2$, then the above equation can be written as

$$
\begin{aligned}
u^{n+1}_{i,j} = \;& (1 - \lambda_x - \lambda_y)\, R_{i,j}(-\lambda_x/2, -\lambda_y/2)\\
&+\lambda_x\, R_{i-1,j}((1 - \lambda_x)/2, -\lambda_y/2)\\
&+\lambda_y\, R_{i,j-1}(-\lambda_x/2, (1 - \lambda_y)/2).
\end{aligned}
\tag{3.3}
$$

This shows that the value of $u_{i,j}$ at the next time step is a convex combination of sampled reconstructed values from the three cells (i, j), $(i - 1, j)$, and $(i, j - 1)$. Moreover, under the stability limit, $\lambda_x \ge 0$, $\lambda_y \ge 0$, $\lambda_x + \lambda_y \le 1$, the points where

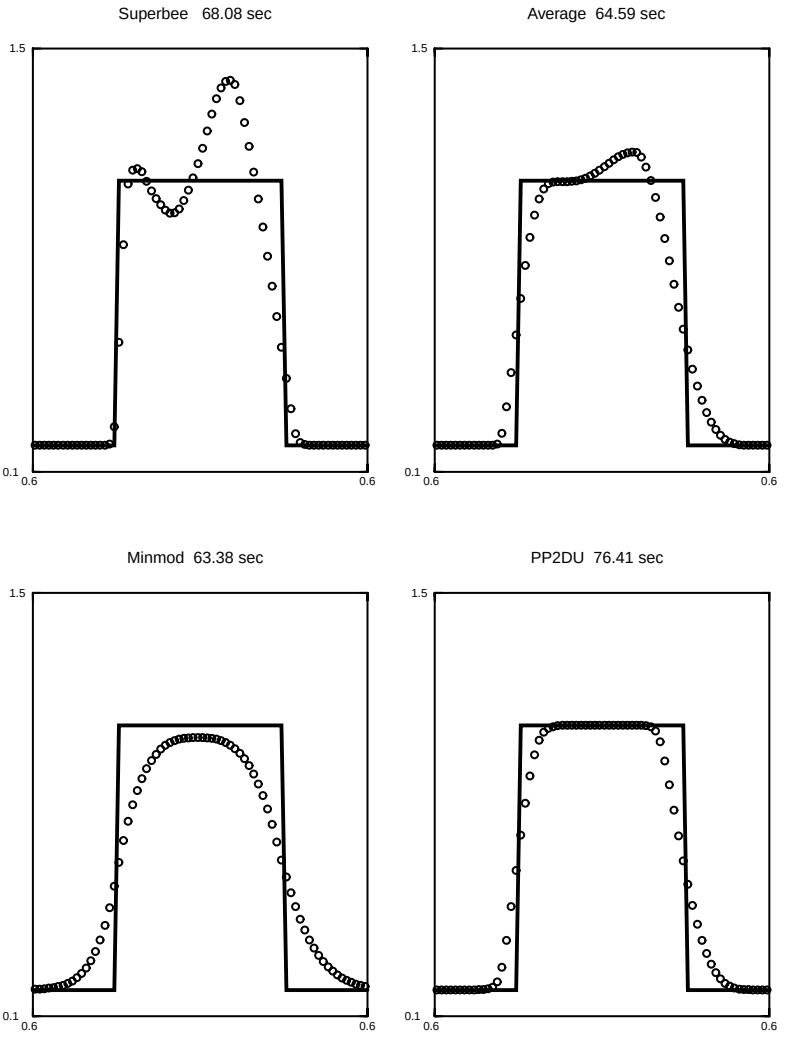


FIG. 1. *Advection of* (2.10) *with* $a = 1.0$, $b = 0.1$, *CFL* $= 0.8$, *and* $t = 20.120 \times 120$ *uniform grid. Results are shown on the line* $y = -0.258$.

the reconstructions are sampled lie inside their respective cells. Thus, a simple way of keeping $u_{i,j}^{n+1}$ bounded is to bound the reconstructions $R_{i,j}$.

To this end, let us define $N_{i,j}$ to be the set of cell averages of the immediate first order neighbors of the cell (i, j) and $u_{i,j}$,

$$N_{i,j} = \begin{bmatrix} u_{i-1,j+1}, & u_{i,j+1}, & u_{i+1,j+1}, \\ u_{i-1,j}, & u_{i,j}, & u_{i+1,j}, \\ u_{i-1,j-1}, & u_{i,j-1}, & u_{i+1,j-1} \end{bmatrix}. \tag{3.4}$$

Let $U_{i,j}$ be the range of variation of u on $N_{i,j}$,

$$U_{i,j} = [\mathrm{Min}[N_{i,j}], \mathrm{Max}[N_{i,j}]]. \tag{3.5}$$

The following result is then obvious from (3.3).

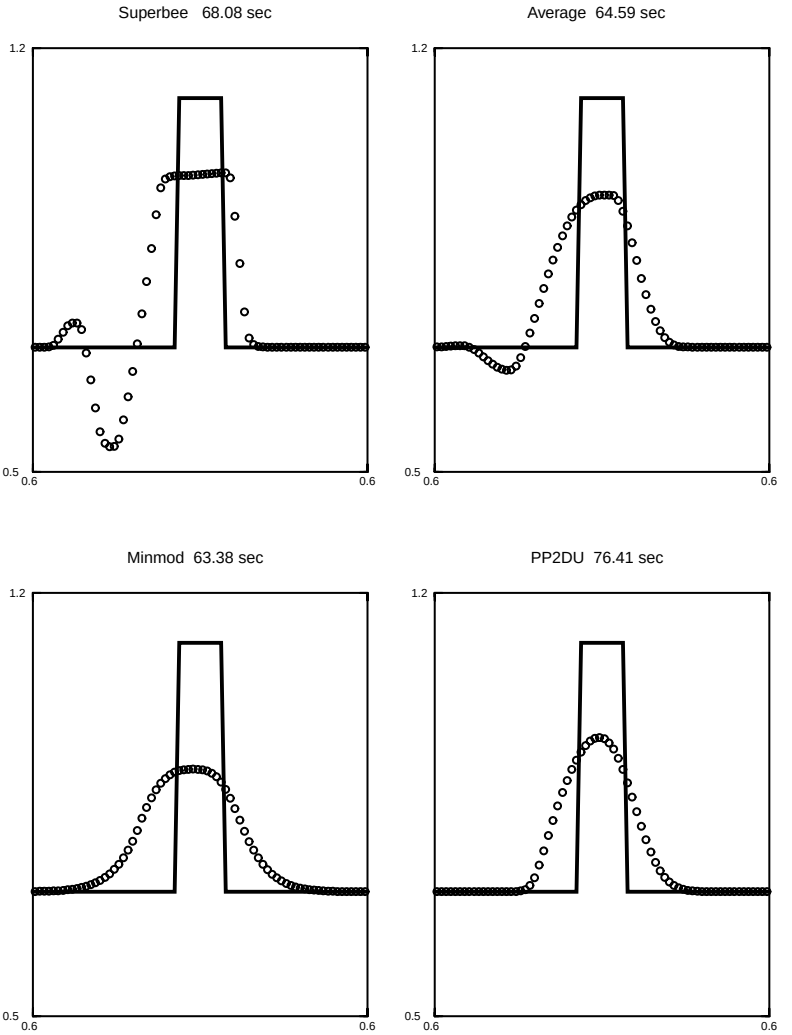


FIG. 2. *Advection of* (2.10) *with* a=1.0, $b = 0.1$, *CFL* $= 0.8$, *and* $t = 20.120 \times 120$ *uniform grid. Results are shown on the line* $y = 0.392$.

THEOREM 3.1. *Assume that for the scheme* (2.2)–(2.4) *the reconstruction in each cell is bounded by its immediate neighbors; i.e., for all* (i, j),

$$R_{i,j} \in U_{i,j}. \tag{3.6}$$

Then, for $\lambda_x + \lambda_y \leq 1$,

$$u_{i,j}^{n+1} \in U_{i,j} \cup U_{i-1,j} \cup U_{i,j-1}, \tag{3.7}$$

i.e., the cell average at the next time step lies inside the union of the averages in the neighborhoods of (i, j), $(i-1, j)$, *and* $(i, j-1)$.

Thus, if (3.6) is satisfied in each cell, the scheme becomes PP. As a counter

example to this theorem, consider positive data given by

$$u_{-2,0} = 100, \quad \begin{array}{lll} u_{-1,1} = 1, & u_{0,1} = 0, & \\ u_{-1,0} = 1, & u_{0,0} = 0, & u_{1,0} = 0, \\ u_{-1,-1} = 1, & u_{0,-1} = 10, & u_{1,-1} = 100, \\ & u_{0,-2} = 100. & \end{array} \tag{3.8}$$

With the Superbee and Average limiters the reconstruction $R_{0,-1}$ has large negative values and does not satisfy (3.6). Not surprisingly, for $\lambda_x = 0.6, \lambda_y = 0.2$, both theses schemes give $u_{0,0}^{n+1} = -8/25$, which shows that these schemes do not preserve positivity. On the other hand, the Minmod limiter gives reconstructions that are bounded by first order neighbors and gives $\bar{u}_{0,0}^{n+1} = 57/50$, which is acceptable. This explains the large undershoots/overshoots observed in Figures 1 and 2.

It is easily verified that Theorem 3.1 holds true for all propagation directions provided the interface values (2.4) are defined from upwind cells. The general CFL limit is then $|\lambda_x| + |\lambda_y| \le 1$.

Results similar to Theorem 3.1 have been proved in the literature in related contexts. Batten, Lambert, and Causon [11] prove a PP condition for multidimensional advection on unstructured meshes. Perthame and Shu [9] and Linde and Roe [12] prove PP conditions on unstructured meshes for the two-dimensional Euler equations. However, both of these are for convex time Runge–Kutta time stepping, for which the TVD limiters applied dimension-by-dimension can be shown to be PP under a reduced CFL limit.

We note that the only thing positivity preservation guarantees is that the cell averages at all future times will lie in a certain range governed by the initial conditions. It does not rule out spurious oscillations in this range. However, in practice, in numerical experiments, we see that the PP constraint goes a long way toward getting rid of all spurious oscillations.

The condition that the reconstruction be bounded over the whole cell by all its immediate neighbors (including corner neighbors) was introduced by Barth and Jesperson [13] in the context of unstructured grids. The above result shows that this condition is useful for constructing PP schemes on structured meshes as well. It is also similar in spirit to the earlier work of Spekreijse [14], although the details are different.

In the next section, we derive an efficient limiter that enforces (3.6) but is not as dissipative as the Minmod limiter. We remark that the choice of stencil, i.e., (3.4) used above, is somewhat arbitrary. In fact, for the Minmod scheme, the reconstruction is so tightly constrained that $R_{i,j}$ actually lies inside the range of $u_{i,j}$ and two linearly independent cells from $u_{i\pm1,j}$ and $u_{i,j\pm1}$. From this it follows that $u_{i,j}^{n+1}$ is bounded by the four values $u_{i,j}$, $u_{i-1,j}$, $u_{i,j-1}$, and $u_{i-1,j-1}$, a much smaller stencil than (3.7).

4. A PP limiter. The procedure for modifying the slopes to satisfy (3.6) is far from unique. The approach described here is similar to the approach of Barth and Jesperson [13], adapted to structured grids and modified so as to depend continuously on the data.

To satisfy (3.6) within a cell, the idea is to restrict the slopes so that the reconstructed values at the four corners of the cell lie inside the required interval, i.e., (3.5). Without loss of generality, we can adjust the value of $u_{i,j}$ to zero and define

$$V_{min} = \text{Min} \begin{bmatrix} u_{i-1,j+1} - u_{i,j}, & u_{i,j+1} - u_{i,j}, & u_{i+1,j+1} - u_{i,j}, \\ u_{i-1,j} - u_{i,j}, & 0, & u_{i+1,j} - u_{i,j}, \\ u_{i-1,j-1} - u_{i,j}, & u_{i,j-1} - u_{i,j}, & u_{i+1,j-1} - u_{i,j} \end{bmatrix}, \tag{4.1}$$

$$V_{max} = \text{Max}\begin{bmatrix} u_{i-1,j+1}-u_{i,j}, & u_{i,j+1}-u_{i,j}, & u_{i+1,j+1}-u_{i,j}, \\ u_{i-1,j}-u_{i,j}, & 0, & u_{i+1,j}-u_{i,j}, \\ u_{i-1,j-1}-u_{i,j}, & u_{i,j-1}-u_{i,j}, & u_{i+1,j-1}-u_{i,j} \end{bmatrix}. \tag{4.2}$$

Then, for the north-east corner, we compute (with the slopes given by (2.5))

$$u_{NE} = \frac{1}{2}S^x_{i,j} + \frac{1}{2}S^y_{i,j} \tag{4.3}$$

and a ϕ_{NE} as follows:

$$\begin{array}{llll} \textit{if} & u_{NE} > & V_{max}, & \phi_{NE} = V_{max}/u_{NE}, \\ \textit{if} & u_{NE} < & V_{min}, & \phi_{NE} = V_{min}/u_{NE}, \\ \textit{else} & & & \phi_{NE} = 1. \end{array} \tag{4.4}$$

We repeat these calculations for all corners, and the final slopes are given by

$$\begin{aligned} S^x_{i,j} &\leftarrow \text{Min}(\phi_{NE}, \phi_{NW}, \phi_{SE}, \phi_{SW})\, S^x_{i,j}, \\ S^y_{i,j} &\leftarrow \text{Min}(\phi_{NE}, \phi_{NW}, \phi_{SE}, \phi_{SW})\, S^y_{i,j}, \end{aligned} \tag{4.5}$$

where the arrow notation stands for replacement or overwriting.

One problem with the above algorithm is the value of ϕ_{NE} does not depend continuously on the data. For example, if $V_{max} = 0$ such as near an extrema, and u_{NE} approaches zero through positive values, the value of ϕ_{NE} jumps from zero to one. This can be fixed by redefining V_{max} and V_{min} as

$$V_{min} = \text{Min}\begin{bmatrix} u_{i-1,j+1}-u_{i,j}, & u_{i,j+1}-u_{i,j}, & u_{i+1,j+1}-u_{i,j}, \\ u_{i-1,j}-u_{i,j}, & -\epsilon, & u_{i+1,j}-u_{i,j}, \\ u_{i-1,j-1}-u_{i,j}, & u_{i,j-1}-u_{i,j}, & u_{i+1,j-1}-u_{i,j} \end{bmatrix}, \tag{4.6}$$

$$V_{max} = \text{Max}\begin{bmatrix} u_{i-1,j+1}-u_{i,j}, & u_{i,j+1}-u_{i,j}, & u_{i+1,j+1}-u_{i,j}, \\ u_{i-1,j}-u_{i,j}, & +\epsilon, & u_{i+1,j}-u_{i,j}, \\ u_{i-1,j-1}-u_{i,j}, & u_{i,j-1}-u_{i,j}, & u_{i+1,j-1}-u_{i,j} \end{bmatrix}, \tag{4.7}$$

where ϵ is a small positive number (10^{-20} in all our computations).

Further simplifications to the algorithm defined by (2.5), (4.3)–(4.7), are possible due to the symmetry of the grid. It turns out that restricting all the corner values to lie inside $[V_{min}, V_{max}]$ is equivalent to restricting $|S^x_{i,j}| + |S^y_{i,j}|$ to lie inside another interval. We skip the details and give the result in algorithm form.

Let the slopes be defined initially by their centered difference values, namely, (2.5), and let V_{min} and V_{max} be defined by (4.6) and (4.7). Then we define

$$V = 2\,\frac{\text{Min}(|V_{min}|, |V_{max}|)}{(|S^x_{i,j}| + |S^y_{i,j}|)}. \tag{4.8}$$

The final slopes are then

$$\begin{aligned} S^x_{i,j} &\leftarrow \text{Min}(1\,, V)\, S^x_{i,j}, \\ S^y_{i,j} &\leftarrow \text{Min}(1\,, V)\, S^y_{i,j}. \end{aligned} \tag{4.9}$$

It can be verified that after limiting, the reconstructed values in the cell (i, j) lie inside the interval $U_{i,j}$ and that the reconstruction depends continuously on the data.

The main problem with this limiter is the loss of accuracy near critical points (a critical point is where the gradient of a function vanishes) which makes these schemes only first order accurate in the Max. norm. If a scheme is required to be second order accurate everywhere (i.e., in the Max. norm), such a scheme will advect second order polynomials exactly. It follows that while advecting smooth extrema such a scheme will give updates not bounded by the data. It thus appears that second order accuracy in the Max. norm and positivity preservation are mutually exclusive, even in one dimension.

In the next section we present a compromise that relaxes the PP constraint near critical points so that uniform second order accuracy can be achieved.

5. Uniform second order accuracy. The essential idea behind this fix is to redefine the interval $[V_{min}, V_{max}]$ so that near discontinuities it is the same as above but near a smooth critical point, such as an extremum, it is strictly larger so that the limiting process leaves the original values of the slopes unaltered. We are able to do this with some success only on a corner by corner basis and not for the whole cell. Thus, we focus on the limiting process defined by (4.3), (4.4), and (4.5), with V_{min} and V_{max} defined by (4.6), (4.7).

When u_{NE} falls outside $[V_{min}, V_{max}]$, there are two possibilities: (1) we are near a discontinuity so that the above limiting process is appropriate; (2) we are near a smooth critical point where no limiting is appropriate. We distinguish between these two situations by looking at the data along the diagonal $u_{i+p,j+p}$, $p = -1, 2$. If the interval $[u_{i,j}, u_{i+1,j+1}]$ falls outside the interval $[u_{i-1,j-1}, u_{i+2,j+2}]$, we conclude that we are near a smooth critical point; otherwise we conclude we are near a discontinuity. Thus, we define

$$
\begin{aligned}
t_{max} &= \mathrm{Max}(u_{i,j}, u_{i+1,j+1}),\\
t_{min} &= \mathrm{Min}(u_{i,j}, u_{i+1,j+1}),\\
w_{max} &= \mathrm{Max}(u_{i-1,j-1}, u_{i+2,j+2}),\\
w_{min} &= \mathrm{Min}(u_{i-1,j-1}, u_{i+2,j+2}),\\
d_1 &= \mathrm{Max}(t_{max} - w_{max}, w_{min} - t_{min}, 0.0),
\end{aligned}
\tag{5.1}
$$

so that d_1 is positive near a critical point and zero otherwise. Then we define new values of V_{min} and V_{max} as

$$
\begin{aligned}
\hat{V}_{min} &= V_{min} - \epsilon_2\, d_1,\\
\hat{V}_{max} &= V_{max} + \epsilon_2\, d_1,
\end{aligned}
\tag{5.2}
$$

where ϵ_2 is a positive constant. Since d_1 is positive, the new interval $[\hat{V}_{min}, \hat{V}_{max}]$ is strictly larger than the old interval and depends continuously on the data. The value of ϕ_{NE} is then calculated as

$$
\begin{aligned}
&\text{if } u_{NE} > V_{max}, && \phi_{NE} = \mathrm{Min}(1, \hat{V}_{max}/u_{NE}),\\
&\text{if } u_{NE} < V_{min}, && \phi_{NE} = \mathrm{Min}(1, \hat{V}_{min}/u_{NE}),\\
&\text{else} && \phi_{NE} = 1.
\end{aligned}
\tag{5.3}
$$

These calculations are repeated for all corners and the final slopes calculated from (4.5).

Let us describe how this algorithm works in words so that its inherent limitations are clear. On fine enough meshes, when one of the corner values falls outside $[V_{min}, V_{max}]$, it means we are either near a discontinuity or a smooth critical point

(maxima, minima, saddle point, etc.). If we are near a discontinuity, we hope the data will be monotone on the diagonal line going through the corner and so d_1 will be zero and the slopes are limited just as in the original PP scheme. If we are near a critical point, however, data along the diagonal will not be monotone. Then $d_1 > 0$ and gives enough room so that the u_{NE} will be swallowed inside the new interval $[\hat{V}_{min}, \hat{V}_{max}]$, yielding $\phi_{NE} = 1$.

Since the two types of data, i.e., data near discontinuities and data near a critical point, are not easily characterized in two dimensions, not much can be theoretically proved about this algorithm. If $u_{i,j}$ is independent of j and $b = 0$, the corresponding one-dimensional scheme can be proved to be monotonicity-preserving and uniformly second order accurate for $\epsilon_2 > 0.25$. Thus we use $\epsilon_2 = 0.275$ in all our computations. In one dimension, this fix for uniform accuracy is a simplified version of the fix for uniform accuracy presented in Suresh and Huynh [15], in the context of higher order Runge–Kutta schemes.

For coding purposes, we summarize the final algorithm. In each cell, we first calculate the slopes by (2.5). Then for the north-east corner, we calculate u_{NE} by (4.3) and V_{min} and V_{max} by (4.6) and (4.7). Then we calculate ϕ_{NE} from (5.3). Note that the modified interval (i.e., (5.1), (5.2)) needs to be computed only when u_{NE} falls outside the original interval, which happens for relatively few cells. This is repeated for all the corner cells and the final slopes obtained from (4.5).

6. Two-dimensional Euler equations. The two schemes presented above are easily extended to the Euler equations. Following the usual approach, the primitive variables $V = (\rho, u, v, p)$ (where ρ is the density, (u, v) are the x and y components of the velocity, and p is the pressure), are reconstructed in each cell using the reconstruction procedure presented above. This yields the gradient of the primitive variables in each cell, which is used to march to the half time step. The values of V at the half time step on each side of a face are used to compute the flux via the Roe approximate Riemann solver modified with an entropy fix. Since the extension is fairly standard, we omit the details, which can be found in [16].

This scheme requires only one Riemann solver call per face per time step, while those using Runge–Kutta time stepping require at least two for second order accuracy. The time step is defined in terms of the CFL number σ by

$$\Delta t = \text{Min}\left(\frac{\sigma}{(|u| + c)/\Delta x + (|v| + c)/\Delta y}\right), \tag{6.1}$$

where the minimum is over all cells and c is the speed of sound given by $(\gamma p/\rho)^{1/2}$, with $\gamma = 1.4$.

7. Numerical experiments. We present some numerical experiments to assess the accuracy, efficiency, and nonoscillatory properties of the two schemes presented above. We refer to the strictly PP scheme (i.e., (4.8), (4.9)) as PP2D and the scheme with the fix for uniform accuracy as PP2DU. We are particularly interested in the following: (1) Does the fix for uniform accuracy in PP2DU give rise to spurious oscillations near discontinuities? (2) How bad is the loss of accuracy at critical points for PP2D? (3) How do these schemes compare in accuracy and efficiency with other schemes like MPDATA [18] or UTOPIA [10]?

All computations were performed on an isolated IRIS 100-MHZ R4000 SGI Indigo workstation. The execution times were obtained by using the timex utility.

We solved (2.1) on rectangular domains with periodic boundary conditions at the boundaries. A number of initial conditions, advection directions, and time steps

TABLE 1
Advection of (7.1) *with* $a = 0.2$, $b = 1.0$, $\sigma = 0.8$, $t = 10.0$, $\Delta x = \Delta y = 2/N$.

Scheme	N	Max. error	Max. order	L_1 error	L_1 order	CPU time - sec.
PP2D	60	1.72(-1)	-	3.02(-2)	-	
	120	5.68(-2)	1.60	6.97(-3)	2.11	
	240	1.88(-2)	1.60	1.65(-3)	2.08	306.77
PP2DU	60	9.16(-2)	-	3.01(-2)	-	
	120	2.03(-2)	2.17	6.80(-3)	2.15	
	240	4.79(-3)	2.08	1.60(-3)	2.09	347.25
Unlim.	60	9.16(-2)	-	3.08(-2)	-	
	120	2.03(-2)	2.17	6.81(-3)	2.18	
	240	4.79(-3)	2.08	1.60(-3)	2.09	228.99

were explored, but only a few cases are reported here. For initial conditions with discontinuities, we scanned the computed solution for overshoots, undershoots, and spurious oscillations, and we present line plots through these regions. These represent the worst-case results. In all these cases, PP2DU performed well without spurious oscillations.

Example 1. We return to the example already cited above. Here the domain of $[-1,1] \times [-1,1]$ is covered by a uniform grid of 120×120 cells, with $a = 1.0$, $b = 0.1$, $\sigma = 0.8$, and final time $t = 20$ (1650 time steps). This corresponds to 10 periods in x and one period in y. The initial condition is (2.10). The minimum and maximum values over the whole domain for Minmod, Average, Superbee, PP2D, and PP2DU are, respectively, $[0, 0.997]$, $[-0.093, 1.108]$, $[-0.400, 1.380]$, $[0, 1]$, and $[0, 1]$. The results of PP2DU, on the line $y = -0.258$, are also shown in Figure 1, which is less diffuse than the Minmod scheme and is nonoscillatory with no overshoots or undershoots. However, shock resolution is still quite poor for all schemes except perhaps Superbee. The result from PP2D is identical and is not shown. In terms of efficiency, PP2DU is about 20% more expensive than the Minmod scheme.

Figure 2 shows results from the same computation but this time on the line $y = 0.392$. Although the discontinuity is rather narrow, the Minmod and PP2DU schemes give acceptable results, while both the Superbee and Average limiters show large undershoots.

Example 2. The next problem involves smooth periodic initial conditions on the domain $[-1,1] \times [-1,1]$, which are given by

$$u(x, y, 0) = \sin^4(\pi(x + 1)) + \sin^4(\pi(y + 1)) \tag{7.1}$$

along with $a = 0.2$, $b = 1.0$, $\sigma = 0.8$, and final time $t = 10$. In this case we solve on three meshes and estimate the order of accuracy of the scheme. We compare the results with the unlimited scheme.

The results are presented in Table 1. As can be seen, there is little difference between PP2DU and the unlimited scheme, especially on the finer meshes. This indicates that the fix for uniform accuracy is working well and is not too expensive (about 12% more than PP2D). As expected, the results from PP2D show much higher Max. errors due to the loss of accuracy at extrema and less than second order accuracy in this norm.

Example 3. In this example, we compare with the scheme of Thuburn [10] for multidimensional advection. This scheme (TH1) (see Table 2) is a limited form of the UTOPIA scheme of Leonard, MacVean, and Lock [8]. Unlike the approach presented here, the limiting is performed directly on the interface fluxes at the half time step,

TABLE 2
Advection of (7.2) *with* $\sigma = 0.5$, $\Delta x = \Delta y = 1/31$. *Problem* 1: $a = 1$, $b = 0$, 62 *time steps; Problem* 2: $a = 1$, $b = 1$, 124 *time steps.*

	Problem 1		Problem 2	
Scheme	Min. value	Max. value	Min. value	Max. value
TH1	0.000	0.927	0.000	0.847
PP2DU	0.000	0.958	0.000	0.877
PP2D	0.000	0.891	0.000	0.760

i,e. on $u^{n+1/2}_{i+1/2,j}$.

The domain here is $[0,1] \times [0,1]$ which is covered by a uniform grid of 31×31 cells and the initial condition is

$$u(x,y,0) = \exp(-\beta((x-x_0)^2 + (y-y_0)^2)) \tag{7.2}$$

with $\beta = 31^2/18$, $x_0 = y_0 = 1/2$. For the first problem the velocities are $a = 1$ and $b = 0$, $\sigma = 0.5$ and the results studied after 62 timesteps. In the second problem, $a = 1$ and $b = 1$, $\sigma = 0.5$ and the number of time steps is 124.

The results are shown in Table 2 and Figure 3. The peak values obtained with PP2DU are slightly better than that of Thuburn's scheme. However, for Problem-2, both PP2D and PP2DU indicate slightly more distortion in the contours than TH1. This distortion can perhaps be reduced further by a more accurate choice of the initial slopes in (2.5).

TABLE 3
Advection of (7.4) *with velocity field* (7.3). $\sigma_{max} = 0.99$, 628 *time steps,* $\Delta x = \Delta y = 1$.

Scheme	Min. value	Max. value
MPDATA #4	0.000	3.263
MPDATA #3	0.000	3.261
PP2DU	0.000	3.463
PP2D	0.000	3.257
Superbee	-0.027	3.439
Average	-0.003	3.271
Minmod	0.000	2.770
Unlim.	-0.065	3.464

Example 4. In this example, we compare our results with the scheme of MPDATA [18] and its many recent enhancements. The initial condition here is a conical initial function that is in solid-body rotation. The domain is $[0,100] \times [0,100]$, which is covered by a uniform grid of 100×100 cells. The velocities are given by

$$\begin{aligned} a(x,y) &= -w\,(y - y_0), \\ b(x,y) &= +w\,(x - x_0), \end{aligned} \tag{7.3}$$

with $(x_0, y_0) = (50, 50)$ and $w = 0.1$. The initial condition can be written as

$$\begin{aligned} r &= ((x-x_c)^2 + (y-y_c)^2)^{1/2}, \\ u(x,y,0) &= 4\,(1 - r/15) \quad \text{for} \quad r < 15, \\ u(x,y,0) &= 0 \quad \text{for} \quad r \geq 15, \end{aligned} \tag{7.4}$$

with $x_c = 75$ and $y_c = 50$. The time step is chosen so that the maximum Courant number is 0.99. The results are studied after 628 time steps and are shown in Table 3. Contour plots of the solution after 3768 time steps are shown in Figure 4.

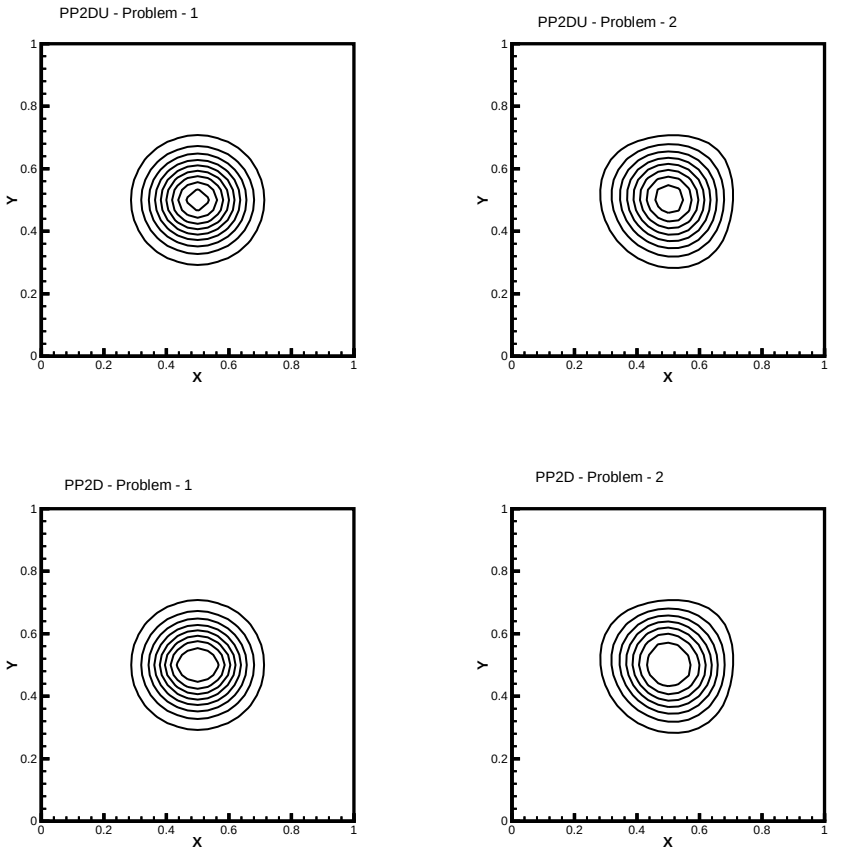


FIG. 3. *Advection of* (7.2) *on a* 31×31 *grid. Problem* 1: $a = 1$, $b = 0$, $\sigma = 0.5$, 62 *time steps; Problem* 2: $a = 1$, $b = 1$, $\sigma = 0.5$, 124 *time steps. The contours are from* $(0.1, 1.0)$ *with an increment of* 0.1. *Max. and Min. values are given in Table* 2.

From Table 3, the peak values of both PP2D and PP2DU are comparable to MPDATA. However, from Figure 4, both seem to have slightly larger distortion than the results of MPDATA.

Example 5. Our last problem is the well known double Mach reflection problem in two-dimensional gas dynamics. The computational domain is $[0, 4] \times [0, 1]$. The reflecting wall is from $(1/6, 0)$ to $(4, 0)$. Initially, a Mach 10 shock is incident on this wall at $(1/6, 0)$ making an angle of 60 degrees with the x-axis. To the right of the shock is undisturbed fluid of uniform pressure 1 and density 1.4. To the left of the shock, the conditions are

$$(\rho, u, v, p) = (8.0, 7.1447, -4.125, 116.5).$$

As the shock reflects off the wall, a diffraction pattern is formed. The final time is $t = 0.2$. A detailed description of the problem and various solutions can be found in [5]. The problem is solved on a uniform grid of 120×30 cells with $\sigma = 0.8$.

The boundary conditions are as follows: at the bottom, from $(0, 0)$ to $(1/6, 0)$, linear extrapolation; from $(1/6, 0)$ to $(4, 0)$, solid boundary; at the right, linear extrapolation; at the left, supersonic inflow; at the top, time-dependent conditions determined by the exact motion of the Mach 10 shock.

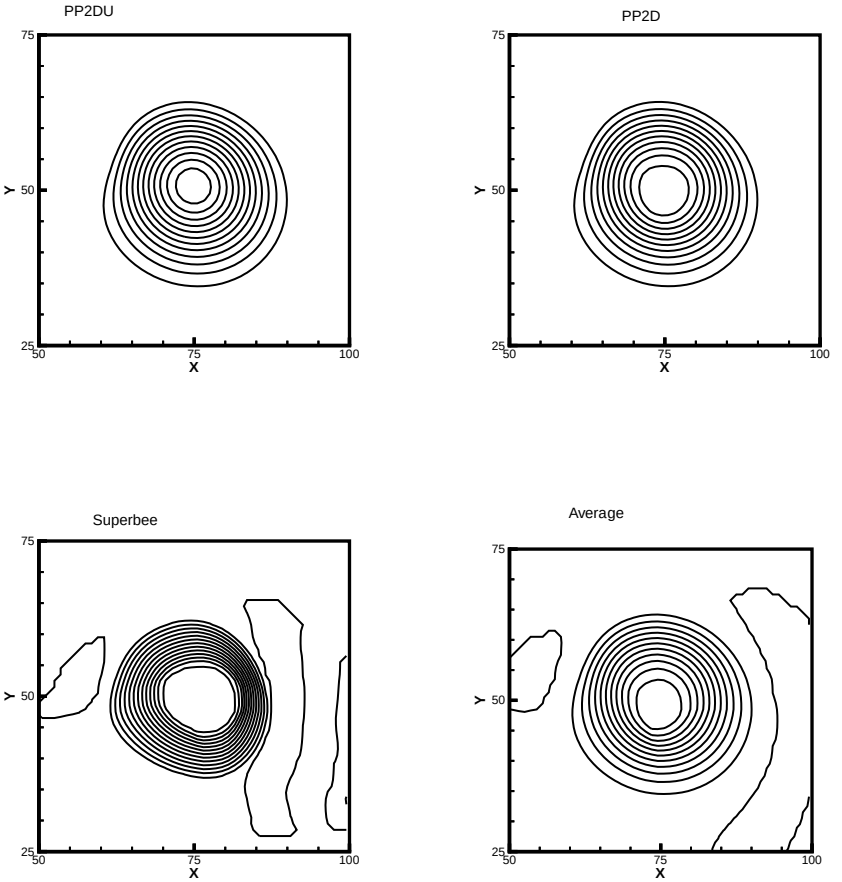


FIG. 4. *Advection of* (7.4) *on a* 100×100 *grid with velocity field* (7.3). $\sigma_{max} = 0.99$, 3768 *time steps. The contours are from* $(-0.25, 3.5)$ *with an increment of* 0.25.

The PP2DU, PP2D solutions are compared with the solution from the TVD scheme using the Average limiter in Figure 5, which shows contour plots of density. In this problem, there does not seem to be much difference between the three solutions.

8. Conclusions. In this paper, we have considered the problem of why the MUSCL–Hancock scheme with one-dimensional TVD limiters applied on a dimension-by-dimension basis gives rise to spurious oscillations near discontinuities on problems of two-dimensional advection. We have shown that if the reconstruction over the whole cell is bounded by its first order neighbors, then the MUSCL–Hancock scheme can be proved to be positivity preserving. An efficient limiter that achieves this and depends continuously on the data is also presented along with a fix for the loss of accuracy near extrema. Numerical experiments reveal that the new schemes perform well and are competitive with other methods for multidimensional advection.

Appendix. In this appendix, we recall a quick proof which states that in one dimension, the MUSCL–Hancock scheme with the slopes given by any of the one-dimensional TVD limiters of (2.8) does indeed satisfy (2.9) for $0 \leq \lambda_x \leq 1$. Using

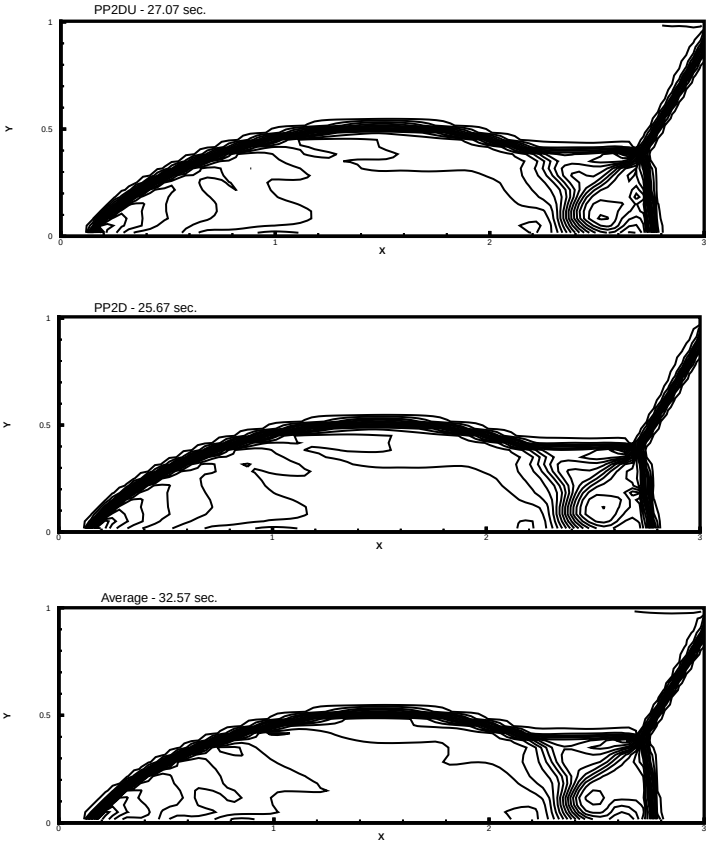


FIG. 5. *The double Mach reflection problem on a* 120×30 *grid with* $\sigma = 0.8$, $t = 0.2$. *The density contours are from* $(0.17, 21)$ *with an increment of* 0.6423.

(2.2) and (2.4) with $b = 0$ and $a > 0$, the update can be written as

$$u_i^{n+1} = (1 - \lambda_x)\left(u_i - \frac{1}{2}\lambda_x\, S_i^x\right) + \lambda_x\left(u_{i-1} + \frac{1}{2}(1 - \lambda_x)\, S_{i-1}^x\right). \tag{A.1}$$

If $u_i = u_{i-1}$, then both the slopes S_i^x and S_{i-1}^x are zero and the result follows. If not, by subtraction and division of the data, we need only consider the case where $u_{i-1} = 0$ and $u_i = 1$. In this case, for all limiters, both slopes are positive and bounded above by 2. This implies that $u_i - \frac{1}{2}\lambda_x\, S_i^x$ lies in $[0, 1]$ and $u_{i-1} + \frac{1}{2}(1 - \lambda_x)\, S_{i-1}^x$ also lies in $[0, 1]$. From the above update equation, the result follows.

It is worth noting that (2.9) can be proved in a more general setting, including reconstructions that are arbitrary functions. For a discussion, see Suresh [17].

Acknowledgments. The author would like to thank H. T. Huynh and D. Paxson for reviewing this paper. The author is also grateful to the Computing and Interdisciplinary Systems Office at Glenn Research Center for permission to publish this work.

REFERENCES

[1] A. Harten, B. Engquist, S. Osher, and S. R. Chakravarthy, *Uniformly high-order accurate essentially nonoscillatory schemes.* III, J. Comput. Phys., 71 (1987), pp. 231–303.

[2] P. L. Roe, *Characteristic-based schemes for the Euler equations,* Ann. Rev. Fluid Mech., 18 (1986), pp. 337–365.

[3] C. Hirsch, *Numerical Computation of Internal and External Flows,* John Wiley & Sons, New York, 1990.

[4] R. J. LeVeque, *Numerical Methods for Conservation Laws,* Birkhäuser-Verlag, Basel, 1990.

[5] P. Woodward and P. Colella, *The numerical simulation of two-dimensional fluid flow with strong shocks,* J. Comput. Phys., 54 (1984), pp. 115–173.

[6] B. Van Leer, *On the relation between the upstream-differencing schemes of Godunov, Engquists–Osher and Roe,* SIAM J. Sci. Statist. Comput., 5 (1984), pp. 1–20.

[7] P. Smolarkiewicz, *A fully multidimensional positive definite advection transport algorithm with small implicit diffusion,* J. Comput. Phys., 54 (1984), pp. 325–362.

[8] B. P. Leonard, M. K. MacVean, and A. P. Lock, *Positivity-Preserving Numerical Schemes for Multidimensional Advection,* NASA TM 106055, NASA Lewis Research Center, Cleveland, OH, 1993.

[9] B. Perthame and C. Shu, *On positivity preserving finite volume schemes for compressible Euler equations,* Numer. Math., 73 (1996), pp. 119–130.

[10] J. Thuburn, *Multidimensional flux-limited advection schemes,* J. Comput. Phys., 123 (1996), pp. 74–83.

[11] P. Batten, C. Lambert, and D. M. Causon, *Positively conservative high-resolution convection schemes for unstructured elements,* Internat. J. Numer. Methods Engrg., 39 (1996), pp. 1821–1838.

[12] T. Linde and P. Roe, *Robust Euler Codes,* AIAA paper 97-2098, AIAA, New York, 1997, pp. 83–93.

[13] T. Barth and D. C. Jesperson, *The Design and Application of Upwind Schemes on Unstructured Meshes,* AIAA paper 89-0366, AIAA, New York, 1989.

[14] S. Spekreijse, *Multigrid solution of monotone second-order discretizations of hyperbolic conservation laws,* Math. Comp., 49 (1987), pp. 135–155.

[15] A. Suresh and H. T. Huynh, *Accurate monotonicity-preserving schemes with Runge-Kutta time stepping,* J. Comput. Phys., 136 (1997), pp. 83–99.

[16] A. Suresh, *An Efficient Upwind Option for NPARC,* AIAA paper 95-1755, AIAA, New York, 1995.

[17] A. Suresh, *The Reconstruction Problem Revisited,* NASA TM 1999-209082, NASA Glenn Research Center, Cleveland, OH, 1999.

[18] L. Margolin and P. K. Smolarkiewicz, *Antidiffusive velocities for multipass donor cell advection,* SIAM J. Sci. Comput., 20 (1998), pp. 907–929.

SIAM J. SCI. COMPUT.
Vol. 22, No. 4, pp. 1199–1219
© 2000 Society for Industrial and Applied Mathematics

A DOMAIN DECOMPOSITION METHOD WITH LAGRANGE MULTIPLIERS AND INEXACT SOLVERS FOR LINEAR ELASTICITY*

AXEL KLAWONN† AND OLOF B. WIDLUND‡

Abstract. A new domain decomposition method with Lagrange multipliers for elliptic problems is introduced. It is based on a reformulation of the well-known finite element tearing and interconnecting (FETI) method as a saddle point problem with both primal and dual variables as unknowns. The resulting linear system is solved with block-structured preconditioners combined with a suitable Krylov subspace method. This approach allows the use of inexact subdomain solvers for the positive definite subproblems. It is shown that the condition number of the preconditioned saddle point problem is bounded independently of the number of subregions and depends only polylogarithmically on the number of degrees of freedom of individual local subproblems. Numerical results are presented for a plane stress cantilever membrane problem.

Key words. domain decomposition, Lagrange multipliers, finite element tearing and interconnecting, preconditioners, elliptic systems, finite elements

AMS subject classifications. 65F10, 65N30, 65N55

PII. S1064827599352495

1. Introduction.

In the past decade a great deal of research has been carried out on nonoverlapping domain decomposition methods using Lagrange multipliers. In these methods the original domain is decomposed into nonoverlapping subdomains. The continuity is then enforced by using Lagrange multipliers across the interface defined by the subdomain boundaries. A computationally quite efficient member of this class of domain decomposition algorithms is the finite element tearing and interconnecting (FETI) method introduced by Farhat and Roux [7]. In its original version, a Neumann problem is solved on each subdomain and the method is known to be scalable in the sense that its rate of convergence is independent of the number of subproblems. In a variant of the FETI method introduced in Farhat, Mandel, and Roux [6] an additional Dirichlet problem is solved exactly on each subdomain, in each iteration. This makes the rate of convergence of the iteration even less sensitive to the number of unknowns of the local problems. The use of inexact Dirichlet solvers is possible without a radical change of the FETI method. However, the use of inexact Neumann solvers does require a redesign of these algorithms; this is the topic of the present work.

In this paper, a new domain decomposition method with Lagrange multipliers is introduced by first reformulating the system of the FETI algorithm as a saddle point problem with both primal and dual variables. The resulting system is then solved

*Received by the editors February 26, 1999; accepted for publication (in revised form) May 1, 2000; published electronically October 18, 2000.
http://www.siam.org/journals/sisc/22-4/35249.html

†SCAI—Institute for Algorithms and Scientific Computing, GMD—German National Research Center for Information Technology, Schloss Birlinghoven, D–53754 Sankt Augustin, Germany (klawonn@gmd.de, http://www.gmd.de/SCAI/people/klawonn). The work of this author was supported in part by the DAAD (German Academic Exchange Service) within the program HSP III (Gemeinsames Hochschulsonderprogramm von Bund und Ländern).

‡Courant Institute of Mathematical Sciences, New York University, 251 Mercer Street, New York, NY 10012 (widlund@cs.nyu.edu, http://www.cs.nyu.edu/cs/faculty/widlund). The work of this author was supported in part by the National Science Foundation under grant NSF-CCR-9732208 and in part by the US Department of Energy under contract DE-FG02-92ER25127.

using block-structured preconditioners and a suitable Krylov subspace method. We can then avoid potentially quite costly direct solvers relying instead on any of a number of well-tested preconditioners for positive definite subproblems, such as incomplete LU methods, (algebraic) multigrid, etc. The good features of the FETI method such as scalability and efficiency are preserved. Essentially, the storage and factorization costs related to use of exact solvers in the FETI algorithm are eliminated while the new method will incur new costs for the preconditioners on the subdomains and additional storage of primal variable components in the iteration; we also have to expect that the iteration count will increase in comparison to when exact subdomain solvers are employed. A complete theory, based quite directly on the pioneering work of Mandel and Tezaur [16], and on earlier work by Klawonn [11], is also developed in this paper; see Theorems 9 and 15. In addition, we show how these results can be improved for a different family of preconditioners developed in [14]; see Theorems 10 and 16.

The remainder of this article is organized as follows. In section 2, we present the equations of linear elasticity and a finite element discretization thereof. In section 3, we review the FETI method and we develop our new method in section 4. In subsection 5.1, the convergence analysis of Mandel and Tezaur [16] is extended from scalar, second order elliptic equations to the system of equations of linear elasticity. A convergence analysis and condition number estimates for the block-diagonal preconditioner are given in subsection 5.2. The paper concludes with section 6, in which we report on some of our numerical experiments.

We note that a short conference paper [13] prepared previously describes and discusses a slightly different version of our main algorithm.

2. The elliptic problem. In this section, we introduce our model problem, the elliptic system arising from the displacement formulation of compressible, linear elasticity and its discretization by conforming finite elements.

2.1. The equations of linear elasticity. The equations of linear elasticity model the displacement of a linear elastic material under the action of external and internal forces. We denote the elastic body by $\Omega \subset \mathbf{R}^d, d = 2, 3$, and its boundary by $\partial\Omega$ and assume that one part of the boundary, Γ_0, is clamped, i.e., with homogeneous Dirichlet boundary conditions, and that the rest, $\Gamma_1 := \partial\Omega \setminus \Gamma_0$, is subject to a surface force $\mathbf{g}$, i.e., a natural boundary condition. We can also introduce a body force $\mathbf{f}$, e.g., gravity. The appropriate space for a variational formulation is the Sobolev space $H^1_{\Gamma_0}(\Omega) := \{\mathbf{v} \in H^1(\Omega)^d \ : \ \mathbf{v}_{|\Gamma_0} = 0\}$. The linear elasticity problem consists of finding the displacement $\mathbf{u} \in H^1_{\Gamma_0}(\Omega)$ of the elastic body Ω, such that

$$G \int_\Omega \varepsilon(\mathbf{u}) : \varepsilon(\mathbf{v}) dx \ + \ G\beta \int_\Omega \operatorname{div} \mathbf{u} \operatorname{div} \mathbf{v} \, dx = \langle \mathbf{F}, \mathbf{v} \rangle \quad \forall \ \mathbf{v} \in H^1_{\Gamma_0}(\Omega). \tag{1}$$

Here G and β are material parameters depending on the Poisson ratio ν and Young's modulus E with $\nu \in (0, 1/2]$ and $E > 0$. In the case of plane stress, we have $G = E/(1+\nu), \beta = \nu/(1-\nu)$, and for plane strain and three-dimensional elasticity we have $G = E/(1+\nu), \beta = \nu/(1-2\nu)$. Furthermore, $\varepsilon_{ij}(\mathbf{u}) := \frac{1}{2}(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i})$ is the linearized strain tensor, and

$$\varepsilon(\mathbf{u}) : \varepsilon(\mathbf{v}) = \sum_{i,j=1}^{d} \varepsilon_{ij}(\mathbf{u}) \varepsilon_{ij}(\mathbf{v}), \quad \langle \mathbf{F}, \mathbf{v} \rangle := \sum_{i=1}^{d} \int_\Omega f_i v_i \, dx \ + \ \sum_{i=1}^{d} \int_{\Gamma_1} g_i v_i d\sigma.$$

The associated bilinear form of linear elasticity is

$$a(\mathbf{u},\mathbf{v}) = G\int_\Omega \varepsilon(\mathbf{u}) : \varepsilon(\mathbf{v})dx \quad + \quad G\,\beta\int_\Omega \operatorname{div}\mathbf{u}\,\operatorname{div}\mathbf{v}\,dx.$$

In this article, we consider only the case of compressible elasticity. This means that the Poisson ratio ν is bounded away from $1/2$. We will also focus on the case of $d = 2$ in order to simplify our notation. We note that all our work extends easily to the case of $d = 3$ and to many other elliptic problems.

We will also use the standard Sobolev space norm

$$\|\mathbf{u}\|_{H^1(\Omega)} := \left(|\mathbf{u}|^2_{H^1(\Omega)} + \|\mathbf{u}\|^2_{L_2(\Omega)}\right)^{1/2}$$

with $\|\mathbf{u}\|^2_{L_2(\Omega)} := \int_\Omega |\mathbf{u}|^2 dx$, and $|\mathbf{u}|^2_{H^1(\Omega)} := \|\nabla\mathbf{u}\|^2_{L_2(\Omega)}$. It is obvious that the bilinear form $a(\cdot,\cdot)$ is continuous with respect to $\|\cdot\|_{H^1(\Omega)}$. However, proving ellipticity is far less trivial but it can be established from Korn's first inequality; see, e.g., Ciarlet [3].

LEMMA 1 (Korn's first inequality). *Let $\Omega \subset \mathbf{R}^d, d \geq 2$, be a Lipschitz domain. Then, there exists a positive constant $c = c(\Omega,\Gamma_0)$, such that*

$$\int_\Omega \varepsilon(\mathbf{u}) : \varepsilon(\mathbf{u})dx \geq c\,|\mathbf{u}|^2_{H^1(\Omega)} \quad \forall\ \mathbf{u} \in H^1_{\Gamma_0}(\Omega).$$

The well-posedness of the linear system (1) follows immediately from the continuity and ellipticity of the bilinear form $a(\cdot,\cdot)$.

It follows from Korn's first inequality that $\|\varepsilon(\mathbf{u})\|_{L_2(\Omega)}$ is equivalent to $\|\mathbf{u}\|_{H^1(\Omega)}$ on $H^1_{\Gamma_0}(\Omega)$. This result is not directly valid for the case of pure natural boundary conditions when we are working with the space $(H^1(\Omega))^d$. This case is of interest when considering interior subregions, i.e., those that do not touch Γ_0. However, a Gårding inequality is provided by Korn's second inequality

LEMMA 2 (Korn's second inequality). *Let $\Omega \subset \mathbf{R}^d, d \geq 2$, be a Lipschitz domain with diameter one. Then, there exists a positive constant $c = c(\Omega)$, such that*

$$\int_\Omega \varepsilon(\mathbf{u}) : \varepsilon(\mathbf{u})dx + \|\mathbf{u}\|^2_{L_2(\Omega)} \geq c\,\|\mathbf{u}\|^2_{H^1(\Omega)} \quad \forall\ \mathbf{u} \in (H^1(\Omega))^d.$$

There are several proofs; see, e.g., Nitsche [19].

We can now derive a Korn inequality on the space $\{\mathbf{u} \in (H^1(\Omega))^d : \mathbf{u} \perp \ker(\varepsilon)\}$. The null space $\ker(\varepsilon)$ is the space of rigid body motions. Thus, for $d = 2$, the linearized strain tensor of $\mathbf{u}$ and its divergence vanish only for the elements of the space spanned by the two translations

$$\mathbf{r}_1 := \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \qquad \mathbf{r}_2 := \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

and the single rotation

$$\mathbf{r}_3 := \begin{bmatrix} x_2 \\ -x_1 \end{bmatrix}.$$

In three dimensions the corresponding space is spanned by three translations and three rotations.

For convenience, we also introduce the notation

$$(\varepsilon(\mathbf{u}),\varepsilon(\mathbf{v}))_{L_2(\Omega)} := \int_\Omega \varepsilon(\mathbf{u}) : \varepsilon(\mathbf{v}) dx$$

and introduce two inner products on $(H^1(\Omega))^2$, for a region Ω with diameter one,

$$(\mathbf{u},\mathbf{v})_{E_1} := (\varepsilon(\mathbf{u}),\varepsilon(\mathbf{v}))_{L_2(\Omega)} + (\mathbf{u},\mathbf{v})_{L_2(\Omega)}$$

and

$$(\mathbf{u},\mathbf{v})_{E_2} := (\varepsilon(\mathbf{u}),\varepsilon(\mathbf{v}))_{L_2(\Omega)} + \sum_{i=1}^{3} l_i(\mathbf{u}) l_i(\mathbf{v}),$$

where the $l_i(\cdot)$ are defined by

$$l_i(\mathbf{u}) := \int_\Omega (\mathbf{r}_i)^t \mathbf{u} dx.$$

By using Lemma 2, $\|\cdot\|_{E_1}$, given by the inner product $(\cdot,\cdot)_{E_1}$, is a norm and not just a seminorm, and so, by construction, is $\|\cdot\|_{E_2}$.

The norms, just introduced, are equivalent.

LEMMA 3. *There exist constants $0 < c \le C < \infty$, such that*

$$c\,\|\mathbf{u}\|_{E_1} \le \|\mathbf{u}\|_{E_2} \le C\,\|\mathbf{u}\|_{E_1} \quad \forall\ \mathbf{u} \in (H^1(\Omega))^2.$$

Proof. The proof of the right inequality follows immediately from the Cauchy–Schwarz inequality. The left inequality is proven by contradiction and by using Rellich's theorem as in a proof of generalized Poincaré–Friedrichs inequalities, cf., e.g., Nečas [18, Chap. 2.7]. ∎

We obviously have

$$\|\varepsilon(\mathbf{u})\|_{L_2(\Omega)} \le \|\nabla \mathbf{u}\|_{L_2(\Omega)} \quad \forall\ \mathbf{u} \in (H^1(\Omega))^2. \tag{2}$$

Using (2) and Lemmas 2 and 3, we obtain the following.

LEMMA 4. *There exists a constant $0 < c$, such that*

$$c\,\|\nabla \mathbf{u}\|_{L_2(\Omega)} \le \|\varepsilon(\mathbf{u})\|_{L_2(\Omega)} \le \|\nabla \mathbf{u}\|_{L_2(\Omega)} \quad \forall\ \mathbf{u} \in (H^1(\Omega))^2, \mathbf{u} \perp \ker(\varepsilon).$$

2.2. Finite elements and the discrete problem. Since we consider only compressible elastic materials, it follows from Lemma 1 that the bilinear form $a(\cdot,\cdot)$ is uniformly elliptic. We can therefore successfully discretize the system (1) with low-order, conforming finite elements, such as linear or bilinear elements.

We assume that a triangulation τ^h of Ω is given which is shape regular and has a typical element diameter of h. We denote by $\mathbf{W}^h(\Omega) \subset H^1_{\Gamma_0}(\Omega)$ the corresponding conforming space of finite element functions, e.g., piecewise linear or bilinear continuous functions. Thus, it is our goal to solve the discrete problem

$$a(\mathbf{u}_h,\mathbf{v}_h) = \langle \mathbf{F},\mathbf{v}_h\rangle \quad \forall\ \mathbf{v}_h \in \mathbf{W}^h(\Omega). \tag{3}$$

In what follows, we work exclusively with the discrete problem and we drop the subscript h from now on.

3. A review of the FETI method. In this section, we give a brief review of the original FETI method introduced in Farhat and Roux [7] and the variant with a Dirichlet preconditioner introduced in Farhat, Mandel, and Roux [6]. For more detailed descriptions and proofs, we refer to [4, 5, 17, 22] and the references therein.

Let the domain $\Omega \subset \mathbf{R}^2$ be decomposed into N nonoverlapping subdomains $\Omega_i, i = 1, \ldots, N$, each of which is the union of elements and such that the finite element nodes on the boundaries of neighboring subdomains match across the interface $\Gamma := (\bigcup_{i=1}^N \partial\Omega_i) \setminus \partial\Omega$. Let the corresponding conforming finite element spaces be $\mathbf{W}_i = \mathbf{W}^h(\Omega_i), i = 1, \ldots, N$, and let $\mathbf{W} := \prod_{i=1}^N \mathbf{W}_i$ be the associated product space. When it is necessary to use vectors of nodal values, which define the elements of a finite element space, we underline; e.g., $\underline{\mathbf{W}}$ is the product space that corresponds to $\mathbf{W}$. Analogously, we denote the vector of nodal values associated with the finite element function $\mathbf{u}$ by $\underline{\mathbf{u}}$.

For each subdomain $\Omega_i, i = 1, \ldots, N$, we assemble the local stiffness matrices K_i and local load vectors $\underline{\mathbf{f}}_i$ by integrating the appropriate expressions over individual subdomains. We denote the local vectors of nodal values by $\underline{\mathbf{u}}_i$.

We can now formulate a minimization problem with constraints given by the intersubdomain continuity conditions.

Find $\underline{\mathbf{u}} \in \underline{\mathbf{W}}$, such that

$$\left.\begin{array}{c} J(\underline{\mathbf{u}}) := \frac{1}{2}\underline{\mathbf{u}}^t K \underline{\mathbf{u}} - \underline{\mathbf{f}}^t \underline{\mathbf{u}} \rightarrow \ \min \\ B\underline{\mathbf{u}} = 0 \end{array}\right\}, \tag{4}$$

where

$$\underline{\mathbf{u}} = \begin{bmatrix} \underline{\mathbf{u}}_1 \\ \vdots \\ \underline{\mathbf{u}}_N \end{bmatrix}, \quad \underline{\mathbf{f}} = \begin{bmatrix} \underline{\mathbf{f}}_1 \\ \vdots \\ \underline{\mathbf{f}}_N \end{bmatrix}, \quad \text{and } K = \begin{bmatrix} K_1 & O & \cdots & O \\ O & K_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & O \\ O & \cdots & O & K_N \end{bmatrix}.$$

The matrix $B = [B_1, \ldots, B_N]$ is constructed from $\{0, 1, -1\}$ such that the values of the solution $\underline{\mathbf{u}}$, associated with more than one subdomain, coincide when $B\underline{\mathbf{u}} = 0$. The local stiffness matrices K_i are positive semidefinite. The problem (4) is uniquely solvable if and only if $\ker(K) \cap \ker(B) = \{0\}$, i.e., K is invertible on the null space of B. This condition holds since the original finite element model is elliptic.

By introducing a vector of Lagrange multipliers $\underline{\lambda}$ to enforce the constraint $B\underline{\mathbf{u}} = 0$, we obtain a saddle point formulation of (4):

Find $(\underline{\mathbf{u}}, \underline{\lambda}) \in \underline{\mathbf{W}} \times \underline{\mathbf{U}}$, such that

$$\left.\begin{array}{rcrcl} K\underline{\mathbf{u}} & + & B^t\underline{\lambda} & = & \underline{\mathbf{f}} \\ B\underline{\mathbf{u}} & & & = & 0 \end{array}\right\}. \tag{5}$$

We note that the solution $\underline{\lambda}$ of (5) is in general only unique up to an additive vector from $\ker(B^t)$. The space of Lagrange multipliers $\underline{\mathbf{U}}$ is therefore chosen as the range (B); see also discussion below.

We will also use a full rank matrix, built from the rigid body motions of the subdomains

$$R = \begin{bmatrix} R_1 & O & \cdots & O \\ O & R_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & O \\ O & \cdots & O & R_N \end{bmatrix},$$

such that range $(R) = \ker(K)$; the subdomains with nonsingular stiffness matrices do not contribute any columns to R.

The solution of the first equation in (5) exists if and only if $\underline{\mathbf{f}} - B^t\underline{\lambda} \in \text{range}\,(K)$; this constraint will lead to the introduction of a projection P. We obtain

$$\underline{\mathbf{u}} = K^\dagger(\underline{\mathbf{f}} - B^t\underline{\lambda}) + R\underline{\alpha} \text{ if } \underline{\mathbf{f}} - B^t\underline{\lambda} \perp \ker(K), \tag{6}$$

where $K^\dagger$ is a pseudoinverse of K and $\underline{\alpha}$ has to be determined; see the discussion below.

We assume, for the time being, that B has full rank; i.e., the constraints are linearly independent. Substituting $\underline{\mathbf{u}}$ into the second equation of (5) gives

$$BK^\dagger B^t\underline{\lambda} = BK^\dagger\underline{\mathbf{f}} + BR\underline{\alpha}.$$

By considering the component orthogonal to BR, we find that

$$\left.\begin{array}{rcl} PF\underline{\lambda} & = & P\underline{\mathbf{d}} \\ G^t\underline{\lambda} & = & \underline{\mathbf{e}} \end{array}\right\} \tag{7}$$

with $G := BR, F := BK^\dagger B^t, \underline{\mathbf{d}} := BK^\dagger\underline{\mathbf{f}}, P := I - G(G^tG)^{-1}G^t$, and $\underline{\mathbf{e}} := R^t\underline{\mathbf{f}}$. The second equation in (7) is a consequence of the orthogonality condition of (6). We note that P is an orthogonal projection from $\underline{\mathbf{U}}$ onto $\ker(G^t)$.

Any solution $\underline{\lambda}$ of (5) and (7) yields the same solution $\underline{\mathbf{u}}$ of (4) and (5) if $\underline{\alpha} := -(G^tG)^{-1}G^t(\underline{\mathbf{d}} - F\lambda)$ is chosen; see Mandel, Tezaur, and Farhat [17, Thm. 2.4].

We define the space of admissible increments by

$$\underline{\mathbf{V}} := \{\underline{\mu} \in \underline{\mathbf{U}} : \underline{\mu} \perp B\underline{\mathbf{w}} \quad \forall\, \underline{\mathbf{w}} \in \ker(K)\} = \ker(G^t).$$

The original FETI method is a conjugate gradient method in the space $\underline{\mathbf{V}}$ applied to

$$PF\underline{\lambda} = P\underline{\mathbf{d}}, \qquad \underline{\lambda} \in \underline{\lambda}_0 + \underline{\mathbf{V}} \tag{8}$$

with an initial approximation $\underline{\lambda}_0$ chosen such that $G^t\underline{\lambda}_0 = \underline{\mathbf{e}}$. To introduce preconditioned variants, let D be a diagonal matrix. Then, the preconditioner M^{-1}, introduced in Farhat, Mandel, and Roux [6], is of the form

$$M^{-1} = B \begin{bmatrix} O & O \\ O & D^{-1}SD^{-1} \end{bmatrix} B^t$$

with

$$S = \begin{bmatrix} S_1 & O & \cdots & O \\ O & S_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & O \\ O & \cdots & O & S_N \end{bmatrix}.$$

The matrix S is the Schur complement of K obtained by eliminating the interior degrees of freedom of each of the subdomains. This computation clearly can be carried out in parallel and results in the block-diagonal matrix S which operates only on the degrees of freedom on the subdomain boundaries. In the application of M^{-1} to a vector, N independent Dirichlet problems have to be solved in each iteration step; M^{-1} is therefore often called the Dirichlet preconditioner. The simplest choice for D

is the identity matrix; this choice is made for the original Dirichlet preconditioner as introduced in Farhat, Mandel, and Roux [6]. Another possibility, which leads to faster convergence, is to choose D as a diagonal matrix, where the diagonal elements equal the number of subdomains to which the interface node belongs. This multiplicity scaling (MS) is discussed in Rixen and Farhat [22, 23]; see also further discussion in [14].

To keep the search directions of this preconditioned conjugate gradient method in the space $\underline{\mathbf{V}}$, the application of the preconditioner M^{-1} has to be followed by another application of the projection P. Hence, the Dirichlet variant of the FETI method is the conjugate gradient algorithm applied to the preconditioned system

$$PM^{-1}PF\,\underline{\lambda} = PM^{-1}P\,\underline{\mathbf{d}}, \qquad \underline{\lambda} \in \underline{\lambda}_0 + \underline{\mathbf{V}}, \tag{9}$$

with an initial approximation $\underline{\lambda}_0$ chosen such that $G\underline{\lambda}_0 = \underline{\mathbf{e}}$. We note that, for $\underline{\lambda} \in \underline{\mathbf{V}}$, $PM^{-1}PF\,\underline{\lambda} = PM^{-1}P^tP^tFP\,\underline{\lambda}$ and that we can therefore view the operator on the left-hand side of the preconditioned FETI system as the product of two symmetric matrices as required for the conjugate gradient algorithm.

4. The block-diagonal preconditioner. Using the decomposition $\lambda = \lambda_0 + \mu$, with $\mu \in V$, we can rewrite (8) as

$$PBK^{\dagger}B^t\underline{\mu} = PBK^{\dagger}(\underline{\mathbf{f}} - B^t\underline{\lambda}_0). \tag{10}$$

Since $\underline{\mathbf{u}} = K^{\dagger}(\underline{\mathbf{f}} - B^t\underline{\lambda}) + R\underline{\alpha}$ and $B\underline{\mathbf{u}} = 0$, we see that the solution of (10) satisfies

$$\begin{bmatrix} K & B^t \\ PB & O \end{bmatrix} \begin{bmatrix} \underline{\mathbf{u}} \\ \underline{\mu} \end{bmatrix} = \begin{bmatrix} \underline{\mathbf{f}} - B^t\underline{\lambda}_0 \\ 0 \end{bmatrix}.$$

We note that the elimination of the displacement variables of this system, by applying $K^{\dagger}$ to the first equation, gives us (10). Using that $\underline{\mu} \in \underline{\mathbf{V}}$, i.e., $P\underline{\mu} = \underline{\mu}$, and that $P^t = P$, we can make the system matrix symmetric such that

$$\begin{bmatrix} K & (PB)^t \\ PB & O \end{bmatrix} \begin{bmatrix} \underline{\mathbf{u}} \\ \underline{\mu} \end{bmatrix} = \begin{bmatrix} \underline{\mathbf{f}} - B^t\underline{\lambda}_0 \\ 0 \end{bmatrix}. \tag{11}$$

The displacement variable is not uniquely defined by this system; we are enforcing $PB\underline{\mathbf{u}} = 0$ but not $B\underline{\mathbf{u}} = 0$. Given any solution $\underline{\mathbf{u}}$ of (11), we obtain a solution that satisfies all the requirements by computing $\underline{\mathbf{u}} - R(G^tG)^{-1}G^tB\underline{\mathbf{u}}$; this is quite similar to the choice of the null space component of (6).

For the solution of the saddle point problem (11), we propose a preconditioned conjugate residual method with a block-diagonal preconditioner. For a detailed description of this algorithm, see Hackbusch [9] or Klawonn [11, 12]. We note that this algorithm will be designed such that the first component of each iterate belongs to range (K).

Our preconditioner has the form

$$\mathcal{B} = \begin{bmatrix} \widehat{K} & O \\ O & \widehat{M} \end{bmatrix}.$$

Here $\widehat{K}$ is assumed to be symmetric and a good preconditioner for $K + D_H\,Q$,

where

$$Q = \begin{bmatrix} Q_1 & O & \cdots & O \\ O & Q_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & O \\ O & \cdots & O & Q_N \end{bmatrix},$$

with Q_i the mass matrices associated with the mesh on Ω_i and $D_H = \operatorname{diag}_{i=1}^{N}(H_i^{-2} I_i)$ is a diagonal matrix. Here H_i denotes the diameter of the subdomain Ω_i. We further assume that $\widehat{M}$ is symmetric and a good preconditioner for M; i.e., we assume there exist constants k_0, k_1, m_0, m_1 with $0 < c \leq m_0 \leq m_1 \leq C < \infty$ and $0 < c \leq k_0 \leq k_1 \leq C < \infty$, such that

$$\begin{array}{ccccc} k_0\, \underline{\mathbf{u}}^t (K + D_H Q) \underline{\mathbf{u}} & \leq & \underline{\mathbf{u}}^t \widehat{K} \underline{\mathbf{u}} & \leq & k_1\, \underline{\mathbf{u}}^t (K + D_H Q) \underline{\mathbf{u}} \quad \forall \underline{\mathbf{u}} \in \underline{\mathbf{W}}, \\ m_0\, \underline{\lambda}^t M \underline{\lambda} & \leq & \underline{\lambda}^t \widehat{M} \underline{\lambda} & \leq & m_1\, \underline{\lambda}^t M \underline{\lambda} \quad \forall \underline{\lambda} \in \underline{\mathbf{V}}. \end{array} \tag{12}$$

Here c and C are generic constants independent of, or only weakly dependent on, the mesh size. Because of the block-diagonal structure of K and M and the preconditioners, c and C do not depend on the number of subdomains.

A preconditioner is often said to be optimal when the constants c and C can be chosen to be independent of the mesh size and the number of subdomains. We also note that all that is required here are preconditioners for quite benign positive definite problems on individual subregions and that the bounds are independent of the number of subdomains.

From these assumptions it is clear that our preconditioner $\mathcal{B}$ is symmetric positive definite and thus it can be used with the preconditioned conjugate residual method. In order to have a computationally efficient preconditioner, we must also assume that $\widehat{K}^{-1}$ and $\widehat{M}^{-1}$ can be applied to a vector at a low cost.

To guarantee that the iterates belong to range (K), we make use of the orthogonal projection P_R onto range (K) which is given by

$$P_R := I - R(R^t R)^{-1} R^t.$$

Consequently, we have that range $(R) = \ker(K)$ and we note that P_R is a block matrix with a 3×3 block for each interior subdomain; the expense of applying P_R to a vector is therefore very modest.

Our domain decomposition method is now the conjugate residual algorithm applied to the preconditioned system

$$\mathcal{B}^{-1} \mathcal{A} \underline{\mathbf{x}} = \mathcal{B}^{-1} \underline{\mathbf{F}}$$

with

$$\mathcal{A} = \begin{bmatrix} K & (PB)^t \\ PB & O \end{bmatrix}, \qquad \mathcal{B}^{-1} = \begin{bmatrix} P_R \widehat{K}^{-1} P_R^t & O \\ O & P \widehat{M}^{-1} P^t \end{bmatrix}, \tag{13}$$

$$\underline{\mathbf{x}} = \begin{bmatrix} \underline{\mathbf{u}} \\ \underline{\mu} \end{bmatrix}, \qquad \underline{\mathbf{F}} = \begin{bmatrix} \underline{\mathbf{f}} - B^t \underline{\lambda}_0 \\ 0 \end{bmatrix}.$$

We note that it is easy to see that only two matrix-vector products with the projection P and one with the projection P_R are required in each step. We note that the iterates of the conjugate residual method belong to $\underline{\mathbf{W}}_R \times \underline{\mathbf{V}}$ with $\underline{\mathbf{W}}_R := \text{range}\,(K)$.

5. Analysis. In this section, we will work with both finite element functions and vectors of nodal values representing them. We will make no distinction between operators and their matrix representation.

We will use the spaces $\mathbf{W}$, $\mathbf{U}$, and $\mathbf{V}$, and we begin by defining a norm $\|\cdot\|_W$ and a seminorm $|\cdot|_W$ on $\mathbf{W}$ by

$$\|\mathbf{v}\|_W^2 := \sum_{i=1}^N \|\mathbf{v}_i\|_{H^1(\Omega_i)}^2, \qquad |\mathbf{v}|_W^2 := \sum_{i=1}^N |\mathbf{v}_i|_{H^1(\Omega_i)}^2,$$

where $\|\mathbf{v}_i\|_{H^1(\Omega_i)}^2 := |\mathbf{v}_i|_{H^1(\Omega_i)}^2 + \frac{1}{H_i^2}\|\mathbf{v}_i\|_{L_2(\Omega_i)}^2$ for $\mathbf{v} = [\mathbf{v}_1, \ldots, \mathbf{v}_N] \in \mathbf{W}$. We also need the orthogonal decomposition of $\mathbf{W}$ into $\mathbf{W}_R := \text{range}\,(K)$ and its orthogonal complement $\mathbf{W}_R^\perp := \ker(K)$, i.e.,

$$\mathbf{W} = \mathbf{W}_R \oplus \mathbf{W}_R^\perp.$$

For the analysis we need to consider the trace space $\mathbf{W}_\Gamma$ of $\mathbf{W}$. We equip the space $\mathbf{W}_\Gamma$ with the seminorm and norm

$$|\mathbf{w}_\Gamma|_{W_\Gamma}^2 := \sum_{i=1}^N |\mathbf{w}_{\Gamma,i}|_{H^{1/2}(\partial\Omega_i)}^2, \qquad \|\mathbf{w}_\Gamma\|_{W_\Gamma}^2 := |\mathbf{w}_\Gamma|_{W_\Gamma}^2 + \sum_{i=1}^N \frac{1}{H_i}\|\mathbf{w}_{\Gamma,i}\|_{L_2(\partial\Omega_i)}^2.$$

We also define

$$B_\Gamma : \mathbf{W}_\Gamma \to \mathbf{U} \quad B_\Gamma := B_{|\mathbf{w}_\Gamma}.$$

From the construction of the jump operator B it is clear that

$$B\mathbf{w} = B_\Gamma \mathbf{w}_\Gamma \text{ for } \mathbf{w} \in \mathbf{W} \text{ where } \mathbf{w}_\Gamma = \mathbf{w}_{|\Gamma},$$

since continuity is enforced only across the interface. As a consequence, we obtain $\ker(B^t) = \ker(B_\Gamma^t)$. A direct computation yields

$$F = BK^\dagger B^t = B_\Gamma S^\dagger B_\Gamma^t.$$

Let us now introduce a norm on $\mathbf{V}$ by $\|\lambda\|_V := |B_\Gamma^t \lambda|_{W_\Gamma}$. This defines a norm, and not only a seminorm, since $\text{range}\,(B_\Gamma^t) \perp \ker(S)$. We will also use the dual space $\mathbf{V}'$ with a norm defined by $\|\lambda\|_{V'} := \sup_{\mu \in V} \frac{(\lambda,\mu)}{\|\mu\|_V}$. For the sake of simplicity, we will identify the space $\mathbf{V}'$ with $\mathbf{V}$, but we will use both norms. The above definitions are essentially the same as in [16].

Let us also define the product space $\mathbf{X} := \mathbf{W} \times \mathbf{V}$ which we equip with the graph norm $\|\mathbf{x}\|_X := \sqrt{\|\mathbf{v}\|_W^2 + \|\lambda\|_{V'}^2}$ for $\mathbf{x} = (\mathbf{v}, \lambda) \in \mathbf{X}$. We also need the subspace $\mathbf{X}_R := \mathbf{W}_R \times \mathbf{V}$ with the same norm as $\mathbf{X}$.

5.1. FETI for linear elasticity. For scalar, second order elliptic equations, it has been shown by Mandel and Tezaur [16] that the condition number of the FETI method with the original Dirichlet preconditioner (D = I) satisfies

$$\kappa(PM^{-1}PF) \leq C\,(1 + \log(H/h))^3.$$

We note that $(H/h)^2$ is proportional to the number of degrees of freedom of a subdomain. An improved result for a family of Dirichlet preconditioners using multiplicity

scaling, which leads to a $C\,(1+\log(H/h))^2$ estimate, is given in Klawonn and Widlund [14]; this work is also extended to problems with discontinuous coefficients.

In this section, we extend the results of Mandel and Tezaur [16] from scalar elliptic equations to the system of linear elasticity. To be able to use the results of [16], we now introduce the vector-valued Laplacian to be used only in our analysis. We denote by

$$K_\Delta := \begin{bmatrix} K_{\Delta,1} & O & \cdots & O \\ O & K_{\Delta,2} & \ddots & \vdots \\ \vdots & \ddots & \ddots & O \\ O & \cdots & O & K_{\Delta,N} \end{bmatrix}$$

a block-diagonal matrix, where the local stiffness matrices $K_{\Delta,i}, i = 1,\dots,N$, are obtained from the discretization of the inner product $(\nabla\mathbf{u},\nabla\mathbf{u})_{H^1(\Omega_i)}$ using the same finite element space as for $a(\cdot,\cdot)$. We denote by S_Δ the Schur complement of K_Δ obtained by eliminating the interior variables of each Ω_i, just as in the case of K and S.

It is well known that

$$(S_\Delta \mathbf{u}_\Gamma, \mathbf{u}_\Gamma) = \min_{\substack{\mathbf{v}\in\mathbf{W}\\ \mathbf{v}_{|\Gamma}=\mathbf{u}_\Gamma}} (K_\Delta \mathbf{v},\mathbf{v}) = (K_\Delta \mathbf{v}_{\rm harm},\mathbf{v}_{\rm harm})$$

for $\mathbf{u}_\Gamma \in \mathbf{W}_\Gamma$. Here $\mathbf{v}_{\rm harm} \in \mathbf{W}$ is the discrete harmonic extension of $\mathbf{u}_\Gamma \in \mathbf{W}_\Gamma$ defined as the unique solution of

$$(K_\Delta \mathbf{v}_{\rm harm},\mathbf{v}) = 0 \ \ \forall\ \mathbf{v}\in\mathbf{W}_0 \text{ with } \mathbf{v}_{{\rm harm}|\Gamma} = \mathbf{u}_\Gamma,$$

where $\mathbf{W}_0$ is the subspace of $\mathbf{W}$ with elements that vanish on the interface Γ.

Returning to the elasticity case, it can also easily be seen that

$$(S\mathbf{u}_\Gamma, \mathbf{u}_\Gamma) = \min_{\substack{\mathbf{v}\in\mathbf{W}\\ \mathbf{v}_{|\Gamma}=\mathbf{u}_\Gamma}} (K\mathbf{v},\mathbf{v}) = (K\mathbf{v}_{\rm elast},\mathbf{v}_{\rm elast})$$

for $\mathbf{u}_\Gamma \in \mathbf{W}_\Gamma$. Here $\mathbf{v}_{\rm elast} \in \mathbf{W}$ is the extension of $\mathbf{u}_\Gamma \in \mathbf{W}_\Gamma$ with the smallest elastic energy defined as the unique solution of

$$(K\mathbf{v}_{\rm elast},\mathbf{v}) = 0 \ \ \forall\ \mathbf{v}\in\mathbf{W}_0 \text{ with } \mathbf{v}_{{\rm elast}|\Gamma} = \mathbf{u}_\Gamma.$$

Using these formulas and the continuity of $a(\cdot,\cdot)$, we find that for $\mathbf{u}_\Gamma \in \mathbf{W}_\Gamma$,

$$\begin{aligned}(S\mathbf{u}_\Gamma, \mathbf{u}_\Gamma) &= \min_{\substack{\mathbf{v}\in\mathbf{W}\\ \mathbf{v}_{|\Gamma}=\mathbf{u}_\Gamma}} (K\mathbf{v},\mathbf{v}) \\ &\le (K\mathbf{v}_{\rm harm},\mathbf{v}_{\rm harm}) \\ &\le C\,(K_\Delta \mathbf{v}_{\rm harm},\mathbf{v}_{\rm harm}) \\ &= C \min_{\substack{\mathbf{v}\in\mathbf{W}\\ \mathbf{v}_{|\Gamma}=\mathbf{u}_\Gamma}} (K_\Delta\mathbf{v},\mathbf{v}) \\ &= C(S_\Delta \mathbf{u}_\Gamma, \mathbf{u}_\Gamma).\end{aligned}$$

To bound $(S\mathbf{u}_\Gamma, \mathbf{u}_\Gamma)$ from below, we restrict ourselves to a subspace. Using the left inequality in Lemma 4, we obtain for $\mathbf{u}_\Gamma \in \text{range}\,(S)$,

$$\begin{aligned}(S_\Delta \mathbf{u}_\Gamma, \mathbf{u}_\Gamma) &= \min_{\substack{\mathbf{v}\in\mathbf{W}\\ \mathbf{v}_{|\Gamma}=\mathbf{u}_\Gamma}} (K_\Delta \mathbf{v}, \mathbf{v})\\ &\le (K_\Delta \mathbf{v}_{\text{elast}}, \mathbf{v}_{\text{elast}})\\ &\le C\,(K\mathbf{v}_{\text{elast}}, \mathbf{v}_{\text{elast}})\\ &= C \min_{\substack{\mathbf{v}\in\text{range}\,(K)\\ \mathbf{v}_{|\Gamma}=\mathbf{u}_\Gamma}} (K\mathbf{v}, \mathbf{v})\\ &= C(S\mathbf{u}_\Gamma, \mathbf{u}_\Gamma).\end{aligned}$$

Since $(S_\Delta \mathbf{u}_\Gamma, \mathbf{u}_\Gamma)$ and $|\mathbf{u}_\Gamma|^2_{W_\Gamma}$ are equivalent for $\mathbf{u}_\Gamma \in \text{range}\,(S)$, we have proven the following.

LEMMA 5. *There exist constants $0 < c \le C < \infty$, independent of the mesh size and the subdomain diameters, such that*

$$c\,|\mathbf{u}_\Gamma|^2_{W_\Gamma} \le (S\mathbf{u}_\Gamma, \mathbf{u}_\Gamma) \le C\,|\mathbf{u}_\Gamma|^2_{W_\Gamma} \quad \forall\ \mathbf{u}_\Gamma\ \in\ \text{range}\,(S).$$

The next lemma follows from Mandel and Tezaur [16, proof of Lem. 3.11].

LEMMA 6.

$$\begin{aligned}\inf_{\lambda\in\mathbf{V}} \sup_{\mathbf{w}_\Gamma\in\mathbf{W}_\Gamma} \frac{(\lambda, B_\Gamma \mathbf{w}_\Gamma)}{\|\lambda\|_{V'}\|\mathbf{w}_\Gamma\|_{W_\Gamma}} &\ge C\,(1+\log(H/h))^{-1/2},\\ \sup_{\lambda\in\mathbf{V}} \sup_{\mathbf{w}_\Gamma\in\mathbf{W}_\Gamma} \frac{(\lambda, B_\Gamma \mathbf{w}_\Gamma)}{\|\lambda\|_{V'}\|\mathbf{w}_\Gamma\|_{W_\Gamma}} &\le C\,(1+\log(H/h)).\end{aligned}$$

The next result is essentially an extension of that lemma to the system of equations of linear elasticity.

LEMMA 7. *There exist constants $0 < c \le C < \infty$, independent of the mesh size and the number of subdomains, such that*

$$c(1+\log(H/h))^{-1}\|\lambda\|^2_{V'} \le (F\lambda, \lambda) \le C(1+\log(H/h))^2\|\lambda\|^2_{V'} \quad \forall\ \lambda \in V.$$

Proof. As shown in [16, proof of Lem. 3.11], we have for $\lambda \in V$

$$(F\lambda, \lambda) = \sup_{\mathbf{w}_\Gamma\in\text{range}\,(S)} \frac{(\lambda, B_\Gamma \mathbf{w}_\Gamma)^2}{(S\mathbf{w}_\Gamma, \mathbf{w}_\Gamma)}.$$

Using Lemma 5 and that $B^t_\Gamma \lambda \perp \ker(S)$ for $\lambda \in \mathbf{V}$, we obtain

$$\begin{aligned}(F\lambda, \lambda) &\le C \sup_{\mathbf{w}_\Gamma\in\text{range}\,(S)} \frac{(\lambda, B_\Gamma \mathbf{w}_\Gamma)^2}{|\mathbf{w}_\Gamma|^2_{W_\Gamma}} \le C \sup_{\mathbf{w}_\Gamma\in\text{range}\,(S)} \frac{(\lambda, B_\Gamma \mathbf{w}_\Gamma)}{\inf_{\mathbf{z}_\Gamma\in\ker(S)} \|\mathbf{w}_\Gamma + \mathbf{z}_\Gamma\|^2_{W_\Gamma}}\\ &= C \sup_{\substack{\tilde{\mathbf{w}}_\Gamma = \mathbf{w}_\Gamma + \mathbf{z}_\Gamma \in \mathbf{W}_\Gamma\\ \mathbf{w}_\Gamma\in\text{range}\,(S), \mathbf{z}_\Gamma\in\ker(S)}} \frac{(\lambda, B_\Gamma \mathbf{w}_\Gamma)}{\|\mathbf{w}_\Gamma + \mathbf{z}_\Gamma\|^2_{W_\Gamma}} = C \sup_{\tilde{\mathbf{w}}_\Gamma\in\mathbf{W}_\Gamma} \frac{(\lambda, B_\Gamma \tilde{\mathbf{w}}_\Gamma)^2}{\|\tilde{\mathbf{w}}_\Gamma\|^2_{W_\Gamma}}.\end{aligned}$$

Analogously, we obtain

$$(F\lambda, \lambda) \ge c \sup_{\mathbf{w}_\Gamma\in\mathbf{W}_\Gamma} \frac{(\lambda, B_\Gamma \mathbf{w}_\Gamma)^2}{\|\mathbf{w}_\Gamma\|^2_{W_\Gamma}}.$$

The bounds of $(F\lambda, \lambda)$ now follow from Lemma 6. ∎

Combining the definitions of the (exact) Dirichlet preconditioner M^{-1} and of the norm $\|\cdot\|_V$ with Lemma 5, we obtain the next lemma.

LEMMA 8. *There exist constants $0 < c \leq C < \infty$, independent of the mesh size and the number of subdomains, such that*

$$c \, \|\lambda\|_V^2 \leq (M^{-1}\lambda, \lambda) \leq C \, \|\lambda\|_V^2 \quad \forall \, \lambda \in V.$$

A condition number estimate of $PM^{-1}PF$ follows easily from these estimates; cf. also [16]. The proof for the algorithm using an inexact Dirichlet solver proceeds along very similar lines.

THEOREM 9. *There exists a positive constant C, independent of the mesh size and the number of subdomains, such that*

$$\kappa(PM^{-1}PF) \leq C \, (1 + \log(H/h))^3,$$

where $\kappa(PM^{-1}PF) := \sigma_{max}/\sigma_{min}$ is the spectral condition number defined by the ratio of the largest and the smallest eigenvalue σ_{max} and σ_{min} of $PM^{-1}PF$.

Similarly, there exists a positive constant C, independent of the mesh size and the number of subdomains, such that

$$\kappa(P\widehat{M}^{-1}PF) \leq C \, (1 + \log(H/h))^3,$$

for any preconditioner $\widehat{M}^{-1}$ that is spectrally equivalent to the exact Dirichlet preconditioner.

We note that Tezaur [27] established a condition number estimate of the form $C(1 + \log(H/h))^2$ for an algebraic FETI method developed by Park and Felippa [20] and Park, Justino, and Felippa [21]; see also the discussion in Rixen et al. [24].

Recently, a modified family of Dirichlet preconditioners, which includes the one with multiplicity scaling for both redundant and nonredundant Lagrange multipliers was introduced and analyzed by the authors in [14]. Using redundant Lagrange multipliers amounts to choosing the maximal number of constraints, pairwise connecting all degrees of freedom which belong to the same point x, and using a Lagrange multiplier for each possible pair. In contrast, nonredundant Lagrange multipliers are obtained by choosing the minimal number which ensures continuity of the displacements at each point on the interface.

For the case of nonredundant Lagrange multipliers, our preconditioner is of the form

$$\widehat{M}_{nr}^{-1} := (BB^t)^{-1} B \begin{bmatrix} O & O \\ O & S \end{bmatrix} B^t (BB^t)^{-1} = (B_\Gamma B_\Gamma^t)^{-1} B_\Gamma S B_\Gamma^t (B_\Gamma B_\Gamma^t)^{-1}$$

and for the redundant case, we have

$$\widehat{M}_r^{-1} := M^{-1} = B \begin{bmatrix} O & O \\ O & D^{-1}SD^{-1} \end{bmatrix} B^t = B_\Gamma D^{-1} S D^{-1} B_\Gamma^t,$$

where D is defined by multiplicity scaling; cf. section 3. We note that in the case of discontinuous coefficients, the preconditioners $\widehat{M}_{nr}$ and $\widehat{M}_r$ have to be enhanced using a more elaborate scaling; we refer to [14] for a more detailed discussion.

In [14], condition number estimates of $C \, (1 + \log(H/h))^2$ are given for second order scalar elliptic equations for both the preconditioners $\widehat{M}_{nr}$ and $\widehat{M}_r$. We note that the construction and analysis in [14] also hold for problems with discontinuous coefficients. In [14, proof of Thm. 1], we show in the nonredundant case

$$\begin{aligned} \sup_{w_\Gamma \in \text{range}\,(S)} \frac{\langle \lambda, B_\Gamma w_\Gamma \rangle}{|w_\Gamma|_S^2} &\geq \langle \widehat{M}_{nr}\lambda, \lambda \rangle, \\ \sup_{w_\Gamma \in \text{range}\,(S)} \frac{\langle \lambda, B_\Gamma w_\Gamma \rangle}{|w_\Gamma|_S^2} &\leq C \, (1 + \log(H/h))^2 \, \langle \widehat{M}_{nr}\lambda, \lambda \rangle. \end{aligned} \tag{14}$$

Similar inequalities can be obtained using $\widehat{M}_r$. In that paper, [14, Lem. 3], we define a norm on V in terms of the preconditioner $\widehat{M}_{nr}$, i.e., $\|\lambda\|_V := \langle \widehat{M}_{nr}^{-1}\lambda, \lambda\rangle$; the roles of $\|\lambda\|_V$ and $\|\lambda\|_{V'}$ are reversed in [14] compared to this paper. The dual norm $\|\lambda\|_{V'}$ can now be defined, as before, using the new V-norm. A simple computation gives $\lambda\|_{V'}^2 = \langle \widehat{M}_{nr}\lambda, \lambda\rangle$. The same arguments are equally valid for $\widehat{M}_r$ in the redundant case.

Using the inf-sup and sup-sup estimates (14), Lemma 5, and the definition of $\|\lambda\|_{V'}$, we obtain a condition number estimate for the FETI preconditioners for linear elasticity, using the same arguments as in the proof of Theorem 9. We note that the improved condition number estimate is a result of the optimality of the inf-sup constant in the first inequality in (14).

THEOREM 10. *There exists a positive constant C, independent of the mesh size and the number of subdomains, such that*

$$\kappa(P\widehat{M}^{-1}PF) \le C\,(1+\log(H/h))^2,$$

where $\widehat{M}$ is either $\widehat{M}_r$ or $\widehat{M}_{nr}$.

5.2. Analysis of the block-diagonal preconditioner. In this section, we give a condition number estimate for the block-diagonal preconditioner for the systems of equations arising from linear elasticity. This results in a convergence estimate for the preconditioned conjugate residual method.

As shown in section 4, the system of equations (11) involves an operator from $\mathbf{W}\times\mathbf{V}$ onto itself. The component of $\mathbf{u}$ in $\ker(K)$ is determined after the completion of the iteration. It is therefore appropriate to consider the restriction of the operator $\mathcal{A}$ to the subspace $\mathbf{X}_R = \mathbf{W}_R \times \mathbf{V}$. Similarly, we can view the preconditioner $\mathcal{B}^{-1}$ as a mapping from $\mathbf{X}_R$ onto itself; see (13).

An upper bound for the convergence rate of the conjugate residual method can be given in terms of the condition number $\kappa(\mathcal{B}^{-1}\mathcal{A})$ of the preconditioned system. A theory of block-diagonal preconditioners for saddle point problems of different origins has been developed by several authors; see Rusten and Winther [25], Silvester and Wathen [26], Kuznetsov [15], and Klawonn [12]. To the best of our knowledge, the first proof for block-preconditioners applied to saddle point problems with a singular block K is given in Klawonn [10, 11] and independently in Arnold, Falk, and Winther [1]. In order to obtain a condition number estimate for $\mathcal{B}^{-1}\mathcal{A}$, we follow the short argument given in [1] which is in the same spirit as that given by Mandel and Tezaur [16, Lem. 3.1] for the positive definite case. For completeness, we include the short proof here using our notation and the norms of X_R and its dual. Denoting by $\rho(\cdot)$ the spectral radius of a matrix, we find

$$\begin{aligned}\kappa(\mathcal{B}^{-1}\mathcal{A}) &= \rho(\mathcal{B}^{-1}\mathcal{A})\rho((\mathcal{B}^{-1}\mathcal{A})^{-1})\\ &\le \|\mathcal{B}^{-1}\mathcal{A}\|_{X_R\to X_R}\|(\mathcal{B}^{-1}\mathcal{A})^{-1}\|_{X_R\to X_R}\\ &\le \|\mathcal{B}^{-1}\|_{X_R'\to X_R}\|\mathcal{A}\|_{X_R\to X_R'}\|\mathcal{B}\|_{X_R\to X_R'}\|\mathcal{A}^{-1}\|_{X_R'\to X_R}.\end{aligned}$$

Hence, we need estimates of the norms of the operators $\mathcal{B}, \mathcal{A}$, and their inverses. The next lemma is due to Brezzi [2] and is given here in our notation.

LEMMA 11. *Let $B : \mathbf{W}_R \to \mathbf{V}$ satisfy an inf-sup and a sup-sup condition, i.e.,*

$$\begin{aligned}\inf_{\lambda\in\mathbf{V}}\sup_{\mathbf{w}\in\mathbf{W}_R}\frac{(\lambda, B\mathbf{w})}{\|\lambda\|_{V'}\|\mathbf{w}\|_W} &\ge \beta_0,\\ \sup_{\lambda\in\mathbf{V}}\sup_{\mathbf{w}\in\mathbf{W}_R}\frac{(\lambda, B\mathbf{w})}{\|\lambda\|_{V'}\|\mathbf{w}\|_W} &\le \beta_1,\end{aligned} \tag{15}$$

where $\beta_0, \beta_1 > 0$.

Furthermore, let $K : \mathbf{W}_R \to \mathbf{W}_R$ *be a symmetric operator satisfying*

$$\begin{array}{rcll} |(K\mathbf{w}, \mathbf{v})| & \leq & \alpha_1 \|\mathbf{w}\|_W \|\mathbf{v}\|_W & \forall\ \mathbf{w}, \mathbf{v} \in \mathbf{W}_R, \\ (K\mathbf{w}, \mathbf{w}) & \geq & \alpha_0 \|\mathbf{w}\|_W^2 & \forall\ \mathbf{w} \in \mathbf{W}_R, \end{array} \tag{16}$$

where $\alpha_0, \alpha_1 > 0$.

Then, $\mathcal{A} : \mathbf{X}_R \to \mathbf{X}_R$ *is an isomorphism with* $\|\mathcal{A}\|_{X_R \to X_R'} \leq \bar{C}(\alpha_1, \beta_1)$ *and* $\|\mathcal{A}^{-1}\|_{X_R' \to X_R} \leq \underline{C}(\alpha_0, \alpha_1, \beta_0)$, *where* $\bar{C}(\alpha_1, \beta_1) := \alpha_1 + \beta_1$ *and* $\underline{C}(\alpha_0, \alpha_1, \beta_0) := \max\{(\alpha_0^{-1} + \beta_0^{-1}(1 + \alpha_1/\alpha_0)), (\beta_0^{-1} + \alpha_1 \beta_0^{-2})(1 + \alpha_1/\alpha_0)\}$.

The uniform boundedness and ellipticity of K on $\mathbf{W}_R$ follow directly from the definition of the norm $\|\cdot\|_W$. Thus, we obtain constants $\alpha_0, \alpha_1 > 0$ which are independent of h, H.

We are left with showing the inf-sup and sup-sup conditions for B.

LEMMA 12.

$$\begin{array}{rcl} \inf\limits_{\lambda \in \mathbf{V}} \sup\limits_{\mathbf{w} \in \mathbf{W_R}} \dfrac{(\lambda, B\mathbf{w})}{\|\lambda\|_{V'} \|\mathbf{w}\|_W} & \geq & C\,(1 + \log(H/h))^{-1/2} =: \beta_0, \\ \sup\limits_{\lambda \in \mathbf{V}} \sup\limits_{\mathbf{w} \in \mathbf{W_R}} \dfrac{(\lambda, B\mathbf{w})}{\|\lambda\|_{V'} \|\mathbf{w}\|_W} & \leq & C\,(1 + \log(H/h)) =: \beta_1. \end{array}$$

Proof. For $\lambda \in \mathbf{V}$, let us now consider

$$\begin{aligned} \sup_{\mathbf{w} \in \mathbf{W_R}} \frac{(\lambda, B\mathbf{w})}{\|\mathbf{w}\|_W} &\leq \sup_{\mathbf{w} \in \mathbf{W_R}} \frac{(\lambda, B\mathbf{w})}{|\mathbf{w}|_W} \\ &\leq C \sup_{\mathbf{w} \in \mathbf{W_R}} \frac{(\lambda, B\mathbf{w})}{(K\mathbf{w}, \mathbf{w})^{1/2}} \\ &= C \sup_{\mathbf{w} \in \mathbf{W_R}} \frac{(\lambda, B_\Gamma \mathbf{w}_\Gamma)}{\inf\limits_{\substack{\mathbf{v} \in \mathbf{W} \\ \mathbf{v}|_\Gamma = \mathbf{w}_\Gamma}} (K\mathbf{v}, \mathbf{v})^{1/2}} \\ &= C \sup_{\mathbf{w}_\Gamma \in \mathrm{range}\,(S)} \frac{(\lambda, B_\Gamma \mathbf{w}_\Gamma)}{(S\mathbf{w}_\Gamma, \mathbf{w}_\Gamma)^{1/2}} \\ &\leq C \sup_{\mathbf{w}_\Gamma \in \mathbf{W}_\Gamma} \frac{(\lambda, B_\Gamma \mathbf{w}_\Gamma)}{\|\mathbf{w}_\Gamma\|_{W_\Gamma}} \\ &\leq \beta_1 \|\lambda\|_{V'} \end{aligned}$$

with $\beta_1 := C\,(1 + \log(H/h))$. The last inequality follows from Lemma 6. Analogously, we obtain

$$\sup_{\mathbf{w} \in \mathbf{W_R}} \frac{(\lambda, B\mathbf{w})}{\|\mathbf{w}\|_W} \geq \beta_0 \|\lambda\|_{V'}$$

with $\beta_1 := C\,(1 + \log(H/h))^{-1/2}$. ∎

Combining these results with Lemma 11, we obtain estimates of the norm of $\mathcal{A}$ and $\mathcal{A}^{-1}$ as follows.

LEMMA 13. *The operator* $\mathcal{A} : \mathbf{X}_R \to \mathbf{X}_R$ *is an isomorphism and satisfies*

$$\begin{aligned} \|\mathcal{A}\|_{X_R \to X_R'} &\leq C\,(1 + \log(H/h)), \\ \|\mathcal{A}^{-1}\|_{X_R' \to X_R} &\leq C\,(1 + \log(H/h)), \end{aligned}$$

where $C > 0$ is a generic constant independent of the mesh size and the subdomain diameters.

The next lemma provides bounds for $\|\mathcal{B}\|_{X_R \to X_R'}$ and $\|\mathcal{B}^{-1}\|_{X_R' \to X_R}$.

LEMMA 14. *There exist constants $0 < c \leq C < \infty$, which depend only on k_0, k_1, m_0, and m_1, such that*

$$\|\mathcal{B}\|_{X_R \to X_R'} \leq C, \quad \|\mathcal{B}^{-1}\|_{X_R' \to X_R} \leq \frac{1}{c}.$$

These bounds are uniform with respect to the mesh size and the number of subdomains if $\mathcal{B}$ is an optimal preconditioner, i.e., if the constants k_0, k_1, m_0, m_1 are uniformly bounded.

Proof. From the first inequalities of (12) and Korn's second inequality (Lemma 2), we obtain by a standard scaling argument that $(\widehat{K}\mathbf{u}, \mathbf{u})$ and $\|\mathbf{u}\|_W^2$ are spectrally equivalent $\mathbf{u} \in \mathbf{W}_R$. From the second inequalities in (12) and Lemma 5, it follows that for $\mu \in \mathbf{V}$

$$(\widehat{M}^{-1}\mu, \mu) \leq C\,(M^{-1}\mu, \mu) = C\,(B_\Gamma S B_\Gamma^t \mu, \mu) \leq C\,|B_\Gamma^t \mu|_{W_\Gamma}^2 = C\,\|\mu\|_V^2.$$

In the last inequality, we have used that $\mu \in \mathbf{V}$ implies $B_\Gamma^t \mu \in \text{range}\,(S)$. Analogously, we obtain

$$(\widehat{M}^{-1}\mu, \mu) \geq c\,\|\mu\|_V^2.$$

By using these inequalities, we get for $\lambda \in \mathbf{V}$

$$\|\lambda\|_{V'}^2 = \sup_{\mu \in V} \frac{(\lambda, \mu)^2}{\|\mu\|_V^2} \leq C \sup_{\mu \in V} \frac{(\lambda, \mu)^2}{(\widehat{M}^{-1}\mu, \mu)} = C \sup_{\nu \in V} \frac{(\lambda, \widehat{M}^{1/2}\nu)^2}{(\nu, \nu)} = C\,(\widehat{M}\lambda, \lambda)$$

and analogously

$$\|\lambda\|_{V'}^2 \geq c\,(\widehat{M}\lambda, \lambda).$$

From the definition of $\mathcal{B}$ follows

$$(\mathcal{B}\mathbf{x}, \mathbf{x}) = (\widehat{K}\mathbf{u}, \mathbf{u}) + (\widehat{M}\lambda, \lambda) \leq C(\|\mathbf{u}\|_W^2 + \|\lambda\|_{V'}^2) = C\,\|\mathbf{x}\|_X^2$$

with $\mathbf{x} = (\mathbf{u}, \lambda) \in \mathbf{X_R}$ and

$$(\mathcal{B}\mathbf{x}, \mathbf{x}) \geq c\,\|\mathbf{x}\|_X^2.$$

The boundedness of $\mathcal{B}$ and $\mathcal{B}^{-1}$ follows by using the following formulas:

$$\|\mathcal{B}\|_{X_R \to X_R'} = \sup_{\mathbf{x} \in \mathbf{X}_R} \frac{(\mathcal{B}\mathbf{x}, \mathbf{x})}{\|\mathbf{x}\|_X^2},$$

$$\|\mathcal{B}^{-1}\|_{X_R' \to X_R}^{-1} = \inf_{\mathbf{x} \in \mathbf{X}_R} \frac{(\mathcal{B}\mathbf{x}, \mathbf{x})}{\|\mathbf{x}\|_X^2}. \qquad \square$$

From Lemmas 13 and 14 the next theorem follows.

THEOREM 15.

$$\kappa(\mathcal{B}^{-1}\mathcal{A}) \leq C\,(1 + \log(H/h))^2$$

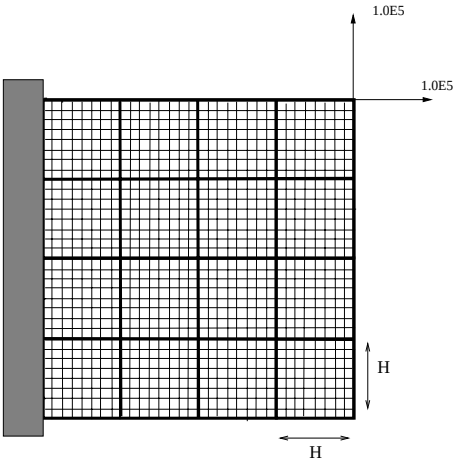


FIG. 1. *Sample domain decomposition of the cantilever with* 16 *subdomains.*

with a constant C *independent of* H, h.

We can also improve the results of Theorem 15 by using the results of [14], just as Theorem 10 represents an improvement of Theorem 9. We see that the inf-sup estimate used in the proof of Lemma 12 can be replaced with the improved constant bound given by the first inequality of (14). Revising the arguments in the proof of Theorem 15 in this respect, we obtain the following.

THEOREM 16. *There exists a constant* $C > 0$ *independent of* H, h *such that*

$$\kappa(\mathcal{B}^{-1}\mathcal{A}) \leq C\,(1 + \log(H/h)).$$

It is known that the convergence rate of the conjugate residual method depends linearly on the condition number, whereas that of the conjugate gradient algorithm does depend on its square root. Thus, from the improved estimates in Theorems 10 and 16, we see that both methods have an asymptotic convergence rate on the order of $(1 + \log(H/h))$. This is also reflected in the numerical experiments in section 6.

6. Numerical results. We have applied our domain decomposition method to a plane stress problem described in section 2. The Poisson ratio is $\nu = 0.3$ and the elasticity modulus $E = 2.1 \cdot 10^{11} N/m^2$, which models steel. The domain Ω is the unit square fixed on the left-hand side and free on the remainder of the boundary except for the upper-right corner element. The only nonzero components of the load vector are associated with the node at that corner; cf. Figure 1.

All our computations have been performed in MATLAB 5.3. Our Krylov subspace method is the preconditioned conjugate residual method with a zero initial guess. The stopping criterion is $\|\underline{\mathbf{r}}_n\|_2/\|\underline{\mathbf{r}}_0\|_2 < 10^{-6}$, where $\underline{\mathbf{r}}_n$ and $\underline{\mathbf{r}}_0$ are the nth and initial residuals.

Our domain Ω is decomposed into $N \times N$ square subdomains with $H := 1/N$; see Figure 1. In our implementation, we use redundant Lagrange multipliers; see the end of subsection 5.1 for a definition. This is known to yield a smaller number of iterations. An explanation from a mechanical viewpoint is given in Rixen and Farhat [23]; see also Klawonn and Widlund [14] for an analysis and connections between the cases of redundant and nonredundant Lagrange multipliers.

TABLE 1

Iteration counts for an increasing number of subdomains of constant size. (I) $\widehat{K} = K+1/H^2M_Q$ *and* $\widehat{M} = M$; (II) $\widehat{K} = K+1/H^2M_Q$ *and* $\widehat{M}$ *using* ILU(0); (III) $\widehat{K}$ ILU(10^{-3}) *of* $K+1/H^2M_Q$ *and* $\widehat{M} = M$; (IV) $\widehat{K}$ *using* ILU(10^{-3}) *of* $K+1/H^2M_Q$ *and* $\widehat{M}$ *using* ILU(0); (V) *FETI using Dirichlet preconditioner. MS : D = multiplicity scaling, Id : D = Identity.*

H/h = 8		Iter (I)		Iter (II)		Iter (III)		Iter (IV)		Iter (V)	
$1/h$	$1/H$	MS	Id	MS	Id	MS	Id	MS	Id	MS	Id
16	2	11	19	11	19	23	36	24	37	9	14
32	4	17	27	17	27	33	67	33	70	13	24
64	8	21	33	23	33	41	84	41	85	17	29
96	12	21	39	23	39	47	89	49	93	20	32
128	16	21	39	25	41	51	91	55	96	21	35

TABLE 2

Iteration counts for a constant number of subdomains of increasing size. (I) $\widehat{K} = K+1/H^2M_Q$ *and* $\widehat{M} = M$; (II) $\widehat{K} = K+1/H^2M_Q$ *and* $\widehat{M}$ *using* ILU(0); (III) $\widehat{K}$ ILU(10^{-3}) *of* $K+1/H^2M_Q$ *and* $\widehat{M} = M$; (IV) $\widehat{K}$ *using* ILU(10^{-3}) *of* $K+1/H^2M_Q$ *and* $\widehat{M}$ *using* ILU(0); (V) *FETI using Dirichlet preconditioner. MS : D = multiplicity scaling, Id : D = Identity.*

H=1/4		Iter (I)		Iter (II)		Iter (III)		Iter (IV)		Iter (V)	
$1/h$	H/h	MS	Id	MS	Id	MS	Id	MS	Id	MS	Id
16	4	15	25	15	25	28	61	28	60	12	22
32	8	17	27	17	27	33	67	33	70	13	24
64	16	19	29	19	29	51	99	58	102	15	27
128	32	19	33	23	47	69	140	79	145	18	31

We report on two different series of experiments for different combinations of preconditioners $\widehat{K}$ and $\widehat{M}$ in order to analyze the numerical scalability of the method. In our first set of runs, we keep the dimension of the subproblems, and H/h, fixed and increase the number of subdomains and thus the overall problem size. In a second series, we keep a fixed number of subdomains and increase the value of H/h resulting in a smaller h.

In order to see how our method behaves in the best possible case, we first report on results for $\widehat{K} = K + \frac{1}{H^2}Q$ and $\widehat{M} = M$; cf. Tables 1 and 2 and Figures 2 and 3. For both cases, we present results for $\widehat{M}$ constructed using $D = I$ as well as with the multiplicity scaling $D = MS$; cf. section 3. We also compare the iteration counts of our new method with those of the original FETI method using the Dirichlet preconditioner M; see Tables 1 and 2 and Figures 2 and 3. In the experiments with the FETI algorithm, we stop the iteration when the preconditioned projected residual is smaller than $10^{-6}\|f\|_2$; see Farhat and Roux [8]. In all cases, the convergence is considerably faster with MS; using this scaling the asymptotic convergence rate is also reached much earlier than for $D = I$. For both choices of D, we obtain scalable domain decomposition methods in both series of experiments.

To gain insight into the convergence behavior with inexact blocks $\widehat{K}$ and $\widehat{M}$, we use preconditioners based on an incomplete Cholesky factorization (ILU). In the following, ILU(0) stands for an incomplete Cholesky factorization with no fill in while ILU(tol) is a threshold ILU factorization, with a threshold of tol, as provided in MATLAB 5.3; any entry in a column of the Cholesky factor L is dropped if its magnitude is smaller than the drop tolerance tol times the norm of its column. We denote by $\hat{S}$ the matrix that replaces S when ILU(0) is used to solve the Dirichlet problems in each subdomain. Three different combinations are considered: 1. $\widehat{K} =$

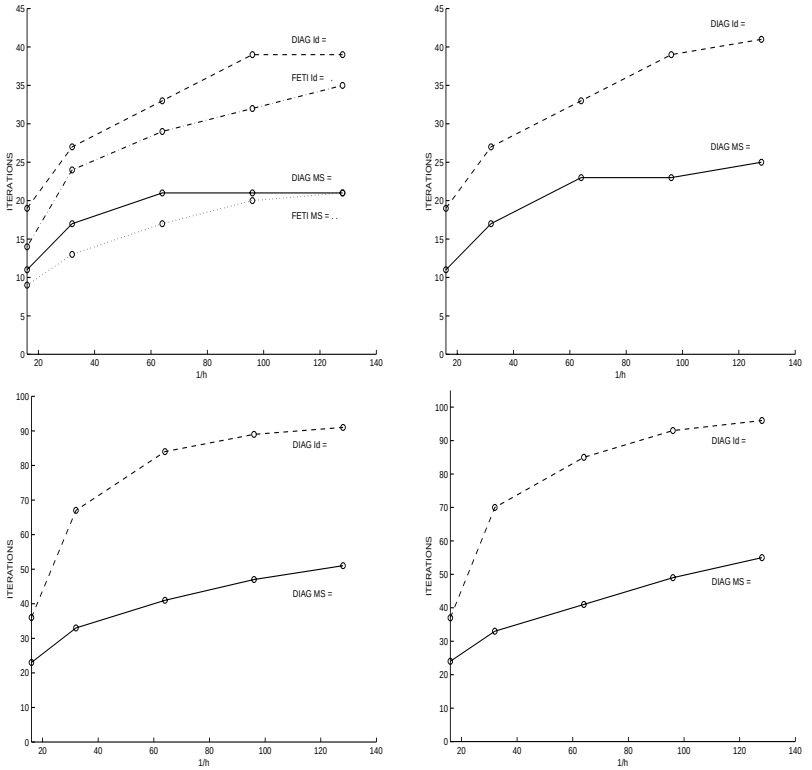


FIG. 2. *Iteration counts for an increasing number of subdomains of constant size.* $H/h = 8$. *Upper left:* $\widehat{K} = K + \frac{1}{H^2}Q, \widehat{M} = M$. *Upper right:* $\widehat{K} = K + 1/H^2 M_Q$ *and* $\widehat{M}$ *using* ILU(0). *Lower left:* $\widehat{K}$ ILU(10^{-3}) *of* $K + 1/H^2 M_Q$ *and* $\widehat{M} = M$. *Lower right:* $\widehat{K}$ *using* ILU(10^{-3}) *of* $K + 1/H^2 M_Q$ *and* $\widehat{M}$ *using* ILU(0).

$K + \frac{1}{H^2}Q$ and

$$\widehat{M}^{-1} = B \begin{bmatrix} O & O \\ O & D^{-1}\hat{S}D^{-1} \end{bmatrix} B^t;$$

2. $\widehat{K}$ is built with ILU(10^{-3}) applied to $K + \frac{1}{H^2}Q$ and $\widehat{M}^{-1} = M^{-1}$; 3. $\widehat{K}$ is built with ILU(10^{-3}) applied to $K + \frac{1}{H^2}Q$ and $\widehat{M}^{-1}$, as in combination 1., by using the inexact Schur complement $\hat{S}$. The computational results are given in Tables 1 and 2; cf. also Figures 2 and 3.

We also present results with a more accurate ILU decomposition based on a threshold tolerance $tol = 10^{-6}$ in Table 3. We see that a more accurate preconditioner $\widehat{K}$ improves the overall rate of convergence significantly. Comparing these results and the number of iterations obtained for the new method and the original FETI algorithm, both with the exact Dirichlet preconditioner, there is a strong indication that the increase in the iteration counts, which results when using the ILU(10^{-3}) preconditioner, would be largely eliminated if a better block-preconditioner, e.g., (algebraic) multigrid, were used as $\widehat{K}$.

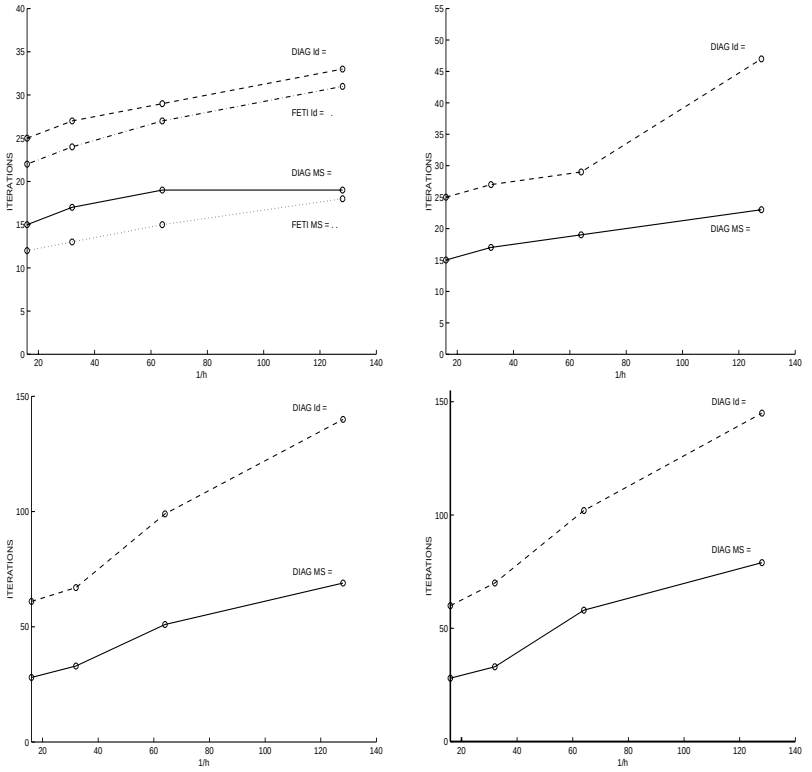


FIG. 3. *Iteration counts for a constant number of subdomains of increasing size.* $H = 1/4$. *Upper left:* $\widehat{K} = K + \frac{1}{H^2}Q, \widehat{M} = M$. *Upper right:* $\widehat{K} = K + 1/H^2 M_Q$ *and* $\widehat{M}$ *using* ILU(0). *Lower left:* $\widehat{K}$ ILU(10^{-3}) *of* $K + 1/H^2 M_Q$ *and* $\widehat{M} = M$. *Lower right:* $\widehat{K}$ *using* ILU(10^{-3}) *of* $K + 1/H^2 M_Q$ *and* $\widehat{M}$ *using* ILU(0).

TABLE 3

Iteration counts. (I) $\widehat{K}$ *based on* ILU(10^{-6}) *of* $K + 1/H^2 M_Q$ *and* $\widehat{M} = M$; (II) $\widehat{K}$ *based on* ILU(10^{-6}) *of* $K + 1/H^2 M_Q$ *and* $\widehat{M}$ *using* ILU(0). *MS : D = multiplicity scaling, Id : D = Identity.*

H/h = 8		Iter (I)		Iter (II)	
$1/h$	$1/H$	MS	Id	MS	Id
16	2	18	30	20	30
32	4	17	27	17	27
64	8	21	33	23	35
128	16	21	39	25	41

Acknowledgments. The first author would like to acknowledge the hospitality of and many discussions with Xiao-Chuan Cai, Charbel Farhat, and Daniel Rixen at the University of Colorado at Boulder and also wishes to thank Daniel Rixen for providing parts of his FETI MATLAB code.

REFERENCES

[1] D. N. ARNOLD, R. S. FALK, AND R. WINTHER, *Preconditioning discrete approximations of the Reissner–Mindlin plate model*, RAIRO Modél. Math. Anal. Numér., 31 (1997), pp. 517–557.

[2] F. Brezzi, *On the existence, uniqueness, and approximation of saddle-point problems arising from Lagrange multipliers*, RAIRO Modél. Math. Anal. Numér., 8 (1974), pp. 129–151.
[3] P. G. Ciarlet, *Mathematical Elasticity Volume* I*: Three-Dimensional Elasticity*, North–Holland, Amsterdam, 1988.
[4] C. Farhat, P. Chen, and J. Mandel, *A scalable Lagrange multiplier based domain decomposition method for time-dependent problems*, Internat. J. Numer. Methods Engrg., 38 (1995), pp. 3831–3853.
[5] C. Farhat, P. Chen, J. Mandel, and F.-X. Roux, *The two-level FETI method—Part* II*: Extensions to shell problems, parallel implementation and performance results*, Comput. Methods Appl. Mech. Engrg., 155 (1998), pp. 153–179.
[6] C. Farhat, J. Mandel, and F.-X. Roux, *Optimal convergence properties of the FETI domain decomposition method*, Comput. Methods Appl. Mech. Engrg., 115 (1994), pp. 367–388.
[7] C. Farhat and F.-X. Roux, *A method of finite element tearing and interconnecting and its parallel solution algorithm*, Internat. J. Numer. Methods Engrg., 32 (1991), pp. 1205–1227.
[8] C. Farhat and F.-X. Roux, *Implicit parallel processing in structural mechanics*, Comput. Mech. Adv., 2 (1994), pp. 1–124.
[9] W. Hackbusch, *Iterative Solution of Large Sparse Systems of Equations*, Springer, New York, 1994.
[10] A. Klawonn, *An Optimal Preconditioner for a Class of Saddle Point Problems with a Penalty Term, Part* II*: General Theory*, Technical report 14/95, Westfälische Wilhelms-Universität Münster, Germany, April 1995. Also available as Technical report 683, Courant Institute of Mathematical Sciences, New York University, New York, 1995.
[11] A. Klawonn, *Preconditioners for Indefinite Problems*, Ph.D. thesis, Westfälische Wilhelms-Universität Münster, 1995; Technical report 716, Courant Institute of Mathematical Sciences, New York University, New York, 1996.
[12] A. Klawonn, *An optimal preconditioner for a class of saddle point problems with a penalty term*, SIAM J. Sci. Comput., 19 (1998), pp. 540–552.
[13] A. Klawonn and O. B. Widlund, *A domain decomposition method with Lagrange multipliers for linear elasticity*, in Proceedings of the 11th International Conference on Domain Decomposition Methods, Greenwich, UK, 1998, C. H. Lai, P. E. Bjørstad, M. Cross, and O. B. Widlund, eds., Domain Decomposition Press, Bergen, Norway, 1998, pp. 49–56. Also available online from http://www.ddm.org/DD11/Klawonn.ps.gz.
[14] A. Klawonn and O. B. Widlund, *FETI and Neumann–Neumann iterative substructuring methods: Connections and new results*, Comm. Pure Appl. Math., to appear.
[15] Yu. A. Kuznetsov, *Efficient iterative solvers for elliptic finite element problems on nonmatching grids*, Russian J. Numer. Anal. Math. Modelling, 10 (1995), pp. 187–211.
[16] J. Mandel and R. Tezaur, *Convergence of a substructuring method with Lagrange multipliers*, Numer. Math., 73 (1996), pp. 473–487.
[17] J. Mandel, R. Tezaur, and C. Farhat, *A scalable substructuring method by Lagrange multipliers for plate bending problems*, SIAM J. Numer. Anal., 36 (1999), pp. 1370–1391.
[18] J. Nečas, *Les méthodes directes en théorie des équations elliptiques*, Academia, Prague, 1967.
[19] J. A. Nitsche, *On Korn's second inequality*, RAIRO Modél. Math. Anal. Numér., 15 (1981), pp. 237–248.
[20] K. C. Park and C. A. Felippa, *A Variational Framework for Solution Method Developments in Structural Mechanics*, Technical report CU-CAS Report 96-21, University of Colorado at Boulder, 1996.
[21] K. C. Park, M. R. Justino, and C. A. Felippa, *An algebraically partitioned FETI method for parallel structural analysis: Algorithm description*, Internat. J. Numer. Methods Engrg., 40 (1997), pp. 2717–2737.
[22] D. Rixen and C. Farhat, *Preconditioning the FETI and balancing domain decomposition methods for problems with intra- and inter-subdomain coefficient jumps*, in Proceedings of the Ninth International Conference on Domain Decomposition Methods in Science and Engineering, Bergen, Norway, 1996, P. Bjørstad, M. Espedal, and D. Keyes, eds., Domain Decomposition Press, Bergen, Norway, 1998, pp. 472–479. Also available online from http://www.ddm.org/DD9/Rixen.ps.gz.
[23] D. Rixen and C. Farhat, *A simple and efficient extension of a class of substructure based preconditioners to heterogeneous structural mechanics problems*, Internat. J. Numer. Methods Engrg., 44 (1999), pp. 489–516.
[24] D. Rixen, C. Farhat, R. Tezaur, and J. Mandel, *Theoretical comparison of the FETI and algebraically partitioned FETI methods, and performance comparisons with a direct sparse solver*, Internat. J. Numer. Methods Engrg., 46 (1999), pp. 501–534.
[25] T. Rusten and R. Winther, *A preconditioned iterative method for saddlepoint problems*,

SIAM J. Matrix Anal. Appl., 13 (1992), pp. 887–904.

[26] D. Silvester and A. Wathen, *Fast iterative solutions of stabilised Stokes systems Part* II*: Using general block preconditioners*, SIAM J. Numer. Anal., 31 (1994), pp. 1352–1367.

[27] R. Tezaur, *Analysis of Lagrange Multiplier Based Domain Decomposition*, Ph.D. thesis, University of Colorado at Denver, 1998; also available online from http://www-math.cudenver.edu/graduate/ thesis/rtezaur.ps.gz.

SIAM J. SCI. COMPUT.
Vol. 22, No. 4, pp. 1220–1237
© 2000 Society for Industrial and Applied Mathematics

QUASI-ORTHOGONAL GRIDS WITH IMPEDANCE MATCHING*

AHMED KHAMAYSEH† AND GLEN HANSEN†

Abstract. An elliptic, quasi-orthogonal grid generation system is formulated based on quasi-conformal mapping for arbitrary anisotropic (long and skinny) regions. The resulting system is a generalization of the well-known elliptic grid generation system derived from conformal mapping. Coupled with the grid generation system, the impedance-matching principle describes a methodology for preserving the discrete accuracy of the simulation, both internal to the domain and near internal geometric interfaces. Empirically, satisfying the impedance principle tends to minimize mesh effects on the solution results; meshes that are impedance-matched tend to reduce or eliminate spurious wave reflection and/or attenuation at internal interfaces. The resulting grid generation system is used to construct impedance-matched quasi-orthogonal grids on domains containing internal geometric constraints given an algebraic grid and a grid impedance function as initial conditions.

Key words. quasi-conformal mapping, elliptic grid generation, grid orthogonality, impedance matching

AMS subject classifications. 65M50, 51P05, 30C62

PII. S1064827599358613

1. Introduction.

In computational modeling using discrete methods, such as the solution of the Navier–Stokes equations on a domain discretized with a grid, the term "grid quality" is often used to describe how a particular grid lends itself to accurate solutions of the governing equations of interest. Indeed, the accuracy of the simulation is a strong function of how the grid matches the problem domain, domain boundaries, and the discretization technique employed. Ideally, one would choose a grid that preserves the accuracy of the discrete equations being solved on the domain, that provides the ability to capture the geometrical complexity and detail present, and that possesses sufficient resolution to capture the smallest length scale of interest to the simulation. Several grid quality metrics are in common use. A "good" grid may possess the following: orthogonality at the boundaries and quasi- orthogonality within critical regions, smoothness, bounded aspect ratios, solution adaptive behavior, etc. While traditional grid generation (and/or relaxation) techniques designed to optimize one or more of the above are used often very effectively, the desire to provide a stronger relationship between observed simulation anomalies and the grid generation/relaxation process remains.

Impedance is classically defined as the ratio of "pressure" to "displacement" (or voltage to current), in describing the response of a physical system. Examples include electrical impedance, acoustic impedance, and mechanical impedance. For the case of grid generation, one could consider the impedance of the grid to wave propagation in terms of establishing a mesh that provides a constant impedance for a chosen wave propagation force per unit area and wave displacement; a desirable grid is one that does not induce aphysical resistance to the wave. In general, the approach of selecting a grid whose characteristics provide the desired response to the physical system could

*Received by the editors July 14, 1999; accepted for publication (in revised form) March 29, 2000; published electronically October 18, 2000. The U.S. Government retains a nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or allow others to do so, for U.S. Government purposes. Copyright is owned by SIAM to the extent not limited by these rights.
http://www.siam.org/journals/sisc/22-4/35861.html

†Applied Physics Division, MS F645, Los Alamos National Laboratory, Los Alamos, NM 87545 (ahmed@lanl.gov, ghansen@lanl.gov).

also be described as selecting a grid whose "impedance" matches the "impedance" of the equations to be solved on the mesh. Thus, in this context, the concept of "impedance matching" will be defined as the use of terms or constraints that arise from the discretization of the governing equations to generate the grid.

Experience shows that impedance mismatches local to internal mesh boundaries (e.g., in the neighborhood of internal geometric boundaries between objects of different material properties) may result in spurious solution behavior, particularly when solution discontinuities are present. Study of this phenomena is warranted, as applications in computational field simulation frequently involve wave propagation as in, for example, the modeling of shock waves moving through solid materials. In this example, physical artifacts may take the form of spurious shock reflections or artificially occluded shocks at an interface. Experimentally, it is possible to reduce these effects with the placement of particular constraints on the mesh near internal interfaces. Global impedance mismatches, where the grid induces errors in a continuous manner within bulk regions, may result in wave diffusion that is not as easily identified in the simulation result.

This paper builds on published work by first presenting a general elliptic grid generation system based on quasi-conformal mapping for arbitrary anisotropic regions. This system may be used to construct quasi-orthogonal smooth grids with an externally specified spacing normal to the boundaries of the domain. Second, this system is improved with the use of the impedance-matching principle, applied to satisfy the above external constraint in the neighborhood of internal geometric interfaces.

2. Quasi-conformal grid generation. The preservation of the discrete accuracy of a simulation does not mandate absolute grid cell orthogonality. If this were a requirement, the types of geometry that could be accurately modeled would be very limited. This paper approaches orthogonality from a global perspective, where orthogonality is "encouraged" within the mesh, but not mandated. A robust grid generator is desired that provides boundary grid orthogonality, and some control over grid aspect ratio and skew angles within the mesh, while preserving simulation accuracy across internal interfaces. This section develops a quasi-elliptic grid generation system that addresses these criteria; this system is derived using quasi-conformal mapping.

Quasi-conformal mapping is viewed often as a generalization of conformal mapping (Ahlfors [1] and Krushkal [2]). Renelt [3] based his treatment of general elliptic systems of first-order partial differential equations entirely on quasi-conformal mapping. Likewise, Bers [4] used quasi-conformal mapping in the study of gas dynamics on the plane. Additionally, there are other classical applications of the conformal transformation of a surface onto a planar region and the reduction of elliptic partial differential equations to canonical form, proposed by Khamayseh and Mastin [5] and Mastin and Thompson [6], respectively.

The primary motivation for the use of quasi-conformal mapping in grid generation is the ease of expressing orthogonal curvilinear coordinate systems on simply connected two-dimensional regions (Godunov and Prokopov [7] and Arina [8]). The function $\mathbf{x}(\zeta) = x(\xi,\eta) + iy(\xi,\eta)$ describes a quasi-conformal mapping of a simply connected region $\mathcal{R}$ contained in the extended complex plane $\mathbb{C}_\infty$, provided that $\mathbf{x}$ maps $\mathcal{R}$ homeomorphically onto a region Ω, and $\mathbf{x}$ is a solution to Beltrami's equation

$$\mathbf{x}_{\bar{\zeta}} = \nu \mathbf{x}_{\zeta},$$

where $\bar{\zeta} = \xi - i\eta$ and $\zeta = \xi + i\eta$. The complex dilation $\nu(\xi,\eta)$ is a measurable function in $\mathcal{R}$ that satisfies the condition $||\nu||_\infty < 1$. By definition, the real and imaginary

parts of $\mathbf{x}$ are solutions to Beltrami's equations

$$\begin{aligned} x_\xi &= \gamma y_\eta + \beta y_\xi, \\ -x_\eta &= \alpha y_\xi + \beta y_\eta, \end{aligned}$$

where α, β, γ are functions of (ξ, η), $\alpha > 0$, $\gamma > 0$, and satisfy $\alpha\gamma - \beta^2 = 1$ in $\mathcal{R}$. This result is regarded as being a generalization to the Cauchy–Riemann equations. In this application, the Beltrami system may be further simplified. If $\beta = 0$, then $\nu = \frac{\mu-1}{\mu+1}$, where $\mu = \alpha = \frac{1}{\gamma}$. This system then reduces to

$$\begin{aligned} \mu x_\xi &= y_\eta, \\ x_\eta &= -\mu y_\xi. \end{aligned} \tag{2.1}$$

In this result, let $\mu = \sqrt{\frac{g_{22}}{g_{11}}}$, given the metrics $g_{11} = \mathbf{x}_\xi \cdot \mathbf{x}_\xi = x_\xi^2 + y_\xi^2$ and $g_{22} = \mathbf{x}_\eta \cdot \mathbf{x}_\eta = x_\eta^2 + y_\eta^2$. From (2.1), it follows that the coordinate functions $x(\xi, \eta)$ and $y(\xi, \eta)$ are solutions of the second-order quasi-linear elliptic system

$$\begin{aligned} \mu^2 \left(x_{\xi\xi} + \frac{\mu_\xi}{\mu} x_\xi \right) + \left(x_{\eta\eta} - \frac{\mu_\eta}{\mu} x_\eta \right) = 0, \\ \mu^2 \left(y_{\xi\xi} + \frac{\mu_\xi}{\mu} y_\xi \right) + \left(y_{\eta\eta} - \frac{\mu_\eta}{\mu} y_\eta \right) = 0. \end{aligned} \tag{2.2}$$

The quantities μ_ξ and μ_η may be derived from (2.1), yielding

$$\begin{aligned} \mu_\xi &= \frac{-\mu^2 \mathbf{x}_\xi \cdot \mathbf{x}_{\xi\xi} - \mathbf{x}_\xi \cdot \mathbf{x}_{\eta\eta}}{\mu g_{11}}, \\ \mu_\eta &= \frac{\mathbf{x}_\eta \cdot \mathbf{x}_{\eta\eta} + \mu^2 \mathbf{x}_\eta \cdot \mathbf{x}_{\xi\xi}}{\mu g_{11}}. \end{aligned} \tag{2.3}$$

Combining equations (2.1), (2.2), and (2.3) forms the basis for the quasi-orthogonal grid generation method. In this case, the coordinates x and y are solutions of the quasi-linear elliptic system

$$g_{22}(\mathbf{x}_{\xi\xi} + \mathcal{P}\mathbf{x}_\xi) + g_{11}(\mathbf{x}_{\eta\eta} + \mathcal{Q}\mathbf{x}_\eta) = 0, \tag{2.4}$$

where

$$\begin{aligned} \mathcal{P} &= \frac{\mu_\xi}{\mu} = -\frac{\mathbf{x}_\xi \cdot \mathbf{x}_{\xi\xi}}{g_{11}} - \frac{\mathbf{x}_\xi \cdot \mathbf{x}_{\eta\eta}}{g_{22}}, \\ \mathcal{Q} &= -\frac{\mu_\eta}{\mu} = -\frac{\mathbf{x}_\eta \cdot \mathbf{x}_{\eta\eta}}{g_{22}} - \frac{\mathbf{x}_\eta \cdot \mathbf{x}_{\xi\xi}}{g_{11}}. \end{aligned} \tag{2.5}$$

The source terms $\mathcal{P}$ and $\mathcal{Q}$ are called *control functions* in the grid generation literature. These control functions are often used to concentrate grid points in certain areas of the grid, such as along particular boundary segments. In this case, the control functions (2.5) will be used to provide additional control over the orthogonality of the mesh. This system is similar to the Thompson et al. quasi-elliptic method [9], given the condition $\mathbf{x}_\xi \cdot \mathbf{x}_\eta = 0$.

To complete the mathematical specification of the system (2.4), it is necessary to address the boundary conditions. Orthogonality of the grid lines at the boundary is desired, leading to the condition

$$g_{12} = \mathbf{x}_\xi \cdot \mathbf{x}_\eta = 0 \quad \text{on} \quad \partial\mathcal{R}. \tag{2.6}$$

3. Implementation of the grid generation system. In this section, the numerical procedures for implementing the quasi-elliptic grid generator (2.4) and the corresponding boundary conditions are detailed. The solution of this system constitutes an elliptic smoothing scheme for planar grids. This result is followed by a presentation of the effects and the methodologies of computing control functions in elliptic grid generation.

Standard central differencing in the ξ and η directions was used as the discretization technique of the system. An iterative method was used as a relaxation procedure to obtain $\mathbf{x}_{i,j} = (x_{i,j}, y_{i,j})$ for the interior nodes. An initial mapping $\mathbf{x}(\xi,\eta) = \big(x(\xi,\eta), y(\xi,\eta)\big)$ from computational space $\mathcal{R} = [0,m]\times[0,n]$ to the bounded physical domain $\Omega \subset \mathbb{R}^2$ was assumed. In this case, m and n are positive integers and *grid lines* are defined as the lines $\xi = i$ or $\eta = j$, with $0 \le i \le m$ or $0 \le j \le n$. The initial mapping $\mathbf{x}$ was obtained using transfinite interpolation, an algebraic grid generation method.

Thompson et al. [9] describes a general method for the construction of curvilinear structured grids that involve the solution of partial differential equations. In this work, given an initial mapping, the coordinate functions x and y are iteratively relaxed until they become solutions of the governing quasi-linear elliptic system. In this paper, successive overrelaxation (SOR) is used to solve the elliptic generation system (2.4).

To implement the SOR method first requires a formulation of the relaxation statement, which isolates the update quantity. In this case, given the discretized version of (2.4) and solving for $\mathbf{x}_{i,j}$ results in

$$\begin{aligned}\mathbf{x}_{i,j} = \frac{1}{4(\mathrm{g}_{11}+\mathrm{g}_{22})_{i,j}}\{&2\,(\mathrm{g}_{22})_{i,j}(\mathbf{x}_{i+1,j}+\mathbf{x}_{i-1,j})\\ &+(\mathrm{g}_{22})_{i,j}\mathcal{P}_{i,j}(\mathbf{x}_{i+1,j}-\mathbf{x}_{i-1,j})\\ &+2(\mathrm{g}_{11})_{i,j}(\mathbf{x}_{i,j+1}+\mathbf{x}_{i,j-1})\\ &+(\mathrm{g}_{11})_{i,j}\mathcal{Q}_{i,j}(\mathbf{x}_{i,j+1}-\mathbf{x}_{i,j-1})\}.\end{aligned} \tag{3.1}$$

To update the solution through the iterative method, the values of the physical coordinates given by (3.1) are applied through an acceleration process

$$\mathbf{x}_{i,j}^{k+1} = \omega_{i,j}\mathbf{x}_{i,j}^{k+1} + (1-\omega_{i,j})\mathbf{x}_{i,j}^{k},$$

where $\omega_{i,j}$ is the acceleration parameter. In practice, for most geometries, the choice of $\omega_{i,j} = 1$ leads to effective convergence. For certain highly curved geometries, the system (2.4) becomes progressively stiffer, requiring underrelaxation (choosing $0 < \omega_{i,j} < 1$) to attain convergence.

Prior to the relaxation process, boundary conditions must be specified. It is necessary to develop a discrete version of the orthogonality condition (2.6) on $\xi = 0, m$, and $\eta = 0, n$.

It is possible to impose orthogonality by adjusting the control functions near the boundary of the domain while holding the coordinates of the boundary points constant. In this case, the control functions (2.5) are evaluated at the boundaries then subsequently interpolated to the interior using linear transfinite interpolation. As described, these functions must be refreshed every iteration for the method to be effective. This approach was originally developed by Sorenson [10] for the imposition of boundary orthogonality on two-dimensional domains. The need for user specification

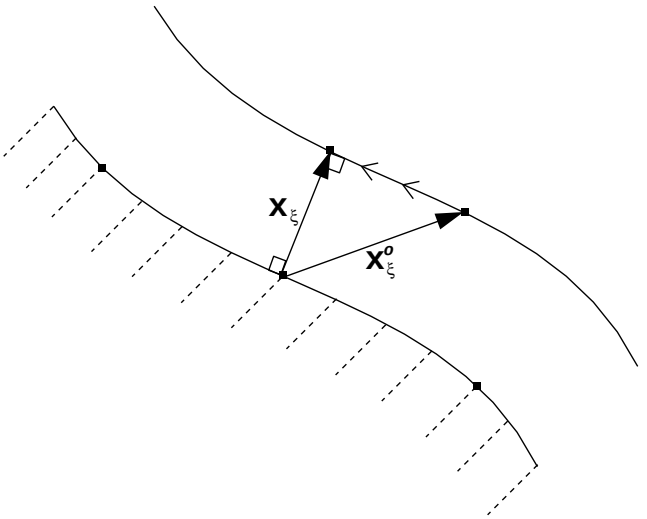


FIG. 3.1. *Projection of interior algebraic grid point to the orthogonal position.*

of grid spacing normal to the boundary in this method is a limitation. It is possible, however, to automatically derive normal grid spacing information from a sufficiently high quality initial algebraic grid.

A convenient way to infer the normal boundary spacing from the initial algebraic grid is to assume that the position of the first interior grid line adjacent to the boundary is acceptable. In practice, one usually desires that the relaxation procedure simply "swing" the boundary intersecting grid lines normal (orthogonal) to the boundary, without adjusting the positions of the grid lines parallel to the boundary (Figure 3.1). Clearly, this approach will provide boundary orthogonality without moving the actual boundary points or the boundary-adjacent grid lines established by the algebraic method.

Unfortunately, this approach also directly prescribes the positions of the first interior layer of grid points on the algebraic grid line adjacent to the boundary. This inflexibility directly leads to two areas of difficulty. First, the points $\mathbf{x}_{1,1}, \mathbf{x}_{m-1,1}, \mathbf{x}_{1,n-1}$, and $\mathbf{x}_{m-1,n-1}$ are directly adjacent to *two* boundaries, leading to contradictory definitions for their placement. Second, the direct specification of the first layer of interior boundary points while elliptically relaxing the remaining interior mesh points usually results in a distinct discontinuity in mesh properties near the grid boundaries (i.e., a very "nonsmooth" transition between the algebraic boundaries and the elliptically smooth interior). Typically, one transitions from an orthogonal mesh at the boundary to a slightly skewed mesh, within the space of a grid line. Nevertheless, these difficulties are generally easily mitigated with simple modifications.

An alternative approach for obtaining grid spacing information from the algebraic initial condition is depicted in Figure 3.2. Here, the orthogonally placed interior point is reflected an equal distance opposite the boundary curve to form a "ghost point." When repeated along the boundary, this procedure forms an "exterior curve" of ghost points adjacent to the boundary creating an image of the first algebraic grid line within the domain. The ghost points are computed prior to mesh relaxation and are not refreshed during the relaxation process. This boundary triad of points (the "ghost," boundary, and interior point) are employed in the calculation of the usual

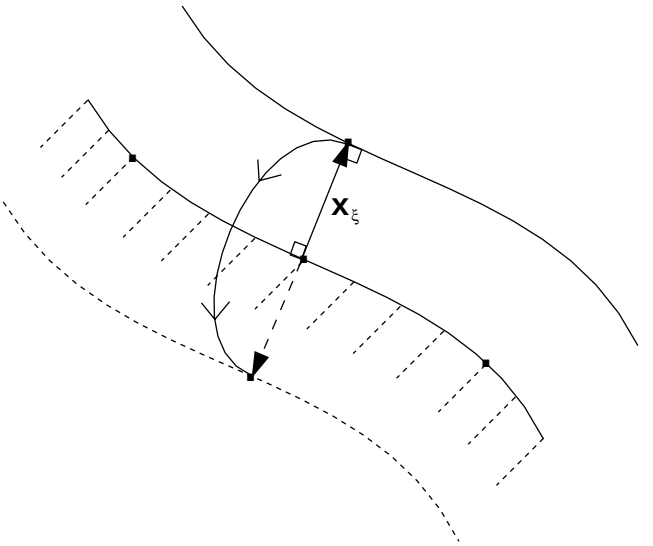


FIG. 3.2. *Reflection of orthogonalized interior grid point to form an external ghost point.*

second derivative ($\mathbf{x}_{\xi\xi}$) at the boundary and the normal spacing ($\sqrt{g_{11}}$) off the boundary. Constraining the ghost point assures that the normal spacing is not lost during the course of iteration, as occasionally occurs in the sliding orthogonality approach. Conversely, all of the interior grid points are mobile throughout the relaxation process, so smoothness of the grid is not compromised.

To quantify this approach, at the $\xi = 0$ boundary, let $(\mathbf{x}_\eta)_{0,j}$ denote the centrally differenced derivative $\frac{1}{2}(\mathbf{x}_{0,j+1} - \mathbf{x}_{0,j-1})$. Let $(\mathbf{x}^o_\xi)_{0,j}$ denote the one-sided derivative $\mathbf{x}_{1,j} - \mathbf{x}_{0,j}$ that is evaluated on the initial algebraic grid. Furthermore, let $\mathbf{a}$ be the unit vector normal to the boundary, which is defined with the use of the orthogonality condition (2.6):

$$\mathbf{a} \equiv \frac{\mathbf{x}_\xi}{||\mathbf{x}_\xi||} = \frac{(y_\eta, -x_\eta)}{\sqrt{x_\eta^2 + y_\eta^2}} = \frac{(y_\eta, -x_\eta)}{\sqrt{g_{22}}}.$$

Quantitatively, $\mathbf{x}_\xi$ in Figure 3.2 is given by

$$\mathbf{x}_\xi = \mathbf{P}_\mathbf{a}(\mathbf{x}^o_\xi), \tag{3.2}$$

where $\mathbf{P}_\mathbf{a} = \mathbf{a}\mathbf{a}^\mathbf{T}$ is the orthogonal projection onto the one-dimensional subspace spanned by the unit vector $\mathbf{a}$. Thus we obtain

$$\mathbf{x}_\xi = \mathbf{a}(\mathbf{a} \cdot \mathbf{x}^o_\xi) = \frac{(y_\eta, -x_\eta)}{g_{22}}(y_\eta x^o_\xi - x_\eta y^o_\xi). \tag{3.3}$$

The reflection operation depicted in the diagram implies that the constrained ghost point location is

$$\mathbf{x}_{-1,j} = \mathbf{x}_{0,j} - (\mathbf{x}_\xi)_{0,j}.$$

Alternatively, this can be viewed as a first-order Taylor expansion involving the orthogonal derivative $(\mathbf{x}_\xi)_{0,j}$:

$$\mathbf{x}_{-1,j} = \mathbf{x}_{0,j} + \Delta\xi(\mathbf{x}_\xi)_{0,j},$$

with $\Delta\xi = -1$. The orthogonal derivative $(\mathbf{x}_\xi)_{0,j}$ is computed in (3.3) using data from the boundary and the algebraic grid. To evaluate the control functions (2.5) at the boundary, the second derivative $\mathbf{x}_{\xi\xi}$ is computed using a central difference approximation involving the ghost point, boundary point, and the iteratively updated interior point. The metric coefficient g_{11} describing the spacing normal to the boundary is computed using (3.3),

$$(g_{11})_{0,j} = (\mathbf{x}_\xi)_{0,j} \cdot (\mathbf{x}_\xi)_{0,j}.$$

Finally, note that the value for $(\mathbf{x}_\xi)_{0,j}$ used in (2.5) is not the fixed value given by (3.3), but the iteratively updated one-sided difference formula given by

$$(\mathbf{x}_\xi)_{0,j} = \mathbf{x}_{1,j} - \mathbf{x}_{0,j}.$$

Evaluation of the boundary conditions for the $\xi = m$ boundary is a similar process, with the ghost point locations given by

$$\mathbf{x}_{m+1,j} = \mathbf{x}_{m,j} + (\mathbf{x}_\xi)_{m,j}.$$

$(\mathbf{x}_\xi)_{m,j}$ is evaluated using (3.3), which is also valid for this boundary.

At the $\eta = 0$ and $\eta = n$ boundaries, the derivatives $\mathbf{x}_\eta$ and $\mathbf{x}_{\eta\eta}$, along with the spacing g_{11}, are evaluated using the fixed boundary data using central differences. For the $\eta = 0$ boundary, the ghost point location is

$$\mathbf{x}_{i,-1} = \mathbf{x}_{i,0} - (\mathbf{x}_\eta)_{i,0},$$

where

$$\mathbf{x}_\eta = \frac{(-y_\xi, x_\xi)}{g_{11}}(-y_\xi x^o_\eta + x_\xi y^o_\eta). \tag{3.4}$$

In this case, $(-y_\eta, x_\eta)$ and g_{11} are evaluated using central differencing of the boundary data, and (x^o_η, y^o_η) represents a one-sided derivative $\mathbf{x}_{i,1} - \mathbf{x}_{i,0}$, again evaluated on the initial algebraic grid. The metric coefficient $(g_{22})_{i,0} = (\mathbf{x}_\eta)_{i,0} \cdot (\mathbf{x}_\eta)_{i,0}$ is computed using (3.4), and $\mathbf{x}_{\eta\eta}$ is computed using the ghost point, boundary point, and iteratively updated interior point. Similarly, the value of $(\mathbf{x}_\eta)_{i,0}$ used in (2.5) is not the fixed value given in (3.4) but is iteratively obtained from

$$(\mathbf{x}_\eta)_{i,0} = \mathbf{x}_{i,1} - \mathbf{x}_{i,0}.$$

Finally, the $\eta = n$ boundary is similar, with the ghost point locations given by

$$\mathbf{x}_{i,n+1} = \mathbf{x}_{i,n} + (\mathbf{x}_\eta)_{i,n},$$

and $(\mathbf{x}_\eta)_{i,n}$ is evaluated using (3.4).

Quantities for the four corner points, $\mathbf{x}_{0,0}$, $\mathbf{x}_{m,0}$, $\mathbf{x}_{0,n}$, and $\mathbf{x}_{m,n}$ are computed in a different manner. The values for $\mathbf{x}_\xi$, $\mathbf{x}_{\xi\xi}$, $\mathbf{x}_\eta$, $\mathbf{x}_{\eta\eta}$, g_{11}, and g_{22} are evaluated using one-sided difference formulas employing the requisite boundary values and are not refreshed during the course of iteration. Near the corners of the domain, *conformality* is usually more important than *orthogonality*. Precisely, orthogonality at the corners should be sacrificed in order to insure that the resulting grid does not "spill" over the physical boundaries. For the case of highly obtuse or highly acute corners, it is often necessary to relax orthogonality at boundary points extending some distance from the

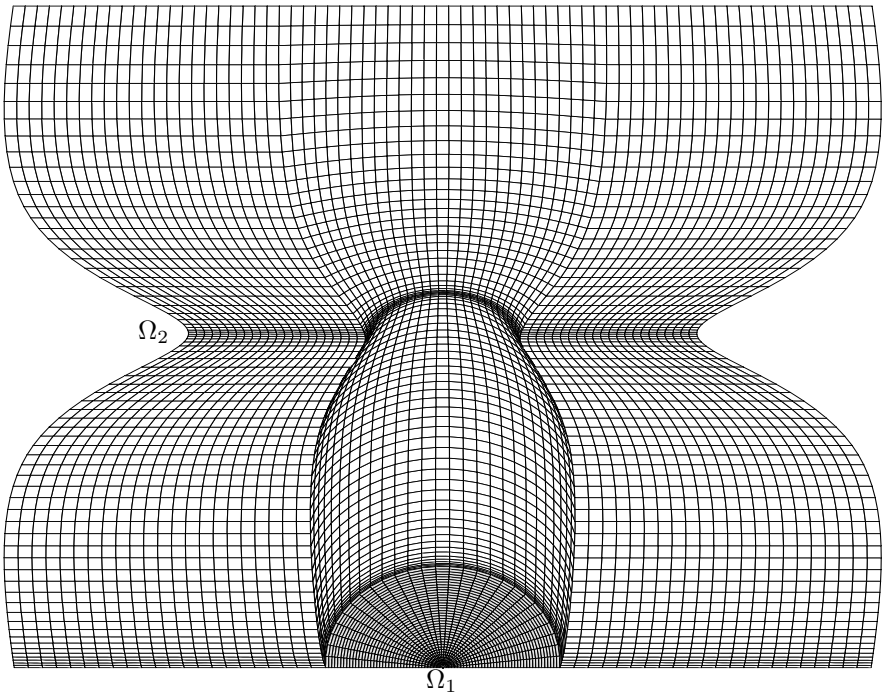


FIG. 3.3. *An algebraic planar grid on a multimaterial domain consisting of regions* Ω_1 *and* Ω_2.

true corner. To implement this case, the relevant ghost points are defined by a smooth blend beginning with the omission of the orthogonal projection operation (3.2) (i.e., definition by simple extrapolation) at the corners. This placement is modified through a transition to a weighted combination of extrapolation and orthogonal placement, then finally to the described orthogonality method some distance from the corner.

To demonstrate the effectiveness of the global technique, consider the initial algebraic planar grid spanning the bicubic geometry detailed in Figure 3.3. This mesh is created on a two material domain, with the material discontinuity located at the transition between the "pie" grid emanating from the singularity at the center of the bottom boundary (Ω_1) and the background mesh (Ω_2). The global mesh is highly skewed at certain points along the boundaries and possesses an undesirable concentration of points in the interior of the grid. Additionally, close inspection reveals folding of the algebraic grid in the central region.

Figure 3.4 shows the effectiveness of the grid generation system. The grid is smooth and boundary orthogonal, and folding at the interior has been removed. Close inspection reveals the use of the described orthogonality technique at the boundaries, as the grid point distribution (both along the boundary and normal to the boundary) remains unchanged. The concept of *preservation* of the initial grid information at the boundaries is essential to the derivation of the boundary impedance mechanism, which follows immediately.

4. Grid impedance matching. As outlined in the introduction, the impedance matching principle is a quantitative method aimed at reducing the errors of a simulation on a particular grid spanning the domain Ω. If possible, one desires a grid that preserves the physical discontinuities present in the domain without the introduction

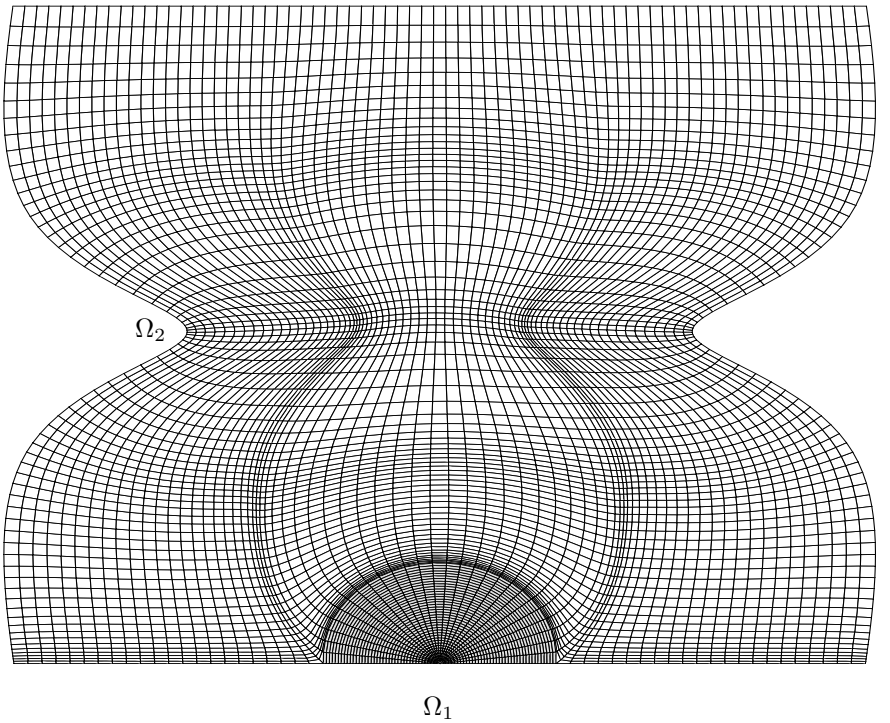


FIG. 3.4. *An elliptic planar grid on a multimaterial domain consisting of regions* Ω_1 *and* Ω_2.

of solution artifacts and one that also accommodates the assumptions of accuracy of the discrete system to be solved on the grid. These requirements are further complicated in cases of multimaterial simulations, where the domain contains dissimilar materials separated by geometric interfaces, $\Omega = \cup_{i=1}^{N} \Omega_i$. This work seeks to present impedance matching in the general, multimaterial case.

Given a mesh spanning a domain consisting of more than one material, three considerations describe the impedance matching principle. First, the mesh must be constructed such that, for an nth-order discretization, $n > 1$, the actual solution "displays" nth-order accuracy on the mesh. Second, one must insure solution accuracy in the neighborhood of internal material interfaces. Given the desire that the solution display an expected level of accuracy at the boundaries of regions leads to a necessary requirement on the mesh in the neighborhood of material interfaces. Last, this mesh must also be constructed such that the magnitude of the "ignored" higher-order terms (truncation error terms) in the discretization of the governing equations solved on the mesh are "small." The following sections discuss each of these topics and lead to the development of constraint expressions that govern the resulting impedance-matched grid generation process.

4.1. Effects of mesh spacing on truncation error. Nonuniform meshes are commonplace in numerical simulation, both due to the desire to generate boundary-fitted grids on domains defined by complex geometric boundaries and due to the desire to economize solution time by adaptively concentrating grid refinement in areas where small length scales are present in the solution. However, Hirt and Ramshaw [11] show that, for the general case, finite-difference approximations on a nonuniform mesh

may result in lower-order formal accuracy than finite-difference approximations on a uniform mesh.

To explore this concept, a truncation error analysis suggested by Hoffman [12] is employed. Let $\phi \equiv \phi(x)$ define the conservation quantity of interest on the domain Ω. Furthermore, consider the case that the discrete governing equations contain terms that require the evaluation of the first spatial derivative of ϕ that may be expressed as the central difference

$$\phi_x = \frac{\phi_{i+1} - \phi_{i-1}}{x_{i+1} - x_{i-1}}. \tag{4.1}$$

To study the truncation error of this approximation, a Taylor series expansion of ϕ about the arbitrary point x_i is developed,

$$\phi_{i\pm1} = \phi\big|_i \pm \phi_x\big|_i \Delta x_\pm + \frac{1}{2}\phi_{xx}\big|_i \Delta x_\pm^2 + O(\Delta x_\pm^3),$$

where $\Delta x_\pm$ is a forward/backward spatial difference about i. The numerator of (4.1) can be written in terms of the expansion as

$$\phi_{i+1} - \phi_{i-1} = \phi_x\big|_i (\Delta x_+ + \Delta x_-) + \frac{1}{2}\phi_{xx}\big|_i (\Delta x_+^2 - \Delta x_-^2) + O(\Delta x_\pm^3).$$

The denominator of (4.1) may be written in terms of Δx on the mesh:

$$x_{i+1} - x_{i-1} = (x_{i+1} - x_i) + (x_i - x_{i-1}) = (\Delta x_+ + \Delta x_-).$$

Finally, substitution into (4.1) reveals the expression for the first derivative of ϕ:

$$\phi_x = \phi_x\big|_i + \frac{1}{2}\phi_{xx}\big|_i (\Delta x_+ - \Delta x_-) + O(\Delta x_\pm^2). \tag{4.2}$$

Clearly, this statement suggests that the discretization is second-order accurate only when $\Delta x_+ = \Delta x_-$. This result appears to paint a bleak picture of the prospect of accurate solutions on nonuniform meshes spanning complex geometries. Fortunately, it is possible to address this apparent limitation effectively with the use of logical mapping and the interface impedance condition.

4.2. Impedance matching at grid interfaces. Material interfaces within the computational domain may be treated as region boundaries between meshes within the multimaterial domain. Given a point distribution on the boundaries and internal interfaces, it is possible to use an algebraic technique to construct a mesh within each subregion of the domain such that the algebraic mesh conforms to the interface geometry. The first step toward the generation of an impedance-matched mesh amenable to simulation is the examination of the algebraic mesh near these internal interfaces. It is first necessary to insure that the algebraic mesh near these interfaces provides second-order accuracy in the discretization of ϕ.

Consider a second-order discretization across the interface discontinuity, where

$$\lim_{\Delta x_+ \to 0} \phi^+ \neq \lim_{\Delta x_- \to 0} \phi^-.$$

In this expression, assume that the mesh point i lies on the interface boundary, $i+1$ is the point immediately to the"right" or + side of the interface, and $i-1$ is the point

immediately to the "left" or $-$ side of the interface. Furthermore, a first derivative expression of the form

$$\phi_r = \frac{(\phi_{i+1} - \phi_i) + (\phi_i - \phi_{i-1})}{r_{i+1} - r_{i-1}}$$

is desired across the interface discontinuity, where r is the coordinate normal to the interface curve. Again, it is possible to expand the components of the numerator on each side of the interface using a Taylor series, resulting in the expressions

$$\phi_{i+1} - \phi_i = \phi_r^+\big|_i \Delta r_+ + \frac{1}{2}\phi_{rr}^+\big|_i \Delta r_+^2 + O(\Delta r_+^3)$$

and

$$\phi_i - \phi_{i-1} = -\phi_r^-\big|_i \Delta r_- - \frac{1}{2}\phi_{rr}^-\big|_i \Delta r_-^2 + O(\Delta r_-^3).$$

This result may be combined to provide

$$(\phi_{i+1}-\phi_i)+(\phi_i-\phi_{i-1}) = \left(\phi_r^+\big|_i \Delta r_+ - \phi_r^-\big|_i \Delta r_-\right)+\frac{1}{2}\left[\phi_{rr}^+\big|_i \Delta r_+^2 - \phi_{rr}^-\big|_i \Delta r_-^2\right]+O(\Delta r_\pm^3).$$

The desired derivative of ϕ then takes the form

$$\phi_r = \frac{\phi_r^+\big|_i \Delta r_+ - \phi_r^-\big|_i \Delta r_-}{\Delta r_+ + \Delta r_-} + \frac{1}{2}\frac{\phi_{rr}^+\big|_i \Delta r_+^2 - \phi_{rr}^-\big|_i \Delta r_-^2}{\Delta r_+ + \Delta r_-} + O(\Delta r_\pm^2). \tag{4.3}$$

At this point, one desires the grid at the interface have the property that the first-order term in (4.3) is "small." If the numerator of this term is unconditionally zero,

$$\phi_{rr}^+\big|_i \Delta r_+^2 - \phi_{rr}^-\big|_i \Delta r_-^2 = 0,$$

the magnitude of the first-order term is zero. Rearranging this result provides the impedance condition on the grid at the interface:

$$\sqrt{\phi_{rr}^+\big|_i}\,\Delta r_+ = \sqrt{\phi_{rr}^-\big|_i}\,\Delta r_-.$$

Let $\rho(r) \equiv \phi_{rr}$ define the impedance quantity of interest, in the direction r normal to the interface. The condition that provides impedance matching across the interface reduces to

$$\sqrt{\rho^+\big|_i}\,\Delta r_+ = \sqrt{\rho^-\big|_i}\,\Delta r_-. \tag{4.4}$$

The initial algebraic mesh on the domain is effectively arbitrary; it serves only as an initial "guess" for the quasi-orthogonal grid generation system and allows the evaluation of the boundary conditions for the elliptic method. Recall, however, that the evaluation of these boundary conditions employs the first algebraically determined grid line internal to the boundary. Clearly, the position of this grid line can be evaluated using the impedance match condition (4.4), in lieu of using data from the algebraic grid, with no loss of generality. Via this mechanism, the interface impedance condition is coupled to the quasi-orthogonal grid generation system.

4.3. Impedance matching in the grid interior. A similar approach may be used within the grid interior, away from discontinuities. As seen in (4.2), this yields the requirement of a uniform mesh. This is certainly one possible impedance-matched mesh. However, using a functional mapping from a computational space in which the discrete nodal points are uniformly distributed to a physical space with nonuniformly distributed nodes also results in a mesh that displays second-order accuracy.

Let $x = x(\xi)$ and $\phi = \phi(x(\xi)) = \phi(\xi)$. The desired first derivative statement becomes

$$\phi_x = \phi_\xi \, \xi_x = \xi_x\big|_i \frac{\phi_{i+1} - \phi_{i-1}}{\xi_{i+1} - \xi_{i-1}}.$$

The numerator is again evaluated using a Taylor expansion:

$$\phi_{i\pm1} = \phi\big|_i \pm \phi_\xi\big|_i \Delta\xi + \frac{1}{2}\phi_{\xi\xi}\big|_i \Delta\xi^2 + O(\Delta\xi^3)$$

or

$$\phi_{i+1} - \phi_{i-1} = 2\phi_\xi\big|_i \Delta\xi + O(\Delta\xi^3).$$

For the uniform computational ξ mesh,

$$\xi_{i+1} - \xi_{i-1} = (\xi_i + \Delta\xi) - (\xi_i - \Delta\xi) = 2\Delta\xi.$$

Combining the above leads to

$$\phi_x = \xi_x\big|_i \, \phi_\xi\big|_i + O(\Delta\xi^2)$$

or

$$\phi_x = \frac{\phi_\xi\big|_i}{x_\xi} + O(\Delta\xi^2). \tag{4.5}$$

This result verifies that, given the mapping $x(\xi)$ and $\phi = \phi(\xi)$, a second-order solution in ξ results. Clearly, the quasi-orthogonal grid generation system (equations (2.4) and (2.5)) provides this mapping; the system was derived based on this transformation.

The effectiveness of this result is only damped by the apparent requirement that the simulation code must now operate in the computational space used to generate the grid, significantly increasing its complexity and run-time due to the required transformations. Hoffman [12] concludes, however, that solving in computational space is not a requirement, as merely the *existence* of the transformation is sufficient to provide second-order behavior on a nonuniform physical mesh. If the values of $\Delta x_\pm$,

$$\Delta x_+ - \Delta x_- = x_{\xi\xi} \, \Delta\xi^2 + O(\Delta\xi^4),$$

are chosen such that $\Delta x_+ - \Delta x_-$ quarters when $\Delta\xi$ is halved, the truncation error ϕ_x quarters (see (4.2)). This behavior of the transformation is independent of the fact that, considering the discretized physical space equations alone, the solution is formally only first-order accurate. This result allows decoupling of the transformation process employed in the grid generator from the solution process.

5. Implementation of impedance matching. Several important details remain in order to fully implement the second-order quasi-orthogonal impedance system. Section 3 developed an implementation of the quasi-conformal grid generation system and posed a set of boundary conditions necessary to enforce grid orthogonality at region interfaces (and the domain boundary). To summarize this result, to solve the quasi-elliptic grid generation system requires

1. a discretization of the boundary or interface curves,
2. use of the impedance condition to locate the first off-boundary row of grid points with the calculated normal spacing to the boundary, and
3. an initial algebraic grid to serve as an initial "guess" for the smoother on the interior of the region.

It was earlier proposed that the initial algebraic grid could be used to supply the boundary discretization. However, it is more useful to explore the use of the impedance condition, in an iterative setting, to place the points on the boundary to satisfy local impedance constraints. Using this concept, the impedance condition specifies grid point location both "along" (tangentially to) the boundary and "across" (normal to) the boundary, at least for the nearest row of grid points to the boundary. The quasi-elliptic system will then decay this off-boundary spacing to the geometric constraints and level of grid refinement in the interior of the region.

To implement the impedance-based grid generation system requires a definition of the geometrical basis of the domain. A general approach uses the assumption that domain boundaries and interfaces are defined by *parametric curves* (e.g., B-spline curves). As such, the curve geometry definition is in the form of a mapping $(x(u), y(u), z(u))$ from a *parametric u-domain* to a *physical* (x, y, z)*-domain*. This mapping is assumed differentiable and is denoted by $\mathbf{x}(u)$, where $\mathbf{x} = (x, y, z)$.

Discretization of these curves entails the placement of node points upon the curve; the actual distribution process is described by the generation of a mapping from the discrete *computational* ξ*-domain* to the *parametric u-domain*, resulting in the composite map $\mathbf{x}(\xi) = (x(\xi), y(\xi), z(\xi))$.

Physical space is a subset of $\mathbb{R}^3$, and the parametric space is a subset of $\mathbb{R}$, specifically the $[0, 1]$ unit interval. Technically, the computational space is a discrete linear set of points $\xi \in \{0, 1, \ldots, m\}$. However, to apply the concept of differentiable mappings between spaces, computational space is extended to become a continuum and is represented as the line $[0, m]$.

Let $\rho \equiv \rho(s)$ represent the impedance function of arclength s on the boundary curve in $\mathbb{R}^3$. This function is defined as the analogue of the one-dimensional case (4.4), or $\rho^{\pm}(s)\big|_i = \phi^{\pm}_{ss}\big|_i$. Similarly, the impedance condition at the boundary curves reduces to

$$\sqrt{\rho^+} s^+_\xi = \sqrt{\rho^-} s^-_\xi .$$

As an overview of the method employed, the impedance condition above is used to position the first free point at each end of the boundary curve. The remaining points on the boundary are subsequently interpolated given these end points to provide an initial guess for an elliptic relaxation procedure defined in terms of arclength s of the curve. This relaxation procedure preserves the required accuracy on the curve as it represents a variant of (4.5). Given the discrete boundaries, the initial mesh for the region is calculated using transfinite interpolation. The first row of off-boundary grid points is then adjusted using the orthogonality condition to define the local boundary normal r, with the impedance condition (4.4) used to calculate the local spacing of the

off-boundary point from the boundary. Finally, application of the quasi-orthogonal grid generation system results in an impedance-matched mesh in the interior of the region of interest.

To develop the impedance-matched point spacing on the boundary curve, consider the elliptic differential equation

$$\xi_{ss} = \Phi(s), \tag{5.1}$$

with boundary conditions

$$\frac{1}{\sqrt{\rho^+}}\xi_s^+ = \frac{1}{\sqrt{\rho^-}}\xi_s^- \quad \text{for} \quad s = s_0, s_m.$$

The impedance spacing function Φ is a function of arclength s of the boundary curve via the governing logical transformation. The change in arclength of the parametric curve is given by $ds = du||\mathbf{x}_u||$. Therefore, $\xi_s = \frac{1}{s_\xi}$, where

$$s_\xi = s_u u_\xi = ||\mathbf{x}_u|| u_\xi.$$

Additionally,

$$\xi_{ss} = -\frac{s_{\xi\xi}}{s_\xi^3},$$

where

$$s_{\xi\xi} = \frac{\mathbf{x}_u \cdot \mathbf{x}_{uu} u_\xi^2 + \mathbf{x}_u \cdot \mathbf{x}_u u_{\xi\xi}}{||\mathbf{x}_u||}.$$

The inverse transformation of (5.1) results in

$$u_{\xi\xi} + \Phi \mathrm{g} u_\xi = -u_\xi^2 \frac{\mathbf{x}_u \cdot \mathbf{x}_{uu}}{\mathbf{x}_u \cdot \mathbf{x}_u}, \tag{5.2}$$

where

$$\mathrm{g} = \mathbf{x}_\xi \cdot \mathbf{x}_\xi = \mathbf{x}_u \cdot \mathbf{x}_u u_\xi^2.$$

The boundary conditions are represented by

$$\begin{aligned} u_\xi &= \sqrt{\frac{\rho^-}{\rho^+}}\, \frac{||\Delta \mathbf{x}^-||}{||\mathbf{x}_u^+||} \quad \text{for} \quad \xi = 0, \\ u_\xi &= \sqrt{\frac{\rho^+}{\rho^-}}\, \frac{||\Delta \mathbf{x}^+||}{||\mathbf{x}_u^-||} \quad \text{for} \quad \xi = m. \end{aligned} \tag{5.3}$$

Equations (5.2) and (5.3) are applied to the boundary (or interface) nodes to iteratively obtain $\mathbf{x}_i = (x_i, y_i, z_i)$.

To implement the above on a given parametric boundary curve, u_0 and u_m are obtained from the algebraic curve grid. The ghost points u_{-1} and u_{m+1} are provided from opposite curves. Placement of the first two points interior of the curve end points is governed by the boundary conditions

$$\begin{aligned} u_1 &= u_0 + \sqrt{\frac{\rho_{-1}}{\rho_1}}\, \frac{||\mathbf{x}_0 - \mathbf{x}_{-1}||}{||\mathbf{x}_{u_0}||}, \\ u_{m-1} &= u_m - \sqrt{\frac{\rho_{m+1}}{\rho_m}}\, \frac{||\mathbf{x}_{m+1} - \mathbf{x}_m||}{||\mathbf{x}_{u_m}||}. \end{aligned}$$

The user-specified point density on the remainder of the curve is provided by the parametric values u_i for $i = 2, \ldots, m-2$. Linear interpolation,

$$u_i = t_i u_{m-1} + (1 - t_i)u_1, \quad t_i = \frac{i-1}{m-1} \quad \text{for} \quad i = 2, \ldots, m-2,$$

is an effective mechanism that may be used to provide the initial locations of these points. Impedance on these interpolated points is imposed by adjusting the impedance spacing function Φ near the boundary while holding the coordinates of the boundary points constant. To accomplish this, the impedance spacing function is evaluated at $i = 1$ and $i = m-1$ using

$$\Phi_i = -\frac{u_{\xi\xi}}{\mathrm{g} u_\xi}\bigg|_i - \frac{u_\xi}{\mathrm{g}} \frac{\mathbf{x}_u \cdot \mathbf{x}_{uu}}{\mathbf{x}_u \cdot \mathbf{x}_u}\bigg|_i \quad \text{for} \quad i = 1 \text{ and } m-1.$$

Φ is distributed to the remaining points on the curve, again using linear interpolation,

$$\Phi_i = t_i \Phi_{m-1} + (1 - t_i)\Phi_1, \quad t_i = \frac{i-1}{m-1} \quad \text{for} \quad i = 2, \ldots, m-2.$$

Let $(u_\xi)_{i,j}$ denote the central difference $\frac{1}{2}(u_{i+1} - u_{i-1})$, and $(u_{\xi\xi})_{i,j} \approx u_{i+1} - 2u_i + u_{i-1}$. In the relaxation process, the old parameter value u_0 is used in the discrete version of (5.2) to obtain u_i. This result allows the refresh of $\mathbf{x}_i = \mathbf{x}(u_i)$ along the curve. Let u_i denote $u(\xi)$ evaluated at the $\xi = i$ grid point. A central difference approach is used to approximate the solution of (5.2), to provide the boundary impedance condition

$$\begin{aligned} u_i =& \frac{1}{2}(u_{i+1} + u_{i-1}) \\ &+ \frac{1}{4}(u_{i+1} - u_{i-1})\big(\Phi_i \mathrm{g}_i\big) \\ &+ \frac{1}{8}(u_{i+1} - u_{i-1})^2 \left(\frac{\mathbf{x}_u \cdot \mathbf{x}_{uu}}{\mathbf{x}_u \cdot \mathbf{x}_u}\right)_i \quad \text{for} \quad i = 2, \ldots, m-2. \end{aligned}$$

This impedance equation is evaluated for u_i each iterative cycle.

Clearly, for the method just described to be effective, the impedance spacing function Φ_i must be refreshed every iteration prior to the computation of u_i. However, the curve boundary points u_0, u_1, u_{m-1}, and u_m only need to be initialized at the beginning of the relaxation cycle for each curve. This relaxation procedure is applied to all interface curves, in turn, until convergence is achieved.

To complete the grid generation process, the interior regions are discretized based on the bounding curves. Transfinite interpolation is used to provide an initial mesh for the quasi-orthogonal grid generation system detailed in section 3.

The boundary conditions for the system are specified by the direct application of the orthogonality condition (3.3) or (3.4), which places the off-boundary and ghost point on the vector r_i normal to the boundary at the interface point in question. The ratio of the distances of these two points from boundary point i along r is established by the direct application of the impedance condition (4.4). The magnitude of the separation distance between the off-boundary point is relaxed by the interior smoother while the ghost point separation distance is relaxed by the smoothing operation in the adjacent region. As such, the boundary condition treatment enforces orthogonality of these points with the boundary and a ratio spacing to satisfy the impedance condition at each iteration of the global smoother.

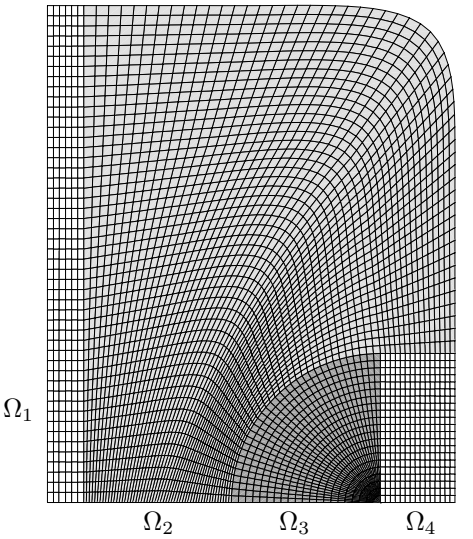


FIG. 5.1. *A multimaterial algebraic grid.*

Figure 5.1 depicts a four-material region with an algebraic grid spanning $\Omega = \cup_{i=1}^{4}\Omega_i$. Ω_1 is the region along the left side of the domain, Ω_3 is the "pie-slice" region emanating from the singularity toward the lower-right side of the figure, Ω_4 is the rectangle at the lower-right side of the figure, and Ω_2 is the remainder of the mesh. This particular figure was chosen to provide a simple example that would show the outcome of the impedance matching principle while illustrating how the coupling between impedance and the smoothing algorithm results in a "globally smooth" mesh. This example contains several interfaces separating materials of different density, with one interface simply spanning the domain and with the remaining interfaces being self intersecting in the area of the "pie slice." This topology, along with the selection of the domain geometry, results in a nontrivial test case sufficiently complex to motivate the effectiveness of the method. With a density difference in the regions of the mesh, one would expect to see a nonuniform spacing at the interface that satisfies the second-order impedance condition $\sqrt{\rho^+}s_\xi^+ = \sqrt{\rho^-}s_\xi^-$, with a smooth decay from this condition as one moves away from the interface.

Figure 5.2 details the result of the impedance-coupled quasi-orthogonal grid generation applied to the algebraic mesh, where the impedance quantities were specified as constants in each of the regions, $\rho_1 = 3.0$, $\rho_2 = 30.0$, $\rho_3 = 2.0$, and $\rho_4 = 1.0$. In this result, the effect of the interface impedance condition is clearly evident. For example, consider the interface between regions Ω_1 and Ω_2, where the internal boundary is vertical. The impedance condition may be expressed as $\sqrt{\rho_{\Omega_1}}s_\xi^{\Omega_1} = \sqrt{\rho_{\Omega_2}}s_\xi^{\Omega_2}$, which specifies an off-interface discontinuous spacing as a function of the ρ-discontinuity at the interface.

Let the ξ coordinate describe a horizontal grid line at this interface, and $\rho_1 = 3.0$ with $\rho_2 = 30.0$. Given a distance δs^+ (the distance from the interface to the first vertical grid line left of it) equal to some constant C, the impedance condition mandates that

$$\delta s^- = \sqrt{\frac{\rho_+}{\rho_-}}s_\xi^+ = \sqrt{\frac{\rho_+}{\rho_-}}C,$$

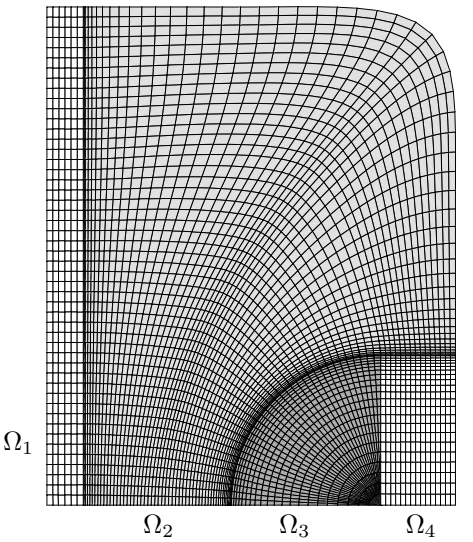


FIG. 5.2. *A quasi-orthogonal, impedance-matched multimaterial grid.*

or $\delta s^- = C/\sqrt{10}$. For this example, satisfying the impedance condition is equivalent to a local grid spacing roughly three times finer to the right of the interface than to the left. Clearly, the impedance condition has the effect of "packing" grid lines toward the lower-ρ region at material discontinuities. Additionally, the overall smoothness and orthogonality provided by the base system is apparent in the example.

The impedance principle can clearly be extended to higher-order terms in the approximation. Several authors have explored the reduction in magnitude of higher-order terms in a second-order approximation. Mastin [13] suggests sufficient conditions that arise from an examination of truncation error. This work provides intuitive bounds on the grid's departure from orthogonality, along with the rate of change of grid spacing, that are derived from a truncation analysis. Further work remains to couple the reduction of higher-order terms to a grid generation system. The results obtained here are qualitatively "smooth" and "orthogonal." However, the desire remains to guarantee that the constraints developed by Mastin et al. will be unconditionally satisfied within the grid generation system.

6. Summary. In this development of a system for the generation of quasi-orthogonal, impedance-matched meshes, an elliptic system was derived and shown to be effective in the generation of quasi-conformal mappings. Boundary conditions were imposed on general parametrically defined boundary curves. These conditions were a function of both geometric and impedance constraints.

The grid generation process begins with the user specifying the number of points desired on each boundary. The first and last points are placed using the impedance condition at the end points of each curve, with the remainder placed based on a transformation using the impedance condition as a boundary condition. The "line" of ghost points and the first line of interior grid points surrounding each region of the grid are placed using the impedance condition to specify the ratio spacing normal to the boundary, with the orthogonality condition used to specify point placement tangentially to the boundary. Transfinite interpolation is used to provide an algebraic spacing interior to each region for the purpose of providing an initial condition for the quasi-orthogonal method.

Given the algebraic mesh with impedance-matched boundaries, the quasi-orthogonal

grid generation system is used to relax the mesh to satisfy the quasi-orthogonal transformation in the interior region. As this relaxation proceeds toward convergence, an impedance-matched mesh with second-order error behavior results across the computational domain.

This approach to grid generation adds generality to grid generation methods based on elliptic systems and explores the mechanics of coupling quantitative quality measures to the grid generation process. Future challenges would include the extension of these techniques to more complex mesh topologies and a detailed examination of the effect and incorporation of higher-order terms into the grid generation system to further refine the simulation qualities of the resulting mesh.

REFERENCES

[1] L. V. Ahlfors, *Lectures on Quasiconformal Mappings*, Van Nostrand, New York, 1966.

[2] S. L. Krushkal, *Quasiconformal Mappings and Riemann Surfaces*, Winston & Sons, Washington, DC, 1979.

[3] H. Renelt, *Elliptic Systems and Quasiconformal Mappings*, Wiley, New York, 1988.

[4] L. Bers, *Mathematical Aspects of Subsonic and Transonic Gas Dynamics*, Wiley, New York, 1958.

[5] A. Khamayseh and C. W. Mastin, *Computational conformal mapping for surface grid generation*, J. Comput. Phys., 123 (1996), pp. 394–401.

[6] C. W. Mastin and J. F. Thompson, *Quasiconformal mappings and grid generation*, SIAM J. Sci. Statist. Comput., 5 (1984), pp. 305–310.

[7] S. K. Godunov and P. G. Prokopov, *On the computation of conformal transformations and the construction of difference meshes*, USSR Comput. Math. Math. Phys., 7 (1967), pp. 89–124.

[8] R. Arina, *Adaptive orthogonal curvilinear coordinates*, in Numerical Methods for Fluid Dynamics III, K. W. Morton and M. J. Baines, eds., Clarendon Press, Oxford, UK, 1988, pp. 353–359.

[9] J. F. Thompson, F. C. Thames, and C. W. Mastin, *Automatic numerical generation of body-fitted curvilinear coordinate system for fields containing any number of arbitrary two dimensional bodies*, J. Comput. Phys., 15 (1974), pp. 299–319.

[10] R. L. Sorenson, *A Computer Program to Generate Two-Dimensional Grids about Airfoils and Other Shapes by the Use of Poisson's Equations*, NASA TM 81198, NASA Ames Research Center, Moffet Field, CA, 1980.

[11] C. W. Hirt and J. D. Ramshaw, *Prospects for numerical simulation of bluff-body aerodynamics*, in Aerodynamic Drag Mechanisms of Bluff Bodies and Road Vehicles, G. Sovran, T. Morel, and W. Mason, Jr., eds., Plenum, New York, 1978, pp. 313–355.

[12] J. D. Hoffman, *Relationship between the truncation errors of centered finite-difference approximations on uniform and nonuniform meshes*, J. Comput. Phys., 46 (1982), pp. 469–474.

[13] C. W. Mastin, *Truncation error on structured grids*, in Handbook of Grid Generation, J. F. Thompson, B. K. Soni, and N. P. Weatherill, eds., CRC Press, New York, 1998, pp. 32.1–32.10.

SIAM J. SCI. COMPUT.
Vol. 22, No. 4, pp. 1238–1268
© 2000 Society for Industrial and Applied Mathematics

MULTIRESOLUTION ANALYSIS AND SUPERCOMPACT MULTIWAVELETS*

RICHARD M. BEAM† AND ROBERT F. WARMING†

Abstract. The Haar wavelets can represent exactly any piecewise constant function. The motivation for the present development is Alpert's family of compact orthogonal multiwavelets that can represent exactly any piecewise polynomial function. We choose to derive the algorithm in the style and notation of Harten's multiresolution analysis as extended to multiwavelets by the authors. We begin with a description of the nested grid hierarchy. Next comes the decomposition, which is the heart of the algorithm, and finally the reconstruction. The basis functions (which are nonfractal) retain the spatial compactness of the Haar basis functions, which enhances the algorithm application to nonperiodic and piecewise continuous data.

Key words. wavelets, multiwavelets, multiresolution

AMS subject classifications. 41, 65

PII. S1064827596311906

1. Introduction.

Strang [6] has observed that the Haar wavelets have *almost* all the right stuff. They are compact, piecewise analytic (nonfractal), and orthogonal; however, they are lacking in approximation (only first-order or one vanishing moment). Daubechies [2] extended the order of approximation and retained the orthogonality, but the Daubechies wavelets are fractal and less compact than the Haar wavelets. Geronimo, Hardin, and Massopust [4] used multiwavelets to obtain wavelets which are second-order in approximation, symmetric, and orthogonal but have the same compactness as Daubechies and are not analytic. Strang and Strela [7] extended the multiwavelet work of Geronimo, Hardin, and Massopust [4] to retain the general order of approximation of Daubechies with more compactness. The Strang–Strela multiwavelets are more compact than the Daubechies wavelets but less compact than Haar wavelets, and they are fractal. Recently, Donovan, Geronimo, and Hardin [3] used multiresolution analyses for the construction of multiwavelets that have arbitrary regularity and orthogonality and are also symmetric and piecewise polynomial. Their wavelets have compact support but not the compactness of Haar. In this paper we present multiwavelets that retain the compactness of Haar wavelets, are piecewise polynomial and orthogonal, and can have arbitrary order of approximation.

In Harten's multiresolution analysis [5] one needs a scalar interpolation formula to transfer information between scales. The present authors extended Harten's multiresolution in [8] to include Hermite interpolation that leads to vector interpolation and multiwavelets. These wavelets are nonfractal and accurate; however, they are not orthogonal (but biorthogonal) and are less compact than Haar wavelets. Our search for the orthogonal basis with the compactness of Haar (which we call *supercompact*) led to the present paper. The supercompact multiwavelets of this paper are closely related to the multiwavelet basis of Alpert [1]. Alpert constructs the multiwavelet

*Received by the editors November 11, 1996; accepted for publication (in revised form) November 28, 1997; published electronically October 25, 2000. The main results of this paper were presented at the SIAM Annual Meeting, Kansas City, Missouri, July 22-26, 1996 and in NASA TM-110405, 1996.

http://www.siam.org/journals/sisc/22-4/31190.html

†Advanced Computational Methods Branch, NASA Ames Research Center, Moffett Field, CA 94035-1000.

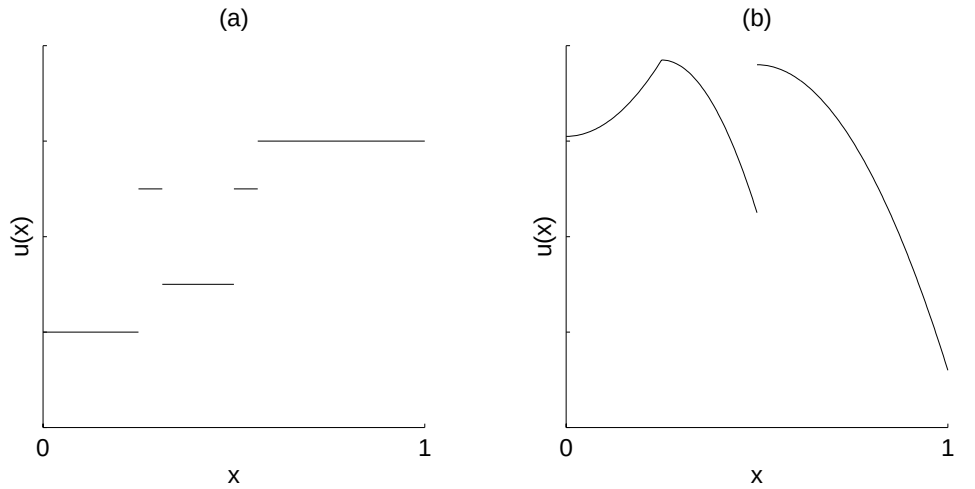


FIG. 1. *Piecewise continuous functions:* (a) *piecewise constant function,* (b) *piecewise polynomial function.*

basis (with vanishing moments) directly, whereas the multiwavelet basis in our development is a byproduct of the polynomial interpolation and the vanishing moments are assured by the order of the interpolation. In fact, neither the development nor the application of our multiresolution algorithm requires an explicit knowledge of the multiwavelet basis.

The advantages of the multiresolution algorithm with supercompact support of the basis are twofold. First, the application to functions defined on a finite interval does not require any special treatment at the boundaries of the interval. Second, the application to functions that are only piecewise continuous (internal boundaries) can be efficiently implemented.

The paper is divided as follows: In section 2 we present the multiresolution algorithm development that uses orthogonal polynomial interpolation. In section 3 we evaluate the coefficients of the multiresolution algorithm that are dependent on the choice of interpolation polynomials and give examples for Legendre polynomials. In section 4 we consider the implementation of the multiresolution algorithm for various types of initial data: analytical, discrete function values, discrete derivative values, and/or discrete integral (cell average) values. Section 5 is devoted to an exposition of the multiwavelet basis. In section 6 we give some application examples that highlight the virtues of the algorithm. We conclude the main paper with some brief concluding remarks in section 7. Details that detract from the continuity of the presentation are relegated to appendices.

2. Algorithm development. The Haar wavelets can represent exactly any piecewise constant function (Figure 1(a)). The motivation for the present development is a family of wavelets (multiwavelets) that can represent exactly any piecewise polynomial function (Figure 1(b)). We choose to derive the algorithm in the style and notation of Harten's multiresolution analysis as extended to multiwavelets by the authors in [8]. We begin with a description of the nested grid hierarchy. Next comes the interpolation, then the decomposition (which is the heart of the algorithm), and finally the reconstruction.

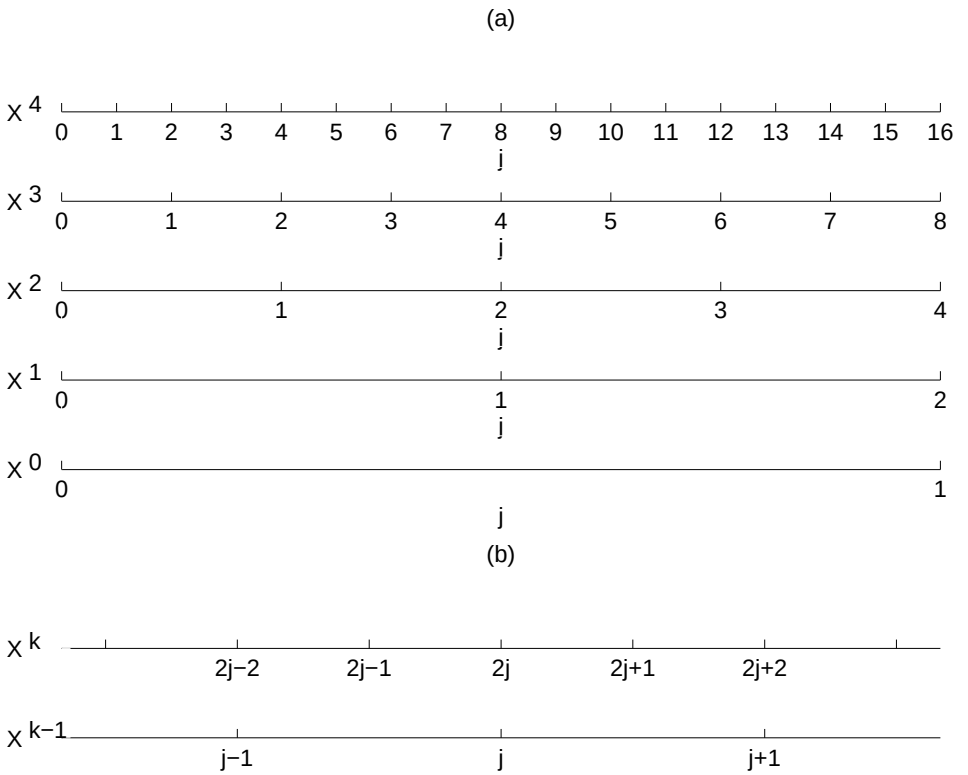


FIG. 2. *Nested grids:* (a) *nested sequence of grids,* (b) *two generic grid levels.*

2.1. Nested grid hierarchy. Let $m > n$ be positive integers and let $\{X^k\}_{k=m,m-1,\dots,n}$ denote a nested sequence of uniform grids on the unit interval $[0,1]$ defined by

$$X^k = \{x_j^k\}_{j=0}^{J_k}, \quad x_j^k = jh_k, \quad h_k = 2^{-k}, \quad J_k = 2^k, \tag{2.1}$$

where k denotes the grid level index (see, e.g., Figure 2(a)). The level index $k = m$ corresponds to the *finest* grid in the hierarchy where the number of subintervals is $J_m = 2^m$ and the subinterval (grid spacing) is $h_m = 2^{-m}$. The level index $k = n$ corresponds to the *coarsest* grid where the number of subintervals is $J_n = 2^n$ and the subinterval is $h_n = 2^{-n}$. Note that X^{k-1} is formed from X^k by removing grid points of X^k with odd indices:

$$x_j^{k-1} = x_{2j}^k, \quad j = 0, 1, 2, \dots, J_{k-1} \tag{2.2}$$

(see Figure 2(b)).

In the multiresolution algorithm development we assume that the discrete data on the finest grid comes from a piecewise continuous function $u(x)$ on the unit interval $[0,1]$. A preprocessing procedure uses a polynomial interpolation to transform $u(x)$ into a matrix $\mathbf{A}^m$ of discrete values on the finest level $k = m$ with grid X^m:

$$\mathbf{A}^m = [\boldsymbol{\alpha}_1^m, \boldsymbol{\alpha}_2^m, \dots, \boldsymbol{\alpha}_{J_m}^m], \quad J_m = 2^m. \tag{2.3}$$

For multiwavelets the vectors $\boldsymbol{\alpha}_j^m$ are column vectors whose length is determined by the order of polynomial approximation. The matrix $\mathbf{A}^m$ is the input to the multiresolution algorithm. For each grid X^k at each level k there is an associated matrix $\mathbf{A}^k$:

$$\mathbf{A}^k = [\boldsymbol{\alpha}_1^k, \boldsymbol{\alpha}_2^k, \ldots, \boldsymbol{\alpha}_{J_k}^k], \quad J_k = 2^k. \tag{2.4}$$

The matrix $\mathbf{A}^{k-1}$ has half the number of columns of $\mathbf{A}^k$ and is formed by a prescription given in the multiresolution algorithm developed below. Specific examples of the matrix $\mathbf{A}^m$ are given in section 4.

2.2. Interpolation. The next step of the multiresolution development is an algorithm to project the discrete data $\mathbf{A}^k$, matrix (2.4), from grid X^k to grid X^{k-1} (fine to coarse) and conversely from grid X^{k-1} to grid X^k (coarse to fine). The fine to coarse projection is a decomposition or decimation (less data) and the coarse to fine projection is a reconstruction or fill-in (more data). In this section we begin the development of the projection algorithms by considering the special case where $u(x)$ is a polynomial function of degree ℓ. On each interval of the coarse grid X^{k-1} we expand $u(x)$ in a family of orthogonal polynomials (degree ℓ) with coefficients $\mathbf{A}^{k-1}$, the discrete data (2.4). This choice of a compact (one grid interval) expansion with orthogonal polynomials of degree ℓ will produce supercompact orthogonal projections with order of accuracy $\ell + 1$, i.e., exact reconstruction (fill-in) for polynomials of degree ℓ. In the next section we consider the practical case where $u(x)$ is a general piecewise continuous function and the polynomial expansion is only an approximation. In section 4 we consider a specific example of a family of orthogonal polynomials, e.g., Legendre polynomials.

First we introduce some notation. Let $\phi_i(\xi)$ be a polynomial of degree ℓ or less with support on the interval $-1 < \xi \le 1$:

$$\phi_i(\xi) = \begin{cases} \sum_{p=0}^{\ell} a_{p,i}\xi^p, & -1 < \xi \le 1, \\ 0, & \text{otherwise.} \end{cases} \tag{2.5}$$

We will assume that the family of polynomials (2.5) $i = 0, 1, \ldots, \ell$ is orthonormal, i.e.,

$$\int_{-1}^{1} \phi_i(\xi)\phi_j(\xi)d\xi = \delta_{ij}. \tag{2.6}$$

We assume that ℓ is sufficiently small so the Runge phenomenon for polynomial interpolation is not an issue. The contracted form of $\phi_i(\xi)$ with support on the jth subinterval of the level k grid is

$$(\phi_i(x))_j^k = \begin{cases} \phi_i\left(1 + \frac{2}{h_k}\left(x - x_j^k\right)\right), & x_{j-1}^k < x \le x_j^k, \\ 0, & \text{otherwise,} \end{cases} \tag{2.7}$$

where we use the subscript j on the function $\phi_i(x)$ to denote the subinterval of support on the grid level indicated by the superscript k.

We denote the family of $\ell + 1$ polynomials of degree ℓ by the vector

$$\boldsymbol{\phi}_j^k = (\boldsymbol{\phi}(x))_j^k = \left[(\phi_0(x))_j^k, \quad (\phi_1(x))_j^k, \quad \ldots, \quad (\phi_\ell(x))_j^k\right]'. \tag{2.8}$$

In (2.8) and throughout the remainder of this paper we use the prime ($'$) to denote a vector or matrix transpose. (The general family of polynomials (2.8) is considered in section 3 and Appendix D.) The polynomials in the family are orthogonal on the interval $x_{j-1}^k < x \le x_j^k$. For notational simplicity we will not explicitly indicate the dependency of $\boldsymbol{\phi}$ on x, except where necessary for clarity. If we properly select a set of $\ell+1$ coefficients

$$\boldsymbol{\alpha}_j^k = \left[(\alpha_0)_j^k, \quad (\alpha_1)_j^k, \quad \dots, \quad (\alpha_\ell)_j^k\right]', \tag{2.9}$$

then any polynomial function, denoted by $u(x)$, of degree ℓ can be represented exactly on the jth subinterval of the level $k-1$ grid by

$$(u(x))_j^{k-1} = \sum_{i=0}^{\ell} (\alpha_i)_j^{k-1} (\phi_i(x))_j^{k-1} = \boldsymbol{\alpha}_j^{k-1\prime} \boldsymbol{\phi}_j^{k-1}. \tag{2.10}$$

We can represent the same function on two subintervals of grid level k:

$$(u(x))_j^{k-1} = \boldsymbol{\alpha}_{2j-1}^{k\prime} \boldsymbol{\phi}_{2j-1}^k + \boldsymbol{\alpha}_{2j}^{k\prime} \boldsymbol{\phi}_{2j}^k \tag{2.11}$$

(see the relation between grid indexing at level $k-1$ and at level k given by (2.2)).

Since each component of $\boldsymbol{\phi}_j^{k-1}$ is a polynomial, it can also be represented exactly on two subintervals of grid level k, i.e.,

$$\boldsymbol{\phi}_j^{k-1} = \mathbf{C}_0 \boldsymbol{\phi}_{2j-1}^k + \mathbf{C}_1 \boldsymbol{\phi}_{2j}^k, \tag{2.12}$$

where $\mathbf{C}_0$ and $\mathbf{C}_1$ are $(\ell+1)\times(\ell+1)$ matrices whose elements depend on the coefficients ($a_{p,i}$'s) of the family of polynomials $\boldsymbol{\phi}$ given by (2.5).

If we use the identity (2.12) in (2.10), we obtain

$$(u(x))_j^{k-1} = \boldsymbol{\alpha}_j^{k-1\prime} (\mathbf{C}_0 \boldsymbol{\phi}_{2j-1}^k + \mathbf{C}_1 \boldsymbol{\phi}_{2j}^k). \tag{2.13}$$

The two families of polynomials $\boldsymbol{\phi}_{2j-1}^k$ and $\boldsymbol{\phi}_{2j}^k$ have nonoverlapping support, and the ϕ_i's are orthogonal; therefore, we can equate their coefficients from (2.11) and (2.13):

$$\boldsymbol{\alpha}_j^{k-1\prime} \mathbf{C}_0 = \boldsymbol{\alpha}_{2j-1}^{k\prime}, \tag{2.14a}$$

$$\boldsymbol{\alpha}_j^{k-1\prime} \mathbf{C}_1 = \boldsymbol{\alpha}_{2j}^{k\prime}. \tag{2.14b}$$

Taking the transpose of (2.14a), (2.14b), one obtains

$$\boldsymbol{\alpha}_{2j-1}^k = \mathbf{C}_0{}' \boldsymbol{\alpha}_j^{k-1}, \tag{2.15a}$$

$$\boldsymbol{\alpha}_{2j}^k = \mathbf{C}_1{}' \boldsymbol{\alpha}_j^{k-1}. \tag{2.15b}$$

If we multiply (2.15a) by $\mathbf{C}_0$ and multiply (2.15b) by $\mathbf{C}_1$ and add the results, we have

$$\mathbf{C}_0 \boldsymbol{\alpha}_{2j-1}^k + \mathbf{C}_1 \boldsymbol{\alpha}_{2j}^k = (\mathbf{C}_0 \mathbf{C}_0{}' + \mathbf{C}_1 \mathbf{C}_1{}') \boldsymbol{\alpha}_j^{k-1}. \tag{2.16}$$

Since $\phi(\xi)$'s (2.5) are assumed to form an orthonormal basis, it can be shown (Appendix A) that

$$\frac{1}{2}(\mathbf{C}_0\mathbf{C}_0{}' + \mathbf{C}_1\mathbf{C}_1{}') = \mathbf{I}. \tag{2.17}$$

If we use identity (2.17) in (2.16) we obtain

$$\boldsymbol{\alpha}_j^{k-1} = \frac{1}{2}(\mathbf{C}_0\boldsymbol{\alpha}_{2j-1}^k + \mathbf{C}_1\boldsymbol{\alpha}_{2j}^k). \tag{2.18}$$

Let's summarize. If $u(x)$ is a polynomial of degree ℓ on the interval $x_{j-1}^{k-1} < x < x_j^{k-1}$, then (2.15a), (2.15b), and (2.18) provide the necessary equations for manipulating the $\boldsymbol{\alpha}$'s between level k and $k-1$. We will use the terminology that (2.18) provides the "decomposition" from the finer level k to the coarser level $k-1$ and (2.15a), (2.15b) provide the "reconstruction" from the coarser level $k-1$ to the finer level k.

2.3. Decomposition. In general the special case where $u(x)$ is a single low degree polynomial on the unit interval is not of interest. The interesting case is for any piecewise continuous function $u(x)$. In this case (2.15a), (2.15b), and (2.18) will not be identities but only approximations. Therefore, in the general case we must save additional information from the decomposition if we want exact reconstruction of $\boldsymbol{\alpha}^k$. The Harten multiresolution approach is to compute and save the interpolation error or, equivalently, a residual. For the decomposition, in the general case, we will use a weighted form of (2.18), i.e.,

$$\boldsymbol{\alpha}_j^{k-1} = \frac{1}{\sqrt{2}}(\mathbf{C}_0\boldsymbol{\alpha}_{2j-1}^k + \mathbf{C}_1\boldsymbol{\alpha}_{2j}^k), \tag{2.19a}$$

where the weighting is chosen to achieve an orthogonal transformation and an orthonormal wavelet basis.

For the residual we choose

$$\mathbf{r}_j^{k-1} = \frac{1}{\sqrt{2}}(\mathbf{D}_0\boldsymbol{\alpha}_{2j-1}^k + \mathbf{D}_1\boldsymbol{\alpha}_{2j}^k), \tag{2.19b}$$

where the matrices $\mathbf{D}_0$ and $\mathbf{D}_1$ are not yet defined. For future reference we write (2.19a), (2.19b), in matrix form:

$$\begin{bmatrix}\boldsymbol{\alpha}_j^{k-1}\\ \mathbf{r}_j^{k-1}\end{bmatrix} = \frac{1}{\sqrt{2}}\begin{bmatrix}\mathbf{C}_0 & \mathbf{C}_1\\ \mathbf{D}_0 & \mathbf{D}_1\end{bmatrix}\begin{bmatrix}\boldsymbol{\alpha}_{2j-1}^k\\ \boldsymbol{\alpha}_{2j}^k\end{bmatrix}. \tag{2.19c}$$

2.4. Reconstruction. For the reconstruction we use weighted forms of (2.15a), (2.15b), as approximations to which must be added the error $\boldsymbol{\epsilon}$:

$$\boldsymbol{\alpha}_{2j-1}^k = \frac{1}{\sqrt{2}}\mathbf{C}_0{}'\boldsymbol{\alpha}_j^{k-1} + \boldsymbol{\epsilon}_{2j-1}^k, \tag{2.20a}$$

$$\boldsymbol{\alpha}_{2j}^k = \frac{1}{\sqrt{2}}\mathbf{C}_1{}'\boldsymbol{\alpha}_j^{k-1} + \boldsymbol{\epsilon}_{2j}^k. \tag{2.20b}$$

We want the decomposition and reconstruction to be an orthogonal transformation (and the wavelet basis to be orthonormal). Therefore, the reconstruction must

be the transpose of the decomposition, i.e., the matrix in the decomposition (2.19c) must be an orthogonal matrix:

$$\left(\frac{1}{\sqrt{2}}\begin{bmatrix}\mathbf{C}_0 & \mathbf{C}_1\\ \mathbf{D}_0 & \mathbf{D}_1\end{bmatrix}\right)^{-1} = \left(\frac{1}{\sqrt{2}}\begin{bmatrix}\mathbf{C}_0 & \mathbf{C}_1\\ \mathbf{D}_0 & \mathbf{D}_1\end{bmatrix}\right)' = \frac{1}{\sqrt{2}}\begin{bmatrix}\mathbf{C}_0{}' & \mathbf{D}_0{}'\\ \mathbf{C}_1{}' & \mathbf{D}_1{}'\end{bmatrix}.$$

Consequently, the reconstruction step is

$$\boldsymbol{\alpha}_{2j-1}^{k} = \frac{1}{\sqrt{2}}(\mathbf{C}_0{}'\boldsymbol{\alpha}_j^{k-1} + \mathbf{D}_0{}'\mathbf{r}_j^{k-1}), \tag{2.21a}$$

$$\boldsymbol{\alpha}_{2j}^{k} = \frac{1}{\sqrt{2}}(\mathbf{C}_1{}'\boldsymbol{\alpha}_j^{k-1} + \mathbf{D}_1{}'\mathbf{r}_j^{k-1}). \tag{2.21b}$$

For (2.20a), (2.20b) and (2.21a), (2.21b), to be equivalent we must find matrices $\mathbf{D}_0$ and $\mathbf{D}_1$ such that

$$\boldsymbol{\epsilon}_{2j-1}^{k} = \frac{1}{\sqrt{2}}\mathbf{D}_0{}'\mathbf{r}_j^{k-1} \tag{2.22a}$$

and

$$\boldsymbol{\epsilon}_{2j}^{k} = \frac{1}{\sqrt{2}}\mathbf{D}_1{}'\mathbf{r}_j^{k-1}. \tag{2.22b}$$

In Appendix B we show that the requirements are met by the following choices:

$$\mathbf{D}_0 = \sqrt{2}\mathbf{B}\left(-\mathbf{I} + \frac{1}{2}\mathbf{C}_0{}'\mathbf{C}_0 - \frac{1}{2}\mathbf{C}_1{}'\mathbf{C}_0\right), \tag{2.23a}$$

$$\mathbf{D}_1 = \sqrt{2}\mathbf{B}\left(\mathbf{I} - \frac{1}{2}\mathbf{C}_1{}'\mathbf{C}_1 + \frac{1}{2}\mathbf{C}_0{}'\mathbf{C}_1\right), \tag{2.23b}$$

where

$$\mathbf{B} = \frac{1}{\sqrt{2}}\left\{\left[\mathbf{I} - \frac{1}{4}(\mathbf{C}_0 - \mathbf{C}_1)'(\mathbf{C}_0 - \mathbf{C}_1)\right]^{-1}\right\}^{\frac{1}{2}}. \tag{2.23c}$$

It should be noted that this choice is not unique.

2.5. Multiresolution algorithm (pseudocode). In summary, the decomposition from grid level m to grid level n is

For $k = m, m-1, \ldots, n+1$

For $j = 1, 2, \ldots, J_{k-1}$

$$\boldsymbol{\alpha}_j^{k-1} = \frac{1}{\sqrt{2}}(\mathbf{C}_0\boldsymbol{\alpha}_{2j-1}^{k} + \mathbf{C}_1\boldsymbol{\alpha}_{2j}^{k}) \tag{2.24a}$$

$$\mathbf{r}_j^{k-1} = \frac{1}{\sqrt{2}}(\mathbf{D}_0\boldsymbol{\alpha}_{2j-1}^{k} + \mathbf{D}_1\boldsymbol{\alpha}_{2j}^{k}) \tag{2.24b}$$

End

End

and the reconstruction from grid level n to grid level m is

For $k = n+1, n+2, \cdots, m$

For $j = 1, 2, \cdots, J_{k-1}$

$$\boldsymbol{\alpha}_{2j-1}^{k} = \frac{1}{\sqrt{2}}(\mathbf{C}_0{}'\boldsymbol{\alpha}_j^{k-1} + \mathbf{D}_0{}'\mathbf{r}_j^{k-1}) \tag{2.25a}$$

$$\boldsymbol{\alpha}_{2j}^{k} = \frac{1}{\sqrt{2}}(\mathbf{C}_1{}'\boldsymbol{\alpha}_j^{k-1} + \mathbf{D}_1{}'\mathbf{r}_j^{k-1}) \tag{2.25b}$$

End

End

2.6. Multiresolution code. The multiresolution algorithm (2.24) and (2.25) is easily implemented as we demonstrate with a Matlab code. A compact and efficient way to code the algorithm is to start with the input matrix $\mathbf{A}^m$ (2.3) and to overwrite at every step of the decomposition algorithm (2.24) to obtain the decomposed matrix stored in $\mathbf{A}^m$. Likewise, this decomposed matrix is the input matrix of the reconstruction algorithm that is again overwritten at every step of the reconstruction algorithm to obtain the reconstructed matrix stored in $\mathbf{A}^m$.

Decomposition algorithm.

```
function [A]=decom(A,C0,C1,D0,D1)
%multiresolution decomposition using supercompact wavelet basis
% A is the l+1 by J_m matrix of alpha column vectors (2.3)
%C0,C1,D0,D1 are multiresolution matrices (e.g., section 3)
n=0;                              %complete decomposition
[l1,J_m]=size(A);                 %l+1,J_m
m=log2(J_m);
s=1/sqrt(2);
for k=m:-1:n+1
 jk=2^k;                          %J_k
 Ao=s*A(:,1:2:jk-1);              %odd alpha columns
 Ae=s*A(:,2:2:jk);                %even alpha columns
 A(:,1:jk/2)=C0*Ao+C1*Ae;         %decimated alpha columns (2.24a)
 A(:,jk/2+1:jk)=D0*Ao+D1*Ae;      %residual columns r (2.24b)
end
```

Reconstruction algorithm.

```
function [A]=recon(A,C0,C1,D0,D1)
%multiresolution reconstruction using supercompact wavelet basis
% A is the l+1 by J_m decomposition matrix from decom
%C0,C1,D0,D1 are multiresolution matrices (e.g., section 3)
n=0;                              %complete reconstruction
[l1,J_m]=size(A);                 %l+1,J_m
```

```
m=log2(J_m);
s=1/sqrt(2);
for k=n+1:m
 jk=2^k;                          %J_k
 Ao=s*A(:,1:jk/2);                %decimated alpha columns
 Ae=s*A(:,jk/2+1:jk);             %residual columns
 A(:,1:2:jk-1)=C0'*Ao+D0'*Ae      %reconstructed odd columns (2.25a)
 A(:,2:2:jk)=C1'*Ao+D1'*Ae;       % reconstructed even columns (2.25b)
end
```

2.7. Matrix notation. For comparison with the Strang and Strela [7] notation for multiwavelets, it is useful to have the decomposition and reconstruction algorithms in matrix notation. Let $\mathcal{L}$ and $\mathcal{H}$ be the rectangular matrices

$$\mathcal{L} = \frac{1}{\sqrt{2}} \begin{bmatrix} \mathbf{C}_0 & \mathbf{C}_1 & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & \mathbf{C}_0 & \mathbf{C}_1 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \dots & \mathbf{C}_0 & \mathbf{C}_1 \end{bmatrix} \tag{2.26a}$$

and

$$\mathcal{H} = \frac{1}{\sqrt{2}} \begin{bmatrix} \mathbf{D}_0 & \mathbf{D}_1 & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & \mathbf{D}_0 & \mathbf{D}_1 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \dots & \mathbf{D}_0 & \mathbf{D}_1 \end{bmatrix}. \tag{2.26b}$$

NOTE: The matrices are block diagonal (with rectangular blocks on the diagonal), which contrasts with the nonsupercompact case where the matrices are not block diagonal.

If we rewrite the $\boldsymbol{\alpha}_j^k$'s (2.4) as a single column vector

$$\mathcal{A}^k = [\boldsymbol{\alpha}_1^{k\prime}, \boldsymbol{\alpha}_2^{k\prime}, \dots, \boldsymbol{\alpha}_{J_k}^{k\prime}]', \quad J_k = 2^k \tag{2.27}$$

(and similarly for the residual), then the decomposition steps (2.24) can be written (in pseudocode)

For $k = m, m-1, \dots, n+1$

$$\mathcal{A}^{k-1} = \mathcal{L}\mathcal{A}^k \tag{2.28a}$$

$$\mathcal{R}^{k-1} = \mathcal{H}\mathcal{A}^k \tag{2.28b}$$

End

Likewise the reconstruction steps (2.25) become

For $k = n+1, n+2, \dots, m$

$$\mathcal{A}^k = \mathcal{L}'\mathcal{A}^{k-1} + \mathcal{H}'\mathcal{R}^{k-1} \tag{2.29}$$

End

where the size of the matrices $\mathcal{L}$, $\mathcal{H}$, $\mathcal{A}$, and $\mathcal{R}$ is implied from the index k. Because of the block diagonal property, the matrices (2.26a), (2.26b), satisfy the same identities as the matrices $\mathbf{L}$ and $\mathbf{H}$ of Appendix B, i.e., (B.6), (B.8), (B.9), (B.10), and (B.11).

3. Evaluation of multiresolution matrices. The application of the multiresolution algorithm (2.24) and (2.25) requires the matrices $\mathbf{C}_0$, $\mathbf{C}_1$, $\mathbf{D}_0$, and $\mathbf{D}_1$. The $\mathbf{C}$ matrices are defined by (2.12). If we denote the elements of $\mathbf{C}_0$ and $\mathbf{C}_1$ by $c^0_{r,s}$ and $c^1_{r,s}$, respectively, they can be evaluated from the expansions (see Appendix C)

$$\phi_i(\xi) = \sum_{p=0}^{\ell} c^0_{i+1,p+1}\phi_p(2\xi+1), \quad -1 < \xi \le 0, \quad i = 0, 1, \dots, \ell \tag{3.1a}$$

and

$$\phi_i(\xi) = \sum_{p=0}^{\ell} c^1_{i+1,p+1}\phi_p(2\xi-1), \quad 0 < \xi \le 1, \quad i = 0, 1, \dots, \ell. \tag{3.1b}$$

If we use the assumed orthogonality of the ϕ's, we can write

$$c^0_{i+1,j+1} = 2\int_{-1}^{0} \phi_j(2\xi+1)\phi_i(\xi)d\xi, \quad i = 0, 1, \dots, \ell, \quad j = 0, 1, \dots, \ell \tag{3.2a}$$

and

$$c^1_{i+1,j+1} = 2\int_{0}^{1} \phi_j(2\xi-1)\phi_i(\xi)d\xi, \quad i = 0, 1, \dots, \ell, \quad j = 0, 1, \dots, \ell. \tag{3.2b}$$

The corresponding $\mathbf{D}$'s are given by (2.23a), (2.23b), and (2.23c).

The general family of orthonormal polynomials (2.8) is considered in Appendix D. In the examples we use a particular family, i.e., the Legendre polynomials.

EXAMPLE 3.1. As an example, consider Legendre polynomials which are a special case of (2.5). The first five Legendre polynomials are

$$\begin{aligned} &P_0(\xi) = 1, \qquad P_1(\xi) = \xi, \qquad P_2(\xi) = \frac{3}{2}\xi^2 - \frac{1}{2}, \\ &P_3(\xi) = \frac{5}{2}\xi^3 - \frac{3}{2}\xi, \qquad P_4(\xi) = \frac{35}{8}\xi^4 - \frac{15}{4}\xi^2 + \frac{3}{8}. \end{aligned} \tag{3.3}$$

The orthonormal basis polynomials (2.5) are

$$\phi_i(\xi) = \begin{cases} \sqrt{i+\frac{1}{2}}P_i(\xi), & -1 < \xi \le 1, \\ 0, & \text{otherwise.} \end{cases} \tag{3.4}$$

The matrices $\mathbf{C}_0$ and $\mathbf{C}_1$ for $\ell = 2$ are, from (3.2a), (3.2b), and (3.4),

$$\mathbf{C}_0 = \begin{bmatrix} 1 & 0 & 0 \\ -\frac{\sqrt{3}}{2} & \frac{1}{2} & 0 \\ 0 & -\frac{\sqrt{15}}{4} & \frac{1}{4} \end{bmatrix}, \tag{3.5a}$$

$$\mathbf{C}_1 = \begin{bmatrix} 1 & 0 & 0 \\ \frac{\sqrt{3}}{2} & \frac{1}{2} & 0 \\ 0 & \frac{\sqrt{15}}{4} & \frac{1}{4} \end{bmatrix}. \tag{3.5b}$$

The corresponding matrices $\mathbf{D}_0$ and $\mathbf{D}_1$ are obtained from (2.23a), (2.23b), and (2.23c)

$$\mathbf{D}_0 = \begin{bmatrix} -\frac{1}{2} & -\frac{\sqrt{3}}{2} & 0 \\ 0 & -\frac{1}{4} & -\frac{\sqrt{15}}{4} \\ 0 & 0 & -1 \end{bmatrix}, \tag{3.6a}$$

$$\mathbf{D}_1 = \begin{bmatrix} \frac{1}{2} & -\frac{\sqrt{3}}{2} & 0 \\ 0 & \frac{1}{4} & -\frac{\sqrt{15}}{4} \\ 0 & 0 & 1 \end{bmatrix}, \tag{3.6b}$$

and for reference the matrix $\mathbf{B}$ defined by (2.23c) is given by

$$\mathbf{B} = \frac{1}{\sqrt{2}} \begin{bmatrix} 2 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{3.6c}$$

In Appendix E we give the Mathematica code for generating the matrices for any ℓ (within the limits of Mathematica) and Legendre polynomials. We also give the $\mathbf{C}$ and $\mathbf{D}$ matrices for $\ell = 4$.

For completeness we give the $\mathbf{C}$ and $\mathbf{D}$ matrices for $\ell = 0$ and $\ell = 1$. The $\mathbf{C}$'s are submatrices of (3.5); however, the $\mathbf{D}$'s are not submatrices of (3.6).

For $\ell = 0$, i.e., Haar,

$$\mathbf{C}_0 = [1], \quad \mathbf{C}_1 = [1], \quad \mathbf{D}_0 = [-1], \quad \mathbf{D}_1 = [1], \tag{3.7}$$

and for $\ell = 1$

$$\mathbf{C}_0 = \begin{bmatrix} 1 & 0 \\ -\frac{\sqrt{3}}{2} & \frac{1}{2} \end{bmatrix}, \mathbf{C}_1 = \begin{bmatrix} 1 & 0 \\ \frac{\sqrt{3}}{2} & \frac{1}{2} \end{bmatrix}, \mathbf{D}_0 = \begin{bmatrix} -\frac{1}{2} & -\frac{\sqrt{3}}{2} \\ 0 & -1 \end{bmatrix}, \mathbf{D}_1 = \begin{bmatrix} \frac{1}{2} & -\frac{\sqrt{3}}{2} \\ 0 & 1 \end{bmatrix}. \tag{3.8}$$

4. Multiresolution implementation. It is highly unlikely that the information available for multiresolution analysis corresponds to the input array $\mathbf{A}^m$ (2.3). In this section we consider several transformations (or preprocessing procedures) that provide the required algorithm input.

4.1. Analytical function. If $u(x)$ is a piecewise continuous function on $0 < x \leq 1$, then we have a simple transformation. On the finest grid X^m we choose a polynomial degree ℓ that adequately represents the function $u(x)$:

$$(u(x))_j^m \approx \sum_{i=0}^{\ell} (\alpha_i)_j^m (\phi_i(x))_j^m, \quad j = 1, 2, \ldots, J_m. \tag{4.1}$$

Since the basis functions are orthogonal, the coefficients are simply

$$(\alpha_i)_j^m = \frac{2}{h_m} \int_{x_{j-1}^m}^{x_j^m} (u(x))_j^m (\phi_i(x))_j^m dx, \quad j = 1, 2, \ldots, J_m, \quad i = 0, 1, \ldots, \ell. \tag{4.2}$$

4.2. Discrete function values. One of the most useful applications of a multiresolution algorithm is the analysis of a vector of discrete function values. In this section we consider the preprocessing transformation from the function values to the input array (2.3).

Let's begin with the case where $u(x)$ on $0 < x \le 1$ is piecewise polynomial of degree ℓ and where (function and/or derivative) discontinuities, if any, lie only on the grid points of the finest grid X^m (see, e.g., Figure 1(b)). Clearly on each interval $x^m_{j-1} < x \le x^m_j$ of the grid X^m we can represent $u(x)$ exactly by

$$(u(x))^m_j = \sum_{i=0}^{\ell} (\alpha_i)^m_j (\phi_i(x))^m_j, \qquad x^m_{j-1} < x \le x^m_j, \tag{4.3}$$

where $(\phi_i(x))^m_j$ is defined by (2.7). Since, in this ideal case, we have the luxury of an analytic representation on each interval, the α's can be determined by choosing $\ell+1$ values of $u(x)$ on the interval. Let's select values at uniformly spaced subintervals:

$$\mathbf{x}^m_j = \begin{cases} [x^m_j], & \ell = 0, \\ [x^m_{j-1},\ x^m_{j-1+\frac{1}{\ell}},\ x^m_{j-1+\frac{2}{\ell}},\ \dots,\ x^m_j]', & \ell > 0. \end{cases} \tag{4.4}$$

The α's are the solution of the system of algebraic equations

$$(u(\mathbf{x}^m_j))^m_j = \sum_{i=0}^{\ell} (\alpha_i)^m_j (\phi_i(\mathbf{x}^m_j))^m_j, \quad j = 1, 2, \dots, J_m. \tag{4.5}$$

The notation implied by (4.5) is clarified if we write out the expansion for a specific case, say $\ell = 2$:

$$\begin{bmatrix} (u(x^m_{j-1}))^m_j \\ (u(x^m_{j-\frac{1}{2}}))^m_j \\ (u(x^m_j))^m_j \end{bmatrix} = \begin{bmatrix} (\phi_0(x^m_{j-1}))^m_j & (\phi_1(x^m_{j-1}))^m_j & (\phi_2(x^m_{j-1}))^m_j \\ (\phi_0(x^m_{j-\frac{1}{2}}))^m_j & (\phi_1(x^m_{j-\frac{1}{2}}))^m_j & (\phi_2(x^m_{j-\frac{1}{2}}))^m_j \\ (\phi_0(x^m_j))^m_j & (\phi_1(x^m_j))^m_j & (\phi_2(x^m_j))^m_j \end{bmatrix} \begin{bmatrix} (\alpha_0)^m_j \\ (\alpha_1)^m_j \\ (\alpha_2)^m_j \end{bmatrix}, \tag{4.6a}$$

$$j = 1, 2, \dots, J_m.$$

Equation (4.6a) requires data at the midpoints of the finest level subinterval $(u(x^m_{j-\frac{1}{2}}))^m_j$ which appears to be information on a grid finer than the given data. However, in practice we obtain the additional values by using a smaller number of subintervals with more points per subinterval (for higher order methods or larger ℓ). For example, see case (c) in section 6 for $\ell = 2$.

The piecewise polynomial $u(x)$ is assumed to be everywhere continuous from the left. If $u(x)$ has a discontinuity at the left end point, x_{j-1}, of the interval, $(x_{j-1}, x_j]$, then the value of $(u(x))^m_j$ at the left end point is defined by

$$(u(x_{j-1}))^m_j = u^+(x_{j-1}) = \lim_{x \to x^+_{j-1}} u(x).$$

In the implementation of a supercompact multiresolution scheme it will be convenient to write the postprocessing transformation (4.6) from $\boldsymbol{\alpha}^m_j$ to $\mathbf{u}^m_j$ in matrix notation:

$$\mathbf{u}^m_j = \mathbf{T}\boldsymbol{\alpha}^m_j, \qquad j = 1, 2, \dots, J_m, \tag{4.6b}$$

where

$$\mathbf{u}_j^m = (u(\mathbf{x}_j^m))_j^m \tag{4.7}$$

and the matrix $\mathbf{T}$ is the matrix on the right-hand side of (4.6). For the demonstration code we rewrite (4.6a) as

$$\mathbf{U}^m = \mathbf{T}\mathbf{A}^m, \tag{4.8}$$

where

$$\mathbf{U}^m = [\mathbf{u}_1^m, \mathbf{u}_2^m, \dots, \mathbf{u}_{J_m}^m]. \tag{4.9}$$

$\mathbf{U}^m$ and $\mathbf{A}^m$ are $\ell+1$ by J_m matrices, and $\mathbf{T}$ is the $\ell+1$ by $\ell+1$ postprocessing transformation matrix. Note that $\mathbf{T}$ is independent of j. Similarly, the preprocessing transformation from u to α is

$$\mathbf{A}^m = \mathbf{T}^{-1}\mathbf{U}^m. \tag{4.10}$$

EXAMPLE 4.1. If we continue the example of the previous section and choose the Legendre polynomials (3.4) and use (2.7), (4.6) becomes

$$\begin{bmatrix} (u(x_{j-1}^m))_j^m \\ (u(x_{j-\frac{1}{2}}^m))_j^m \\ (u(x_j^m))_j^m \end{bmatrix} = \begin{bmatrix} \sqrt{\frac{1}{2}} & -\sqrt{\frac{3}{2}} & \sqrt{\frac{5}{2}} \\ \sqrt{\frac{1}{2}} & 0 & -\frac{\sqrt{10}}{4} \\ \sqrt{\frac{1}{2}} & \sqrt{\frac{3}{2}} & \sqrt{\frac{5}{2}} \end{bmatrix} \begin{bmatrix} (\alpha_0)_j^m \\ (\alpha_1)_j^m \\ (\alpha_2)_j^m \end{bmatrix}, \quad j = 1, 2, \dots, J_m \tag{4.11a}$$

or

$$\mathbf{U}^m = \mathbf{T}\mathbf{A}^m, \tag{4.11b}$$

and, after solving (4.11) for $\boldsymbol{\alpha}_j^m$,

$$\begin{aligned} (\alpha_0)_j^m &= \frac{\sqrt{2}}{6}[(u(x_{j-1}^m))_j^m + 4(u(x_{j-\frac{1}{2}}^m))_j^m + (u(x_j^m))_j^m], \\ (\alpha_1)_j^m &= \frac{1}{\sqrt{6}}[-(u(x_{j-1}^m))_j^m + (u(x_j^m))_j^m], \\ (\alpha_2)_j^m &= \frac{\sqrt{10}}{15}[(u(x_{j-1}^m))_j^m - 2(u(x_{j-\frac{1}{2}}^m))_j^m + (u(x_j^m))_j^m], \\ j &= 1, 2, \dots, J_m, \end{aligned} \tag{4.12a}$$

or

$$\mathbf{A}^m = \mathbf{T}^{-1}\mathbf{U}^m. \tag{4.12b}$$

Equation (4.12a) has been taken out of matrix form to indicate that α_0, α_1, and α_2 are (weighted) undivided discrete approximations to the cell average, first derivative, and second derivative, respectively.

Equation (4.10), e.g., (4.12a), provides the preprocessing transformation from the piecewise continuous function to the discrete input matrix $\mathbf{A}$ required in the multiresolution algorithm. If the piecewise continuous function $u(x)$ is not polynomial of degree ℓ we can still use (4.5) for an approximate representation.

For completeness we give the postprocessing transformations for $\ell = 0$ and $\ell = 1$ corresponding to the **C** and **D** matrices (3.7) and (3.8):

$$\left[(u(x_j^m))_j^m\right] = \left[\sqrt{\tfrac{1}{2}}\right]\left[(\alpha_0)_j^m\right], \quad \left[(\alpha_0)_j^m\right] = \left[\sqrt{2}\right]\left[(u(x_j^m))_j^m\right], \tag{4.13}$$

$$\begin{bmatrix}(u(x_{j-1}^m))_j^m\\ (u(x_j^m))_j^m\end{bmatrix} = \begin{bmatrix}\sqrt{\frac{1}{2}} & -\sqrt{\frac{3}{2}}\\ \sqrt{\frac{1}{2}} & \sqrt{\frac{3}{2}}\end{bmatrix}\begin{bmatrix}(\alpha_0)_j^m\\ (\alpha_1)_j^m\end{bmatrix}, \quad \begin{bmatrix}(\alpha_0)_j^m\\ (\alpha_1)_j^m\end{bmatrix} = \begin{bmatrix}\sqrt{\frac{1}{2}} & \sqrt{\frac{1}{2}}\\ -\sqrt{\frac{1}{6}} & \sqrt{\frac{1}{6}}\end{bmatrix}\begin{bmatrix}(u(x_{j-1}^m))_j^m\\ (u(x_j^m))_j^m\end{bmatrix}. \tag{4.14}$$

Return now to the case where only the vector of discrete function values (with multiple values for discontinuities) is given. An obvious preprocessing transformation to the α's is to require point values at grid spacing h_m/ℓ (for $\ell > 1$) rather than h_m on the interval $0 < x \le 1$. Now we can use the postprocessing transformation (4.5) for a vector of discrete function values.

4.3. Discrete cell average values and derivative values. In some applications it is convenient to represent a function by discrete function values, discrete function derivative values, and/or discrete integral values. We could also include multiple values of each, i.e., choose multiple locations for interpolation as we did in the previous section. For demonstration we choose discrete cell average values and the first p derivatives evaluated at a single interpolation point, e.g., the midpoint of the interval:

$$\mathbf{x}_j^m = \left[x_{j-\frac{1}{2}}^m\right]. \tag{4.15}$$

The cell average is defined by

$$(\bar{u})_j^m = \frac{1}{h_m}\int_{x_{j-1}^m}^{x_j^m} (u(\eta))_j^m d\eta. \tag{4.16}$$

We choose $\ell = p$. The α's are the solution to the system of $\ell + 1$ algebraic equations:

$$\begin{aligned}
(\bar{u})_j^m &= \frac{1}{h_m}\sum_{i=0}^{\ell}(\alpha_i)_j^m \int_{x_{j-1}^m}^{x_j^m} (\phi_i(\eta))_j^m d\eta,\\
\tfrac{d}{dx}\left(u\left(x_{j-\frac{1}{2}}^m\right)\right)_j^m &= \sum_{i=0}^{\ell}(\alpha_i)_j^m \frac{d}{dx}\left(\phi_i\left(x_{j-\frac{1}{2}}^m\right)\right)_j^m,\\
&\vdots\\
\frac{d^p}{dx^p}\left(u\left(x_{j-\frac{1}{2}}^m\right)\right)_j^m &= \sum_{i=0}^{\ell}(\alpha_i)_j^m \frac{d^p}{dx^p}\left(\phi_i\left(x_{j-\frac{1}{2}}^m\right)\right)_j^m, \quad j = 1, 2, \ldots, J_m.
\end{aligned} \tag{4.17}$$

Example 4.2. If we choose the Legendre polynomials with $\ell = 2$, the solution

of (4.17) produces

$$
\begin{aligned}
(\alpha_0)_j^m &= \sqrt{2}(\bar{u})_j^m, \\
(\alpha_1)_j^m &= \frac{1}{\sqrt{6}} h_m \frac{d}{dx}\left(u\left(x_{j-\frac{1}{2}}^m\right)\right)_j^m, \\
(\alpha_2)_j^m &= \frac{\sqrt{10}}{15}\left(\frac{h_m}{2}\right)^2 \frac{d^2}{dx^2}\left(u\left(x_{j-\frac{1}{2}}^m\right)\right)_j^m, \quad j = 1, 2, \ldots, J_m.
\end{aligned} \tag{4.18}
$$

For this choice of variables (cell averages and derivatives) and polynomials (Legendre) there is a one-to-one correspondence ($\mathbf{T}$ is a diagonal matrix) between the α's and the variables. Note that (4.12) is a difference approximation to (4.18). This example can be included in the notation (4.8) if the vector $\mathbf{u}_j^m$ (4.7) is redefined:

$$
\mathbf{u}_j^m = \left[(\bar{u})_j^m, \frac{d}{dx}\left(u\left(x_{j-\frac{1}{2}}^m\right)\right)_j^m, \frac{d^2}{dx^2}\left(u\left(x_{j-\frac{1}{2}}^m\right)\right)_j^m\right]'. \tag{4.19}
$$

4.4. Accuracy of a multiwavelet analysis. From section 2.2 it is clear that if we expand a polynomial function of degree ℓ in a family of orthogonal polynomials of degree ℓ, the expansion will be exact and the interpolation error or residual defined by (2.19b) will be zero. In terms of a multiresolution algorithm, the order of accuracy, or number of vanishing moments, can be defined as follows.

Let the function $u(x)$ be the polynomial

$$
u(x) = x^q, \qquad 0 \le x \le 1, \tag{4.20}
$$

where q is a nonnegative integer. From the decomposition step (2.24b), the residual vector for $k = m$ is

$$
\mathbf{r}_j^{m-1} = \frac{1}{\sqrt{2}}(\mathbf{D}_0 \boldsymbol{\alpha}_{2j-1}^m + \mathbf{D}_1 \boldsymbol{\alpha}_{2j}^m), \quad j = 1, \ldots, J_{m-1}. \tag{4.21}
$$

The number p of vanishing residuals (moments) of a multiwavelet algorithm is defined by

$$
\left[\mathbf{r}_j^{m-1} = 0, \quad j = 1, 2, \ldots, J_{m-1}\right], \qquad q = 0, 1, \ldots, p-1 \tag{4.22a}
$$

and

$$
\left[\mathbf{r}_j^{m-1} \neq 0, \quad j = 1, 2, \ldots, J_{m-1}\right], \qquad q = p. \tag{4.22b}
$$

The order of accuracy is defined to be p which is equal to $\ell + 1$.

As a consistency check on the programming of the preprocessing step yielding the input matrix $\mathbf{A}^m$ and the decomposition algorithm (2.24), one should verify the accuracy numerically by checking for the proper number of vanishing moments. After one step of the decomposition algorithm of section 2.6, the *right half* of the matrix $\mathbf{A}$ should be zero for $q \le p - 1 = \ell$ since it contains the residual vectors $\mathbf{r}_j^{m-1}$.

As an example, for the discrete function values of section 4.2 with $\ell = 2$, one obtains for the polynomial (4.20) the vector (4.8):

$$
\mathbf{u}_j^m = \begin{bmatrix} ((x^q)_{j-1}^m)_j^m \\ \left((x^q)_{j-\frac{1}{2}}^m\right)_j^m \\ ((x^q)_j^m)_j^m \end{bmatrix}.
$$

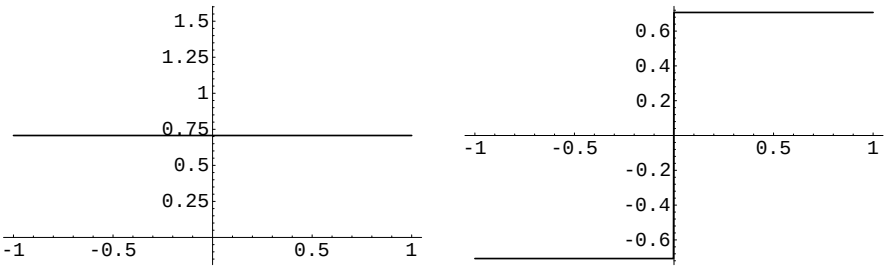


FIG. 3(a). *Scaling functions (left-hand column) and wavelets (right-hand column),* $\ell = 0$.

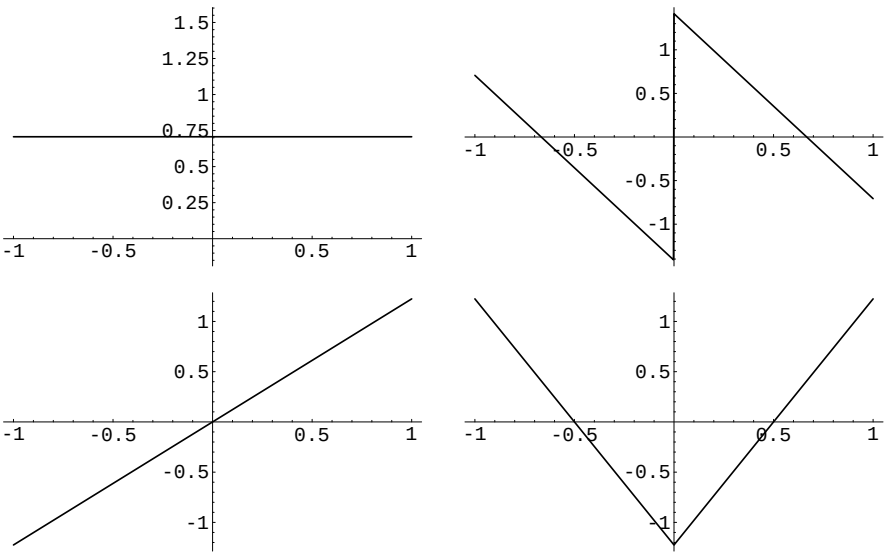


FIG. 3(b). *Scaling functions (left-hand column) and wavelets (right-hand column),* $\ell = 1$.

One finds that the number of vanishing residuals is $p = \ell + 1 = 3$ and the order of accuracy is 3.

5. Scaling functions and wavelets. Neither the development nor the application of the multiresolution algorithm (2.24) and (2.25) requires an explicit knowledge of the multiwavelets. (The multiscaling functions are the orthonormal polynomials (2.5), which must be known in order to compute the $\mathbf{C}$ and $\mathbf{D}$ matrices.) However, curiosity alone is sufficient to warrant an exposition of the wavelet basis functions.

The decomposition (2.11), i.e.,

$$(u(x))_j^{k-1} = \boldsymbol{\alpha}_{2j-1}^{k\,\prime} \boldsymbol{\phi}_{2j-1}^k + \boldsymbol{\alpha}_{2j}^{k\,\prime} \boldsymbol{\phi}_{2j}^k \tag{5.1}$$

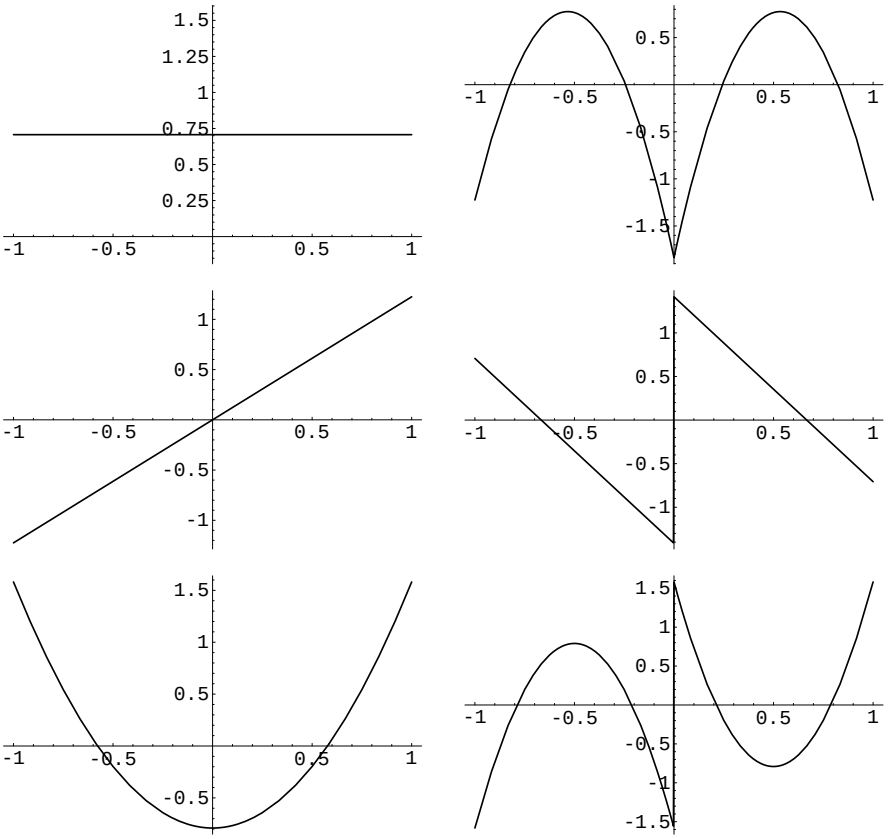


FIG. 3(c). *Scaling functions (left-hand column) and wavelets (right-hand column),* $\ell = 2$.

is exact if $u(x)$ is a polynomial of degree ℓ. In the general (nonexact) case we must replace the $\boldsymbol{\alpha}$'s with unweighted (2.20a), (2.20b):

$$(u(x))_j^{k-1} = [\mathbf{C}_0{}'\boldsymbol{\alpha}_j^{k-1} + \boldsymbol{\epsilon}_{2j-1}^k]'\boldsymbol{\phi}_{2j-1}^k + [\mathbf{C}_1{}'\boldsymbol{\alpha}_j^{k-1} + \boldsymbol{\epsilon}_{2j}^k]'\boldsymbol{\phi}_{2j}^k. \tag{5.2}$$

The error in the decomposition of $(u(x))_j^{k-1}$ is denoted by ϵ_u:

$$(\epsilon_u)_j^{k-1} = (\boldsymbol{\epsilon}_{2j-1}^k)'\boldsymbol{\phi}_{2j-1}^k + (\boldsymbol{\epsilon}_{2j}^k)'\boldsymbol{\phi}_{2j}^k. \tag{5.3}$$

If we now use unweighted (2.22a), (2.22b) in (5.3) and rearrange terms we obtain

$$(\epsilon_u)_j^{k-1} = \mathbf{r}_j^{k-1\prime}\boldsymbol{\psi}_j^{k-1}, \tag{5.4}$$

where $\boldsymbol{\psi}_j^{k-1}$ is defined by

$$\boldsymbol{\psi}_j^{k-1} = \mathbf{D}_0\boldsymbol{\phi}_{2j-1}^k + \mathbf{D}_1\boldsymbol{\phi}_{2j}^k \tag{5.5}$$

and

$$\boldsymbol{\psi}_j^k = (\boldsymbol{\psi}(x))_j^k = \left[(\psi_0(x))_j^k, \quad (\psi_1(x))_j^k, \quad \ldots, \quad (\psi_\ell(x))_j^k\right]'.$$

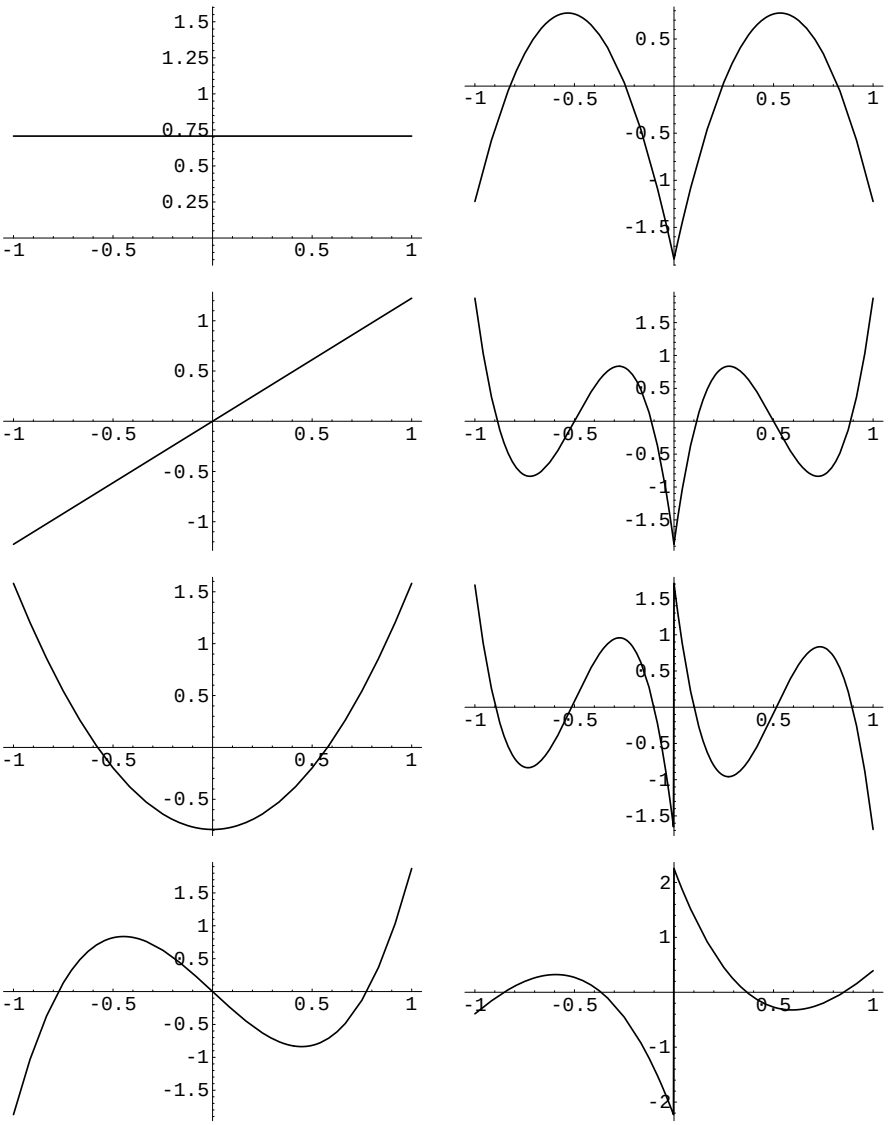


FIG. 3(d). *Scaling functions (left-hand column) and wavelets (right-hand column),* $\ell = 3$.

The orthonormal polynomials $\boldsymbol{\psi}$ are the multiwavelets that form the basis for the expansion of the error (with coefficients $\mathbf{r}$). In the following example, we display wavelets transformed to the ξ-space.

EXAMPLE 5.1. We continue our example with Legendre polynomials (3.4). It is easier to evaluate the multiwavelets defined by (5.5) by transforming the components of the vectors ϕ^k_{2j-1} and ϕ^k_{2j} to a form where the $\phi_i(\xi)$'s appear explicitly (see the transformations (C.3) of Appendix C). The resulting analytical multiwavelets given below are defined on the interval $(-1, 1]$.

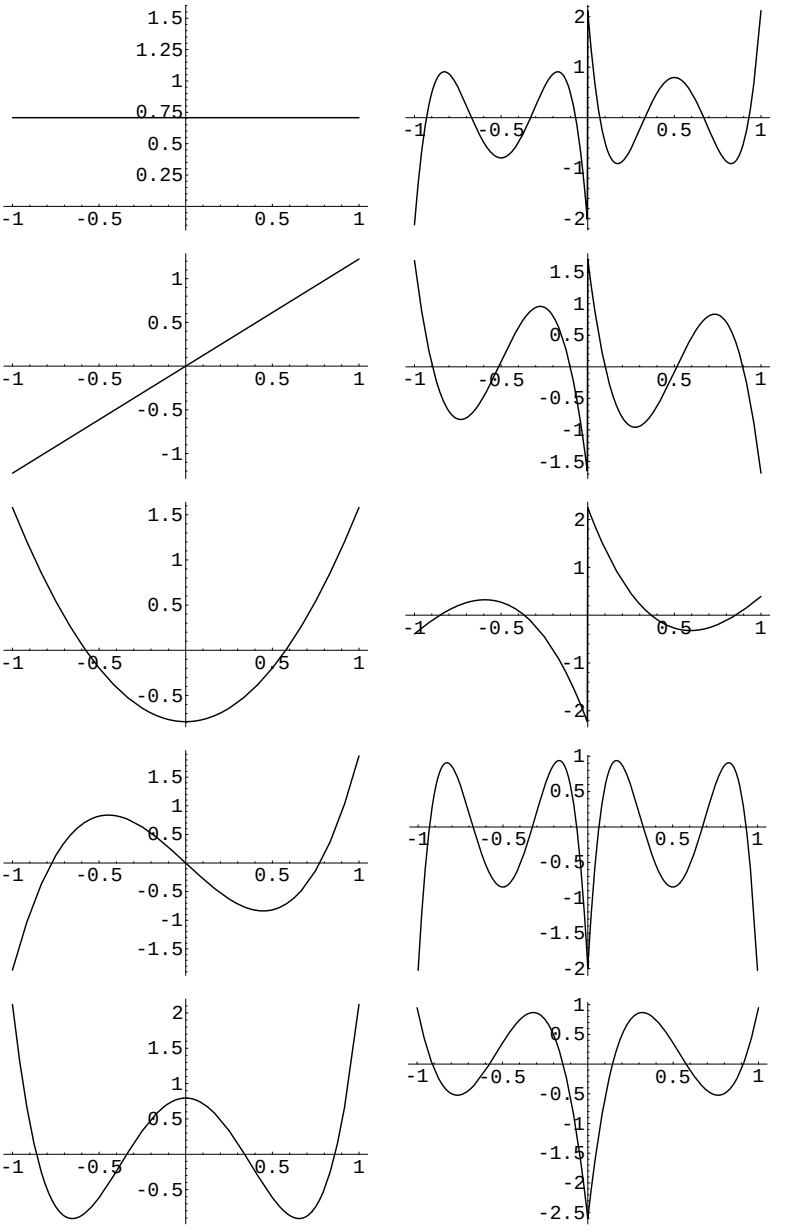


FIG. 3(e). *Scaling functions (left-hand column) and wavelets (right-hand column),* $\ell = 4$.

$\ell = 0$:

$$\psi_0(\xi) = \begin{cases} -\sqrt{\frac{1}{2}}, & -1 < \xi \le 0, \\ \sqrt{\frac{1}{2}}, & 0 < \xi \le 1; \end{cases} \tag{5.6}$$

$\ell = 1$:

$$\psi_0(\xi) = \begin{cases} -\sqrt{\frac{1}{2}}(2+3\xi), & -1 < \xi \le 0, \\ \sqrt{\frac{1}{2}}(2-3\xi), & 0 < \xi \le 1, \end{cases} \tag{5.7a}$$

$$\psi_1(\xi) = \begin{cases} -\sqrt{\frac{3}{2}}(1+2\xi), & -1 < \xi \le 0, \\ \sqrt{\frac{3}{2}}(-1+2\xi), & 0 < \xi \le 1; \end{cases} \tag{5.7b}$$

$\ell = 2$:

$$\psi_0(\xi) = \begin{cases} -\frac{1}{2}\sqrt{\frac{3}{2}}(3+16\xi+15\xi^2), & -1 < \xi \le 0, \\ \frac{1}{2}\sqrt{\frac{3}{2}}(-3+16\xi-15\xi^2), & 0 < \xi \le 1, \end{cases} \tag{5.8a}$$

$$\psi_1(\xi) = \begin{cases} -\sqrt{\frac{1}{2}}(2+3\xi), & -1 < \xi \le 0, \\ \sqrt{\frac{1}{2}}(2-3\xi), & 0 < \xi \le 1, \end{cases} \tag{5.8b}$$

$$\psi_2(\xi) = \begin{cases} -\sqrt{\frac{5}{2}}(1+6\xi+6\xi^2), & -1 < \xi \le 0, \\ \sqrt{\frac{5}{2}}(1-6\xi+6\xi^2), & 0 < \xi \le 1. \end{cases} \tag{5.8c}$$

Plots of the multiscaling functions and multiwavelets for the Legendre polynomials and $\ell = 0, 1, 2, 3, 4$ are shown in Figures 3(a)–3(e). Discrete multiscaling functions and multiwavelets are obtained by evaluating $\phi_i(x)$ and $\psi_i(x)$ on the grid X^k.

6. Applications. Two typical applications for multiresolution algorithms are the analysis of data by decomposition into scales and the compression of data after decomposition. In this section we give some simple examples that demonstrate the attributes of multiresolution using supercompact multiwavelets. We choose the discrete value examples of sections 4.2 and 4.3. The preprocessing transformations for these examples fall within the general notation (4.8), i.e.,

$$\mathbf{U}^m = \mathbf{T}\mathbf{A}^m.$$

The examples are best presented in the context of a multiresolution analysis demonstration code. As an example we use the following Matlab code.

```
function [dA,rU]=mranal(U,C0,C1,D0,D1,T)
%MATLAB Multiresolution Analysis Demonstration Code
%  INPUT
%U is an l+1 by J_m matrix defined in section 4
%C0,C1,D0,D1 are l+1 by l+1 matrices, section 3
%T is the postprocessing transformation matrix U=TA (alpha to u) (4.8)
%  OUTPUT
%dA is the decomposition of alpha (A) (2.24)
%rU is the reconstructed U matrix section 4
 %preprocessing transform U to A (u to alpha)
 A=T^(-1)*U;
 %decomposition of alpha (dA) (2.24) section 2.6 for decom code
 dA=decom(A,C0,C1,D0,D1);
 %truncate data (dA)
  %normally at this step the decomposition would
  %be truncated ("small" elements set to zero) if data compression
  % is desired,  for this example code no compression is done
  %so the reconstruction will be exact (rU=U)
 %reconstruction (rA) (2.25) section 2.6 for recon code
 rA=recon(dA,C0,C1,D0,D1);
 %postprocessing transform reconstructed alpha (rA) to reconstructed u (rU)
 rU=T*rA;
```

EXAMPLE 6.1. We have selected a simple example which is presented in sufficient detail to illustrate (1) the most intimate features of the algorithm implementation, (2) the improved "compressibility" of the decomposition with increasing order of approximation, (3) the supercompact multiwavelet resolution of discontinuities, and (4) the "filled-in" function or subinterval function representation (discussed in detail at the end of this section). For $u(x)$ on $0 < x \le 1$ we choose

$$u(x) = \begin{cases} 1 + 256x^2, & 0 < x \le \dfrac{1}{4}, \\ 17 - 512(x - \frac{1}{4})^2, & \dfrac{1}{4} < x \le \dfrac{1}{2}, \\ 16 - 256(x - \frac{1}{2})^2, & \dfrac{1}{2} < x \le 1, \end{cases} \tag{6.1}$$

which is plotted in Figure 4(d).

For the fine grid we choose $m = 4$ ($J_m = 16$ and $h_m = 1/16$) and evaluate $u(x)$ and $u^+(x)$ on the fine grid:

(6.2)

j	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
x_j	0	$\frac{1}{16}$	$\frac{2}{16}$	$\frac{3}{16}$	$\frac{4}{16}$	$\frac{5}{16}$	$\frac{6}{16}$	$\frac{7}{16}$	$\frac{8}{16}$	$\frac{9}{16}$	$\frac{10}{16}$	$\frac{11}{16}$	$\frac{12}{16}$	$\frac{13}{16}$	$\frac{14}{16}$	$\frac{15}{16}$	$\frac{16}{16}$
u_j	$*$	2	5	10	17	15	9	-1	-15	15	12	7	0	-9	-20	-33	-48
u_j^+	1	2	5	10	17	15	9	-1	16	15	12	7	0	-9	-20	-33	$*$

,

where the * indicates that the value is not defined.

For the numerical example we assume that the given data are the discrete values u_j and u_j^+, and we will use the discrete function value implementation of section 4.2. For

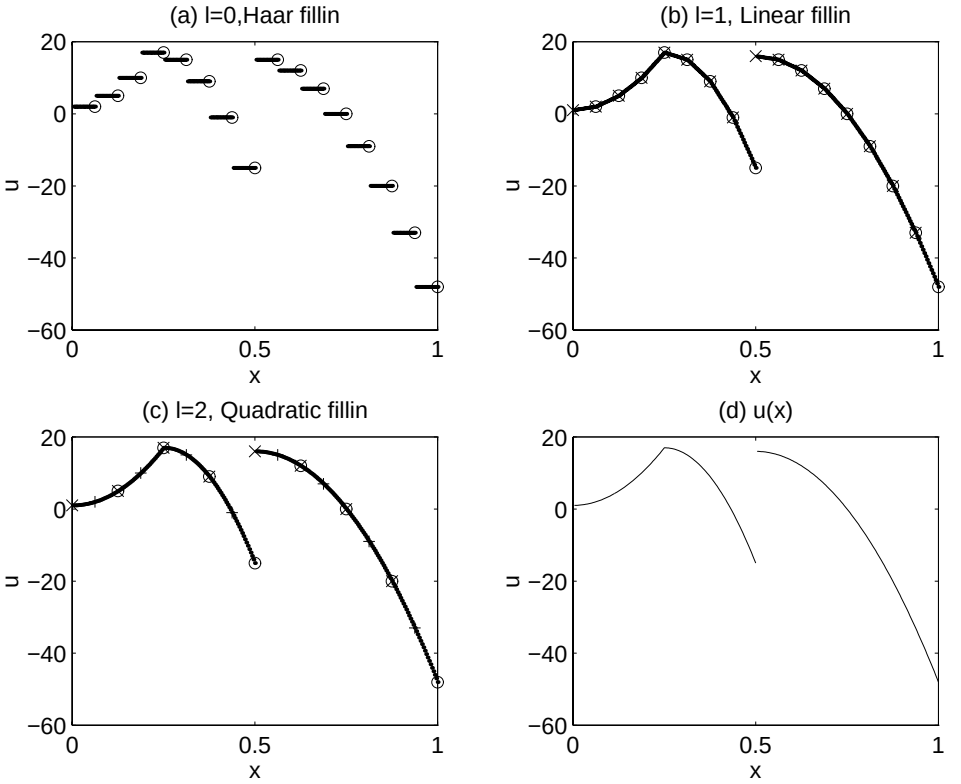


FIG. 4. *Filled-in functions for the discrete data* (6.2).

each of three multiresolution algorithms ($\ell = 0, 1, 2$) we compute $\mathbf{U}^m$, the decomposed values of $\mathbf{A}$ (i.e., $d\mathbf{A}$), and the filled-in function. First we present the numerical input and results for each of the cases, and then we follow with the discussion.

(a) $\ell = 0$, constant interpolation (Haar). For $\ell = 0$ we need the discrete function value at the right-hand end of each interval, and we use the given values of u_j; i.e., $\mathbf{U}^m$, (4.8), is

(6.3)
$$\mathbf{U}^m = \begin{bmatrix} 2 & 5 & 10 & 17 & 15 & 9 & -1 & -15 & 15 & 12 & 7 & 0 & -9 & -20 & -33 & -48 \end{bmatrix}.$$

The decomposed (residual) values $d\mathbf{A}$ (from the demonstration code) are plotted in Figure 5(a) with the symbol $\circ$. The filled-in function is plotted in Figure 4(a) where the dots are the filled-in function and the $\circ$'s are the discrete u_j input values (6.3).

(b) $\ell = 1$, linear interpolation. For $\ell = 1$ we need the discrete function values at each end of each interval. For the left-hand end we use the u_j^+ values, and for the right-hand end we use the u_j values, i.e.,

(6.4)
$$\mathbf{U}^m = \begin{bmatrix} 1 & 2 & 5 & 10 & 17 & 15 & 9 & -1 & 16 & 15 & 12 & 7 & 0 & -9 & -20 & -33 \\ 2 & 5 & 10 & 17 & 15 & 9 & -1 & -15 & 15 & 12 & 7 & 0 & -9 & -20 & -33 & -48 \end{bmatrix}.$$

Note the repetition of values between rows except at the boundaries and the function discontinuity. The decomposed values of $\mathbf{A}$ (from the demonstration code) are plotted

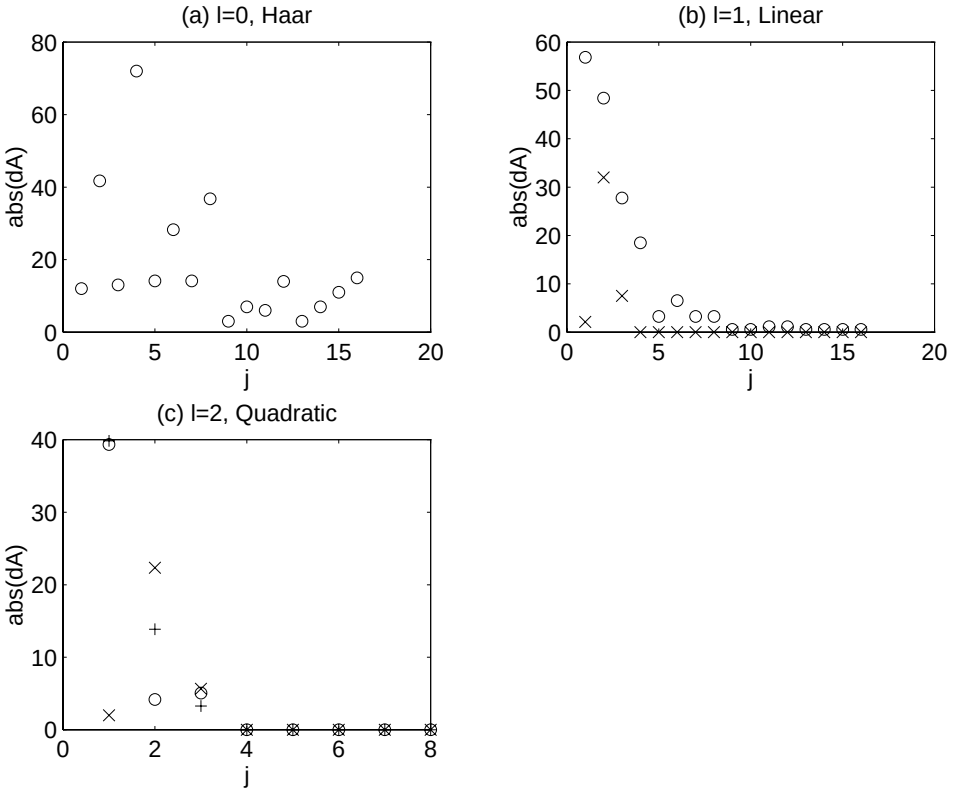


FIG. 5. *Decomposed* $\mathbf{A}$ *values,* $d\mathbf{A}$*, from the demonstration code applied to the discrete data* (6.2).

in Figure 5(b). The first row matrix values are plotted with the symbol $\times$, and the second row matrix values are plotted with the symbol $\circ$. The filled-in function is plotted in Figure 4(b), where the dots are the filled-in function, the $\times$'s are the discrete u_j^+ input values, and the $\circ$'s are the discrete u_j values (6.4).

(c) $\ell = 2$, quadratic interpolation. For $\ell = 2$ we need interval midpoint values in addition to the end point values. For this case we use only 8 intervals instead of 16 and use the "extra" function values as the midpoint values, i.e.,

$$\mathbf{U}^m = \begin{bmatrix} 1 & 5 & 17 & 9 & 16 & 12 & 0 & -20 \\ 2 & 10 & 15 & -1 & 15 & 7 & -9 & -33 \\ 5 & 17 & 9 & -15 & 12 & 0 & -20 & -48 \end{bmatrix}. \tag{6.5}$$

The decomposed (residual) values $d\mathbf{A}$ (from the demonstration code) are plotted in Figure 5(c). The first row matrix values are plotted with the symbol $\times$, the second row matrix values are plotted with the + symbol, and the third row matrix values are plotted with the symbol $\circ$. The filled-in function is plotted in Figure 4(c), where the dots are the filled-in function, the $\times$'s are the discrete u_j^+ input values, the +'s are the discrete u_j "midpoint" values, and the $\circ$'s are the discrete u_j "end-point" values (6.5).

The decomposed (residual) values $d\mathbf{A}$, Figure 5, provide a measure of the compressibility of the data; the more "small" values, the more the data can be truncated or compressed without significant loss of information. Note that as the order of

approximation increases, ℓ increases, and the number of relatively small $d\mathbf{A}$ values increases. Since the original function is piecewise quadratic, the case $\ell = 2$ has only nine nonzero $d\mathbf{A}$ values—the minimum number of values necessary to represent the three quadratic segments of $u(x)$. Note also that most of the $\times$ symbols of Figure 5(b) ($\ell = 1$) are zero. This is a result of having only one function discontinuity while the algorithm has the capability to support a discontinuity at each fine grid point.

The filled-in function (Figure 4) is the subinterval function representation (or subgrid resolution) of the particular interpolation approximation, i.e., $\ell = 0, 1, 2$. The case $\ell = 0$, or Haar, is "constant" interpolation, $\ell = 1$ is linear interpolation, and $\ell = 2$ is quadratic interpolation, which for the present example is an exact representation of the original function $u(x)$. There are several methods to numerically compute the filled-in functions (in addition to the analytical option). The filled-in functions plotted in Figure 4 were obtained by "padding" the decomposed $\mathbf{A}$ matrices ($d\mathbf{A}$) with zeros, multiplying by the appropriate power (which depends on the number of filled-in points) of the weighting factor ($\sqrt{2}$), applying the reconstruction code (recon), and multiplying the result by the postprocessing transformation matrix ($\mathbf{T}$). In Matlab speak this process is

```
Uf=T*recon([dA zeros(l+1,2^nf-2^ni)]*sqrt(2)^(nf-ni),C0,C1,D0,D1),
```

where Uf is the filled-in $\mathbf{U}$, nf is the $\log_2$ of the number of columns in the filled-in matrix Uf, and ni is the $\log_2$ of the number of columns in the matrix dA.

In many (perhaps most) applications of multiresolution algorithms, the given data are a discrete sampling of a function, e.g., in time or space. Therefore, rarely would multiple values, i.e., the u^+ values (or jump values) be available. For optimum performance of the supercompact algorithms, these additional values are required. We are currently testing algorithms that insert the multiple values (u^+) in the original sampled data. One strategy inserts discontinuities when the difference of one-sided (left *and* right) finite differences exceeds a local average value of the finite differences. These results, as well as those for data compression, will be presented elsewhere.

7. Concluding remarks. The multiresolution analysis with supercompact multiscaling functions (and wavelets) is simple to implement and has special appeal in applications on a finite interval and/or for analysis of functions that are piecewise continuous (internal boundaries). Thus the simplicity of the Haar wavelets is preserved while increased accuracy (more vanishing moments) is achieved.

Appendix A. The $\phi(\xi)$'s (2.5) are assumed to form an orthonormal basis. The $\phi(x)$'s (2.7) are orthogonal:

$$\int_{x_{j-1}^{k-1}}^{x_j^{k-1}} \boldsymbol{\phi}_j^{k-1} \boldsymbol{\phi}_j^{k-1\prime} dx = \frac{h_{k-1}}{2}\mathbf{I} \tag{A.1}$$

where $\mathbf{I}$ is the identity matrix of order $\ell + 1$. Using (2.12), one obtains

$$\int_{x_{j-1}^{k-1}}^{x_j^{k-1}} (\mathbf{C}_0\boldsymbol{\phi}_{2j-1}^k + \mathbf{C}_1\boldsymbol{\phi}_{2j}^k)(\mathbf{C}_0\boldsymbol{\phi}_{2j-1}^k + \mathbf{C}_1\boldsymbol{\phi}_{2j}^k)' dx = \frac{h_{k-1}}{2}\mathbf{I}. \tag{A.2}$$

If we expand the integrand of (A.2)

$$\int_{x_{j-1}^{k-1}}^{x_j^{k-1}} (\mathbf{C}_0\boldsymbol{\phi}_{2j-1}^k\boldsymbol{\phi}_{2j-1}^{k\prime}\mathbf{C}_0' + \mathbf{C}_0\boldsymbol{\phi}_{2j-1}^k\boldsymbol{\phi}_{2j}^{k\prime}\mathbf{C}_1' + \mathbf{C}_1\boldsymbol{\phi}_{2j}^k\boldsymbol{\phi}_{2j-1}^{k\prime}\mathbf{C}_0' + \mathbf{C}_1\boldsymbol{\phi}_{2j}^k\boldsymbol{\phi}_{2j}^{k\prime}\mathbf{C}_1')dx \tag{A.3}$$

and use the identities

$$\int_{x_{j-1}^{k-1}}^{x_j^{k-1}} \phi_{2j-1}^k \phi_{2j-1}^{k\,\prime}\, dx = \frac{h_{k-1}}{4}\mathbf{I}, \qquad \int_{x_{j-1}^{k-1}}^{x_j^{k-1}} \phi_{2j-1}^k \phi_{2j}^{k\,\prime}\, dx = 0,$$

$$\int_{x_{j-1}^{k-1}}^{x_j^{k-1}} \phi_{2j}^k \phi_{2j-1}^{k\,\prime}\, dx = 0, \qquad \int_{x_{j-1}^{k-1}}^{x_j^{k-1}} \phi_{2j}^k \phi_{2j}^{k\,\prime}\, dx = \frac{h_{k-1}}{4}\mathbf{I}, \tag{A.4}$$

then we have

$$\frac{1}{2}(\mathbf{C}_0\mathbf{C}_0{}' + \mathbf{C}_1\mathbf{C}_1{}') = \mathbf{I}. \tag{A.5}$$

Appendix B. In this appendix we derive (2.23). First we offer some notation to simplify the algebra. Let

$$\mathbf{L} = \frac{1}{\sqrt{2}}\begin{bmatrix}\mathbf{C}_0 & \mathbf{C}_1\end{bmatrix}, \tag{B.1}$$

$$\mathbf{H} = \frac{1}{\sqrt{2}}\begin{bmatrix}\mathbf{D}_0 & \mathbf{D}_1\end{bmatrix}. \tag{B.2}$$

The objective is to find $\mathbf{D}_0$ and $\mathbf{D}_1$ if we are given $\mathbf{C}_0$, $\mathbf{C}_1$ and relations (2.19a), (2.19b), (2.20a), (2.20b), (2.22a), (2.22b), and identity (2.17). In the notation of (B.1) and (B.2) we are given $\mathbf{L}$ and

$$\boldsymbol{\alpha}^{k-1} = \mathbf{L}\boldsymbol{\alpha}^k, \tag{B.3}$$

$$\mathbf{r}^{k-1} = \mathbf{H}\boldsymbol{\alpha}^k, \tag{B.4}$$

$$\boldsymbol{\alpha}^k = \mathbf{L}'\boldsymbol{\alpha}^{k-1} + \mathbf{H}'\mathbf{r}^{k-1}, \tag{B.5}$$

$$\mathbf{L}\mathbf{L}' = \mathbf{I}. \tag{B.6}$$

For simplicity of notation we have used

$$\mathbf{r}_j^{k-1} \to \mathbf{r}^{k-1}, \boldsymbol{\alpha}_j^{k-1} \to \boldsymbol{\alpha}^{k-1}, \begin{bmatrix}\boldsymbol{\alpha}_{2j-1}^k \\ \boldsymbol{\alpha}_{2j}^k\end{bmatrix} \to \boldsymbol{\alpha}^k. \tag{B.7}$$

B.1. Useful identities. In the following derivation it will be useful to have some identities that can be derived from (B.3), (B.4), (B.5), and (B.6). If we use (B.5) in (B.3), then

$$\boldsymbol{\alpha}^{k-1} = \mathbf{L}\mathbf{L}'\boldsymbol{\alpha}^{k-1} + \mathbf{L}\mathbf{H}'\mathbf{r}^{k-1}.$$

But if we use (B.6) and simplify,

$$\mathbf{L}\mathbf{H}'\mathbf{r}^{k-1} = 0$$

or

$$\mathbf{LH}' = 0 \tag{B.8}$$

since $\mathbf{r}^{k-1}$ is arbitrary. The transpose of (B.8) is

$$\mathbf{HL}' = 0. \tag{B.9}$$

If we use (B.3) and (B.4) in (B.5), then

$$\boldsymbol{\alpha}^k = \mathbf{L}'\mathbf{L}\boldsymbol{\alpha}^k + \mathbf{H}'\mathbf{H}\boldsymbol{\alpha}^k$$

or

$$\mathbf{I} = \mathbf{L}'\mathbf{L} + \mathbf{H}'\mathbf{H}. \tag{B.10}$$

Finally, if we use (B.5) in (B.4)

$$\mathbf{r}^{k-1} = \mathbf{HL}'\boldsymbol{\alpha}^{k-1} + \mathbf{HH}'\mathbf{r}^{k-1},$$

and if we now use (B.9), the result is

$$\mathbf{I} = \mathbf{HH}'. \tag{B.11}$$

B.2. Finding H. Now let's return to the objective of finding $\mathbf{H}$, given $\mathbf{L}$. Start with (B.4)

$$\mathbf{r}^{k-1} = \mathbf{H}\boldsymbol{\alpha}^k,$$

then premultiply by $\mathbf{H}'$ to obtain

$$\mathbf{H}'\mathbf{r}^{k-1} = \mathbf{H}'\mathbf{H}\boldsymbol{\alpha}^k,$$

and use (B.10) to obtain

$$\mathbf{H}'\mathbf{r}^{k-1} = (\mathbf{I} - \mathbf{L}'\mathbf{L})\boldsymbol{\alpha}^k. \tag{B.12}$$

Let the left-hand side of (B.12) be the error vector, i.e. (see (2.22a), (2.22b)),

$$\boldsymbol{\epsilon}^k = \begin{bmatrix} \boldsymbol{\epsilon}^k_{2j-1} \\ \boldsymbol{\epsilon}^k_{2j} \end{bmatrix} = \mathbf{H}'\mathbf{r}^{k-1} \tag{B.13}$$

and

$$\begin{bmatrix} \boldsymbol{\epsilon}^k_{2j-1} \\ \boldsymbol{\epsilon}^k_{2j} \end{bmatrix} = (\mathbf{I} - \mathbf{L}'\mathbf{L})\boldsymbol{\alpha}^k. \tag{B.14}$$

If we substitute $\mathbf{L}$ from (B.1) into (B.14) we obtain

$$\begin{bmatrix} \boldsymbol{\epsilon}^k_{2j-1} \\ \boldsymbol{\epsilon}^k_{2j} \end{bmatrix} = \begin{bmatrix} (\mathbf{I} - \frac{1}{2}\mathbf{C}'_0\mathbf{C}_0) & (-\frac{1}{2}\mathbf{C}'_0\mathbf{C}_1) \\ (-\frac{1}{2}\mathbf{C}'_1\mathbf{C}_0) & (\mathbf{I} - \frac{1}{2}\mathbf{C}'_1\mathbf{C}_1) \end{bmatrix} \boldsymbol{\alpha}^k. \tag{B.15}$$

Next we define the vector d^{k-1}, which is the difference of the error vector components, i.e.,

$$\mathbf{d}^{k-1} = \boldsymbol{\epsilon}^k_{2j} - \boldsymbol{\epsilon}^k_{2j-1}, \tag{B.16}$$

or if we use (B.15)

$$\mathbf{d}^{k-1} = \mathcal{D}\boldsymbol{\alpha}^k, \tag{B.17}$$

where

$$\mathcal{D} = \left[(-\mathbf{I} + \tfrac{1}{2}\mathbf{C}_0'\mathbf{C}_0 - \tfrac{1}{2}\mathbf{C}_1'\mathbf{C}_0) \quad (\mathbf{I} - \tfrac{1}{2}\mathbf{C}_1'\mathbf{C}_1 + \tfrac{1}{2}\mathbf{C}_0'\mathbf{C}_1)\right]. \tag{B.18}$$

If we now substitute $\boldsymbol{\alpha}^k$ from (B.5) into (B.17)

$$\mathbf{d}^{k-1} = \mathcal{D}(\mathbf{L}'\boldsymbol{\alpha}^{k-1} + \mathbf{H}'\mathbf{r}^{k-1}),$$

but it can easily be shown, using (B.18), (B.1), and (B.6), that

$$\mathcal{D}\mathbf{L}' = 0; \tag{B.19}$$

therefore,

$$\mathbf{d}^{k-1} = \mathcal{D}\mathbf{H}'\mathbf{r}^{k-1}. \tag{B.20}$$

It is clear from (B.20) that the vectors $\mathbf{d}^{k-1}$ and $\mathbf{r}^{k-1}$ are related by a square matrix which we denote by $\mathbf{B}$, i.e.,

$$\mathbf{r}^{k-1} = \mathbf{B}\mathbf{d}^{k-1} \tag{B.21}$$

or, from (B.20),

$$\mathbf{d}^{k-1} = \mathcal{D}\mathbf{H}'\mathbf{B}\mathbf{d}^{k-1}$$

or

$$\mathbf{I} = \mathcal{D}\mathbf{H}'\mathbf{B}. \tag{B.22}$$

NOTE: From (B.20) the residual vector $\mathbf{r}^{k-1}$ is seen to be the weighted difference of the interpolation errors.

If we postmultiply (B.22) by $\mathbf{B}^{-1}\mathbf{H}$, then use (B.10) and (B.19), and premultiply the result by $\mathbf{B}$, we have

$$\mathbf{H} = \mathbf{B}\mathcal{D}, \tag{B.23}$$

and we have found $\mathbf{H}$, except for the scaling matrix $\mathbf{B}$. The matrix $\mathbf{B}$ is determined as follows: eliminate $\mathbf{H}$ from (B.11) using (B.23), i.e.,

$$\mathbf{B}\mathcal{D}(\mathbf{B}\mathcal{D})' = \mathbf{I},$$

or

$$\mathbf{B}\mathcal{D}\mathcal{D}'\mathbf{B}' = \mathbf{I}$$

and

$$\mathcal{D}\mathcal{D}' = \mathbf{B}^{-1}(\mathbf{B}')^{-1},$$

and finally,

$$\mathbf{B}'\mathbf{B} = (\mathcal{D}\mathcal{D}')^{-1}. \tag{B.24}$$

The choice of $\mathbf{B}$ is not unique. We choose a symmetric $\mathbf{B}$ and (B.24) becomes

$$\mathbf{B}^2 = (\mathcal{D}\mathcal{D}')^{-1}$$

or

$$\mathbf{B} = ((\mathcal{D}\mathcal{D}')^{-1})^{\frac{1}{2}}. \tag{B.25}$$

After some algebra one can show, using (B.18) and (B.6), that

$$\mathcal{D}\mathcal{D}' = 2\mathbf{I} - \frac{1}{2}(\mathbf{C}_0 - \mathbf{C}_1)'(\mathbf{C}_0 - \mathbf{C}_1). \tag{B.26}$$

If we use (B.18) in (B.23) we obtain (2.23a), (2.23b). If we substitute (B.26) in (B.25) we obtain (2.23c).

Appendix C. The vector-matrix equation (2.12) can be written in component form as

(C.1)
$$(\phi_i(x))_j^{k-1} = \sum_{p=0}^{\ell} c^0_{i+1,p+1}(\phi_p(x))^k_{2j-1} + \sum_{p=0}^{\ell} c^1_{i+1,p+1}(\phi_p(x))^k_{2j}, \quad i = 0, 1, \ldots, \ell.$$

Since the orthonormal polynomials $\phi_i(\xi)$ are supported on the interval $(-1 < \xi \le 1]$, it is easier to evaluate the matrix coefficients $c^0_{r,s}$ and $c^1_{r,s}$ by transforming (C.1) to a form where the $\phi_i(\xi)$'s appear explicitly.

To scale x back to ξ, one uses

$$(x^{k-1}_{j-1} < x \le x^{k-1}_j] \longrightarrow (-1 < \xi \le 1],$$

(C.2)
$$(x^k_{2j-2} < x \le x^k_{2j-1}] \longrightarrow (-1 < \xi \le 0], \qquad (x^k_{2j-1} < x \le x^k_{2j}] \longrightarrow (0 < \xi \le 1].$$

The functions map as

$$(\phi_i(x))_j^{k-1} \longrightarrow \phi_i(\xi), \quad -1 < \xi \le 1,$$

$$(\phi_i(x))^k_{2j-1} \longrightarrow \phi_i(2\xi + 1), \qquad (\phi_i(x))^k_{2j} \longrightarrow \phi_i(2\xi - 1), \quad -1 < \xi \le 1. \tag{C.3}$$

Here we have used the compact support of the polynomials, i.e., $\phi_i(\xi) = 0$ for $\xi \notin [-1, 1]$.

Using (C.3), one can rewrite (C.1) as

$$\phi_i(\xi) = \sum_{p=0}^{\ell} c^0_{i+1,p+1}\phi_p(2\xi + 1) + \sum_{p=0}^{\ell} c^1_{i+1,p+1}\phi_p(2\xi - 1). \tag{C.4}$$

Since $\phi_p(2\xi - 1)$ and $\phi_p(2\xi + 1)$ have nonoverlapping support, we can split (C.4) into two equations, i.e., (3.1).

Appendix D. In this appendix we consider the family of orthonormal polynomials (2.8). The family consists of $\ell + 1$ polynomials of degree ℓ. For convenience, we

will choose the ξ-space representation (2.5) so that the polynomials are orthonormal on the interval $-1 < \xi \leq 1$ (see Appendix C).

For $\ell = 0$ there is only one member of the family, i.e., $\phi_0(\xi) = a_{0,0}$, and the coefficient $a_{0,0}$ from the orthonormal condition (A.1) is $a_{0,0} = 1/\sqrt{2}$, i.e.,

$$\phi_0(\xi) = \frac{1}{\sqrt{2}}. \tag{D.1}$$

The elements of the $\mathbf{C}_0$ and $\mathbf{C}_1$ matrices are defined by (3.2a), (3.2b), i.e.,

$$\mathbf{C}_0 = [1], \quad \mathbf{C}_1 = [1], \tag{D.2}$$

which corresponds to the Haar case.

For $\ell > 0$ the families of polynomials have one or more free parameters. For example, if we choose $\ell = 1$ the family has one free parameter, which we denote by b. The two orthonormal polynomials are

$$\phi_0(\xi) = \sqrt{\frac{1}{3}\left(\frac{3}{2} - b^2\right)} + b\,\xi, \tag{D.3a}$$

$$\phi_1(\xi) = -\frac{b}{\sqrt{3}} + \sqrt{\frac{3}{2} - b^2}\,\xi; \quad -\sqrt{\frac{3}{2}} \leq b \leq \sqrt{\frac{3}{2}}, \tag{D.3b}$$

and the corresponding $\mathbf{C}$ matrices are (from (3.2))

$$\mathbf{C}_0 = \begin{bmatrix} 1 - \frac{1}{3}b^2 - b\sqrt{\frac{1}{3}(\frac{3}{2} - b^2)} & \frac{1}{\sqrt{3}}b^2 - \frac{1}{\sqrt{3}}b\sqrt{\frac{1}{3}(\frac{3}{2} - b^2)} \\ -\frac{1}{\sqrt{3}}(\frac{3}{2} - b^2) - \frac{1}{\sqrt{3}}b\sqrt{\frac{1}{3}(\frac{3}{2} - b^2)} & \frac{1}{2} + \frac{1}{3}b^2 + b\sqrt{\frac{1}{3}(\frac{3}{2} - b^2)} \end{bmatrix},$$

$$\mathbf{C}_1 = \begin{bmatrix} 1 - \frac{1}{3}b^2 + b\sqrt{\frac{1}{3}(\frac{3}{2} - b^2)} & -\frac{1}{\sqrt{3}}b^2 - \frac{1}{\sqrt{3}}b\sqrt{\frac{1}{3}(\frac{3}{2} - b^2)} \\ \frac{1}{\sqrt{3}}(\frac{3}{2} - b^2) - \frac{1}{\sqrt{3}}b\sqrt{\frac{1}{3}(\frac{3}{2} - b^2)} & \frac{1}{2} + \frac{1}{3}b^2 - b\sqrt{\frac{1}{3}(\frac{3}{2} - b^2)} \end{bmatrix}. \tag{D.4}$$

If we choose $b = 0$, the polynomials (D.3) reduce to the first two orthonormal Legendre polynomials (3.4) and the corresponding $\mathbf{C}$ matrices are given by (3.8).

The Legendre polynomials produce relatively sparse $\mathbf{C}$ and $\mathbf{D}$ matrices and are used in the examples of sections 3 and 4. The advantages, if any, of other families of polynomials (e.g., $b \neq 0$ for $\ell = 1$) have not been determined.

Appendix E. In this appendix we give the Mathematica code for generating the matrices $\mathbf{C}_0$, $\mathbf{C}_1$ for any ℓ and Legendre polynomials. We also give the code for generating $\mathbf{D}_0$, $\mathbf{D}_1$ for the general case if $\mathbf{C}_0$ and $\mathbf{C}_1$ are given.

```
(*Mathematica Code*)
(*Compute C0 and C1 matrices for Legendre polynomials*)
phi[i_,x_] :=Sqrt[i+1/2] LegendreP[i,x]
c0[i_,j_] :=2 Integrate[phi[j-1,2x+1] phi[i-1,x], {x, -1, 0}]
C0m[l_] :=Array[c0,{l+1,l+1}]
c1[i_,j_] :=2 Integrate[phi[j-1,2x-1] phi[i-1,x], {x, 0, 1}]
C1m[l_] :=Array[c1,{l+1,l+1}]
(*Compute D0 and D1 matrices -Assumes C0 and C1 are input*)
Mat[m0_,m1_] :=(l=Length[m0]-1;
                im=IdentityMatrix[l+1];
                m0t=Transpose[m0];
                m1t=Transpose[m1];
                bs=Inverse[im-1/4 (m0t-m1t) .(m0-m1)];
                {e,m}=Eigensystem[bs];
                mt=Transpose[m];
                de=DiagonalMatrix[e];
                b=(1/Sqrt[2]) mt.Sqrt[de].Inverse[mt] ;
                d0m=Sqrt[2] b.    (-im+(1/2)(m0t.m0-m1t.m0)) ;
                d1m=Sqrt[2] b.    (im-(1/2)(m1t.m1-m0t.m1)) ;
                (*output C0,C1,D0,D1 matrices*)
{m0,m1,d0m,d1m} )
(*EXAMPLE {C0,C1,D0,D1}=Mat[C0m[2],C1m[2]] produces (3.5) and (3.6)*)
```

The Mathematica statement $\{C0, C1, D0, D1\} = Mat[C0m[4], C1m[4]]$ generates the array form of the matrices for the first five ($\ell = 4$) Legendre polynomials (3.3), i.e.,

$$\mathbf{C}_0 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ \frac{-\sqrt{3}}{2} & \frac{1}{2} & 0 & 0 & 0 \\ 0 & \frac{-\sqrt{15}}{4} & \frac{1}{4} & 0 & 0 \\ \frac{\sqrt{7}}{8} & \frac{\sqrt{21}}{8} & \frac{-\sqrt{35}}{8} & \frac{1}{8} & 0 \\ 0 & \frac{\sqrt{3}}{8} & \frac{3\sqrt{5}}{8} & \frac{-3\sqrt{7}}{16} & \frac{1}{16} \end{bmatrix}, \tag{E.1}$$

$$\mathbf{C}_1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ \frac{\sqrt{3}}{2} & \frac{1}{2} & 0 & 0 & 0 \\ 0 & \frac{\sqrt{15}}{4} & \frac{1}{4} & 0 & 0 \\ \frac{-\sqrt{7}}{8} & \frac{\sqrt{21}}{8} & \frac{\sqrt{35}}{8} & \frac{1}{8} & 0 \\ 0 & \frac{-\sqrt{3}}{8} & \frac{3\sqrt{5}}{8} & \frac{3\sqrt{7}}{16} & \frac{1}{16} \end{bmatrix}. \tag{E.2}$$

The exact expressions for the $\mathbf{D}$ matrices are quite complex and we give only the approximate numerical values:

$$\mathbf{D}_0 = \begin{bmatrix} -0.239593 & -0.414988 & -0.288479 & 0.828951 & 0 \\ 0 & -0.0349215 & -0.13525 & -0.120023 & 0.982895 \\ -0.288479 & -0.49966 & -0.608198 & -0.545174 & 0 \\ 0 & -0.120023 & -0.464847 & -0.859942 & -0.173238 \\ 0 & 0 & 0 & 0 & -1 \end{bmatrix}, \tag{E.3}$$

$$\text{(E.4)} \qquad \mathbf{D}_1 = \begin{bmatrix} 0.239593 & -0.414988 & 0.288479 & 0.828951 & 0 \\ 0 & 0.0349215 & -0.13525 & 0.120023 & 0.982895 \\ 0.288479 & -0.49966 & 0.608198 & -0.545174 & 0 \\ 0 & 0.120023 & -0.464847 & 0.859942 & -0.173238 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

For completeness we give the corresponding approximate $\mathbf{B}$ matrix:

$$\text{(E.5)} \qquad \mathbf{B} = \begin{bmatrix} 6.88098 & 0 & -3.26377 & 0 & 0 \\ 0 & 38.9165 & 0 & -5.43161 & 0 \\ -3.26377 & 0 & 2.71069 & 0 & 0 \\ 0 & -5.43161 & 0 & 1.58037 & 0 \\ 0 & 0 & 0 & 0 & 0.707107 \end{bmatrix}.$$

Acknowledgment. We are grateful to Dennis Jespersen for comments and suggestions that improved our manuscript.

REFERENCES

[1] B. K. Alpert, *A class of bases in L^2 for the sparse representation of integral operators*, SIAM J. Math. Anal., 24 (1993), pp. 246–262.

[2] I. Daubechies, *Orthonormal basis of compactly supported wavelets*, Comm. Pure Appl. Math., 41 (1988), pp. 909–996.

[3] G. Donovan, J. Geronimo, and D. Hardin, *Intertwining Multiresolution Analyses and the Construction of Piecewise Polynomial Wavelets*, to appear.

[4] J. Geronimo, D. Hardin, and P. Massopust, *Fractal functions and wavelet expansions based on several scaling functions*, J. Approx. Theory, 78 (1994), pp. 373–401.

[5] A. Harten, *Multiresolution Representation and Numerical Algorithms: A brief review*, ICASE Report No. 94-59, 1994.

[6] G. Strang, *Wavelets and dilation equations: A brief introduction*, SIAM Rev., 31 (1989), pp. 614–627.

[7] G. Strang and V. Strela, *Orthogonal Multiwavelets with Vanishing Moments*, to appear.

[8] R. Warming and R. Beam, *Multiresolution representation using biorthogonal multiwavelets*, SIAM J. Sci. Comput., 22 (2000), pp. 1269–1317.

SIAM J. SCI. COMPUT.
Vol. 22, No. 4, pp. 1269–1317
© 2000 Society for Industrial and Applied Mathematics

DISCRETE MULTIRESOLUTION ANALYSIS USING HERMITE INTERPOLATION: BIORTHOGONAL MULTIWAVELETS*

ROBERT F. WARMING† AND RICHARD M. BEAM†

Abstract. We generalize Harten's interpolatory multiresolution representation to include Hermite interpolation. Compact Hermite interpolation with optimal order accuracy is used in both the decomposition and reconstruction algorithm. The resulting multiple basis functions (biorthogonal multiwavelets) are symmetric or skew-symmetric, compact, and analytic. Harten's approach has several advantages: the multiresolution scheme is inherently discrete, nonperiodic boundary conditions are easy to implement, and the representation can be extended to nonuniform grids in bounded domains. We demonstrate the compression features of the new multiple basis functions by application to several examples.

Key words. wavelets, multiwavelets, multiresolution

AMS subject classifications. 41, 65

PII. S1064827597315236

1. Introduction.

A multiresolution analysis is the decomposition of a function, e.g., $u(x)$, into scales. A *discrete* multiresolution analysis is a decomposition of a vector array $\mathbf{u}$ representing a point value discretization of $u(x)$ on a uniform partition of $[0,1]$. (In practice, the input array may come from a finite difference or finite element approximation to the underlying function $u(x)$.) A discrete multiresolution algorithm consists of a decomposition (analysis) which generates the scale coefficients from the input array and a reconstruction (synthesis) which recovers the input array from the scale coefficients. The reconstruction is conservative in the sense that it reproduces exactly the input array.

An interpolatory multiresolution algorithm due to Harten [Ha93, Ha94, Ha95] can be briefly described as follows. At each step of the decomposition there is a dyadic coarsening of the grid. The grid level index, denoted by the integer k, decreases with grid coarsening. In going from level k to $k-1$, the even-point values are projected and the odd-point values are decimated, i.e., discarded. At level $k-1$ of the reconstruction step, interpolation is used to predict the missing odd-point values at the finer level k. Of course the interpolated values are not, in general, exact since there is an interpolation error. However, in each decomposition step the interpolation errors are computed and saved before the odd-point values are decimated. Consequently, in each step of the reconstruction algorithm the interpolation errors are added to the interpolated values to obtain exact reconstruction. In an interpolatory multiresolution analysis, the interpolation errors at each level k are the scale coefficients. It should be noted that the input array for an interpolatory multiresolution algorithm may consist of cell-averaged values rather than point values. In this case the even-point values are not simply projected from level k to $k-1$ [Ha95].

Harten's interpolatory multiresolution analysis has a beautiful simplicity. The multiresolution algorithm for a uniform grid and periodic boundary conditions can

*Received by the editors January 20, 1997; accepted for publication (in revised form) May 18, 1998; published electronically October 25, 2000. The main results of this paper were presented at the SIAM Annual Meeting, Charlotte, North Carolina, 1995 and in NASA TM 110434, 1997.
http://www.siam.org/journals/sisc/22-4/31523.html
†Advanced Computational Methods Branch, NASA Ames Research Center, Moffett Field, CA 94035-1000.

easily be extended to nonuniform grids and nonperiodic boundary conditions since the requisite interpolation formula does not depend on uniform spacing and can be modified at or near a boundary. This is in contrast to the classical theory of discrete wavelets where it is not possible to expand a function by translates and dilates of a single function on a nonuniform grid and nonperiodic boundary conditions are difficult to implement.

The scalar version of Harten's algorithm is closely related to the work of Donoho [Do92], Sweldens [Sw95], and Sweldens and Schroder [Sw96]. Donoho developed interpolating wavelet transforms where the coefficients are obtained by linear combinations of samples rather than integration. The resulting coefficients have decay properties comparable to the decay properties of smooth orthogonal wavelet decompositions. Sweldens introduced the terminology *second generation wavelets* to describe wavelets that are not necessarily translates and dilates of a single scalar function. Harten's interpolatory wavelets and their generalization to multiwavelets have this property when applied on nonuniform grids and/or finite domains. Sweldens [Sw95] describes a lifting scheme for constructing biorthogonal wavelets by generalization of a basic interpolatory multiresolution algorithm. The lifting scheme imposes an additional global property such as the conservation of some scalar quantity; however, the support of the wavelet is increased while the order of accuracy remains the same as the basic algorithm. Cai and Wang [Ca96] have developed a spline wavelet-like decomposition for a Sobolev space. The resulting wavelet expansion interpolates function values by using cubic splines.

In an interpolatory multiresolution algorithm the interpolation process plays a crucial role. For the scalar algorithm on a uniform grid, the interpolation process is equivalent to a scheme proposed by Dubuc [Du86] with extensions by Deslauriers and Dubuc [De89]. One assumes that discrete function values are given on a uniform partition of an interval. The question is how to extend these discrete values smoothly by interpolation to a nested sequence of uniformly refined dyadic grids. Dubuc defines a method of interpolation generated through a symmetric iterative process on the refined dyadic grids. The Dubuc iterative process is an interpolation scheme, i.e., a *fill-in* scheme, but not a multiresolution scheme.

In this paper we consider *compact Hermite* interpolation not considered by Harten or Dubuc. An interpolation is said to be compact Hermite if it interpolates function values on a subinterval using function values and derivative values at the subinterval end points, i.e., the interpolation points. The advantage of using compact interpolation is that for a multiresolution analysis the support of the basis functions is only two subintervals and the basis functions are piecewise polynomials (nonfractal). In contrast, the cubic interpolation used by Dubuc [Du86] and Harten [Ha93] leads to a basis function which is not analytic and has three times the support of a basis function resulting from compact Hermite interpolation. In Harten's *scalar* interpolatory multiresolution analysis a single variable (scalar) is required at each grid point. In our extension to compact Hermite interpolation, multiple variables are required at each grid point and we use the notation *vector* multiresolution analysis. Compact Hermite interpolation leads to multiple basis functions or, for brevity, *multiwavelets*. In this paper we restrict our attention to one spatial dimension. A two-dimensional multiwavelet version in nonstandard form will be described elsewhere. This work was motivated by a paper on multiwavelets by Strang and Strela [St94]. Their wavelets are orthogonal but piecewise linear and fractal. The basis vectors in this paper are biorthogonal but of arbitrary order and smooth. The biorthogonal Hermite multi-

wavelets developed in this paper can be viewed (for uniform spacing) as a particular class coming from the general theory of oblique multiwavelet transforms developed by Aldroubi [Al97]. For oblique multiwavelets there are no constraints about orthogonality or even biorthogonality.

In general, a wavelet comes from the scaling function by taking "differences" [St89], i.e., the wavelet is a linear combination of translates of the scaling function at the next finer level. In the case of a uniform grid, periodic boundary conditions, compact interpolation, and projection of the even-point values, Harten's interpolatory multiresolution approach leads to a wavelet which is a translate of the scaling function at the next finer level. Consequently, the scaling function and the wavelet have the same functional form, i.e., they are similar functions. This same property extends to multiwavelets where there are several scaling functions and the corresponding wavelets have the same functional form as the scaling functions. The resulting multiwavelets are biorthogonal. In the finite element approach of Strela and Strang [Stre95] each vector multiwavelet is a linear combination of the vector scaling functions translated at the next finer level. This leads to multiwavelets with a larger support (three intervals) as compared with our multiwavelets (two intervals). Strela and Strang achieve semiorthogonality, which is more than biorthogonality and less than orthogonality. The procedure for determining the coefficient matrices of their wavelet equation is rather complicated, and the matrices are not given in analytical form.

This paper is organized as follows. In the next section we define a nested sequence of uniform dyadic grids for a multiresolution analysis. Since our vector multiresolution analysis is a generalization of Harten's scalar analysis, we outline the development of scalar multiresolution analysis in section 3. The definition of accuracy is reviewed in section 4. Compact Hermite interpolation is presented in section 5. In section 6 we generalize Harten's scalar multiresolution analysis to a vector multiresolution analysis with point values and first derivative values specified on a given mesh. A general vector multiresolution algorithm is then developed (section 7) for any combination of function values and one or more derivative values. If only grid-point function values are given in the input array, then the requisite derivative values are computed in a preprocessing step using finite differences (section 8). Accuracy for a vector multiresolution algorithm is defined in section 9. The data compression features of the multiwavelet algorithm are illustrated in section 10. Some concluding remarks are in the final section.

Neither the development nor the application of an interpolatory multiresolution algorithm requires a knowledge of the basis vectors, the scaling functions, or the wavelets. Additionally, implementation of the algorithm does not require explicit knowledge of the dilation (scaling) equation or wavelet equation. However, one needs to select an algorithm with basis functions (which are derived from the scaling functions) appropriate for the particular application, and naturally, one would like to know what scaling functions the multiresolution algorithm generates. Analysis related to the scaling equation, the scaling function, the wavelets, and the discrete basis vectors are relegated to several appendices. In a compact interpolatory multiresolution algorithm the *unit interpolation function* plays the role of the scaling function. In Appendix A we define the unit interpolation function and derive the dilation equation which it satisfies. In Appendix B we examine the discrete basis vectors which can easily be computed numerically from the reconstruction algorithm or analytically from the scaling function. The *fill-in* algorithm, which is an iterative method of interpolating between the specified function values on a discrete grid, is described

in Appendix C. The filled-in function is the subinterval function representation (or subgrid resolution). The fill-in algorithm for vector multiresolution is described in Appendix D. Analytical formulas for the unit interpolation functions for particular cases are derived in Appendix E.

2. Nested grid hierarchy. Let $m > n$ be positive integers, and let $\{X^k\}_{k=m,m-1,\dots,n}$ denote a nested sequence of uniform dyadic grids on the unit interval $[0,1]$ defined by

$$X^k = \{x_j^k\}_{j=0}^{J_k}, \quad x_j^k = jh_k, \quad h_k = 1/2^k, \quad J_k = 2^k, \tag{2.1}$$

where the positive integer k denotes the grid level index. The level index $k = m$ corresponds to the *finest* grid in the hierarchy where the number of subintervals is $J_m = 2^m$ and the subinterval (grid spacing) is $h_m = 2^{-m}$. The level index $k = n$ corresponds to the *coarsest* grid where the number of subintervals is $J_n = 2^n$ and the increment is $h_n = 2^{-n}$. Note that X^{k-1} is formed from X^k by removing grid points of X^k with odd indices:

$$x_j^{k-1} = x_{2j}^k, \quad j = 0, 1, 2, \dots, J_{k-1}. \tag{2.2}$$

A nested sequence of uniform grids is illustrated in Figure 2.1 for $m = 4$ and $n = 0$.

At each level k a grid vector is defined by

$$\mathbf{x}^k = [x_1^k, x_2^k, \dots, 1]^T \qquad x_j^k = j/2^k. \tag{2.3}$$

The grid points x_j^k are sometimes called dyadic points. At each level k a vector $\mathbf{u}^k$ of point values is associated with the grid vector $\mathbf{x}^k$:

$$\mathbf{u}^k = [u_1^k, u_2^k, \dots, u_{J_k}^k]^T, \quad J_k = 2^k. \tag{2.4}$$

The vector $\mathbf{u}^{k-1}$ is half the length of $\mathbf{u}^k$ and is formed by a prescription given in the multiresolution algorithm developed in section 3.

Implicit in this paper is the assumption of a periodic input array with a unit period. For discrete data, periodicity with a unit period is expressed as

$$u_{j+\ell J_k}^k = u_j^k \tag{2.5}$$

for every integer ℓ. Note that the assumption of periodicity implies $u_0^k = u_{J_k}^k$. Other boundary conditions are easy to implement and will be described in a subsequent paper.

At the finest level $k = m$, the vector of point values associated with the grid $\mathbf{x}^m$ is

$$\mathbf{u}^m = [u_1^m, u_2^m, \dots, u_{J_m}^m]^T, \qquad J_m = 2^m. \tag{2.6}$$

Discrete data on the finest grid may come from a continuous periodic function $u(x)$ sampled on the unit interval $[0,1]$, i.e.,

$$u(x) \in C[0,1], \tag{2.7}$$

where $C[0,1]$ is the space of continuous periodic functions on $[0,1]$. The discrete values $u(x_j^m)$ are obtained from a point discretization on the finest grid:

$$u_j^m = u(x_j^m), \quad j = 1, 2, \dots, J_m. \tag{2.8}$$

In the following development of a multiresolution scheme, the vector $\mathbf{u}^m$ can be an arbitrary vector, and there is no assumption that it represents a point discretization of the form (2.8). However, if one considers the issue of accuracy (section 4), then an appropriate assumption on the number of continuous derivatives must be made.

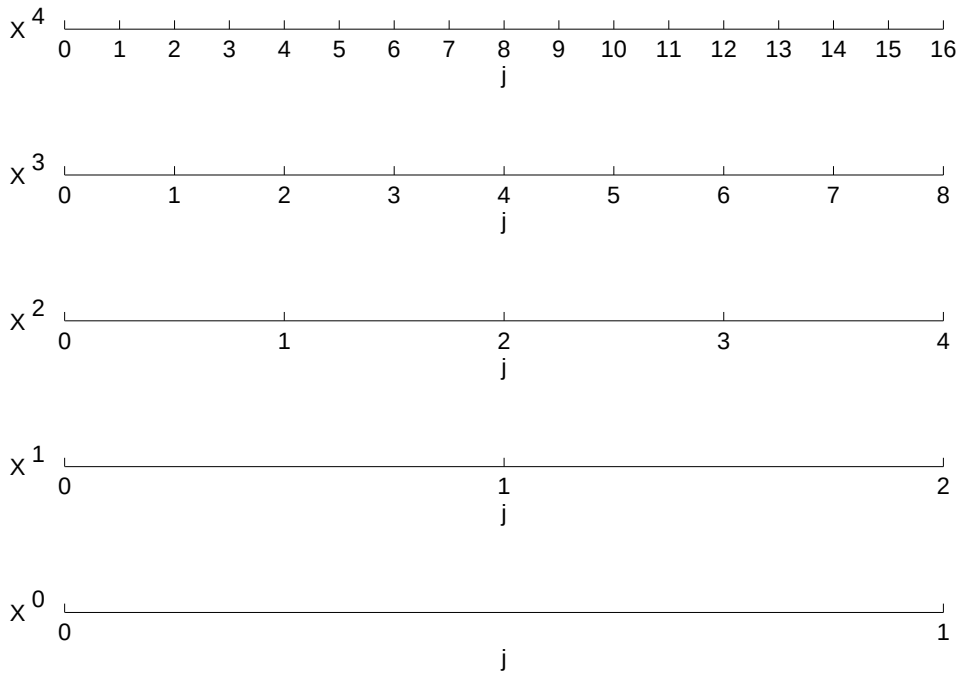


FIG. 2.1. *Nested sequence of grids.*

3. Scalar multiresolution algorithm. In this section we describe a scalar interpolatory multiresolution algorithm due to Harten [Ha93, Ha94, Ha95]. We start at an arbitrary grid level k and go down one level to $k-1$ and then go back to level k. Two generic grids at levels k and $k-1$ are shown pictorially in Figure 3.1. The goal is to achieve scale decomposition and perfect reconstruction of the original data at level k.

3.1. Algorithm development. To go down one level we project the even-point values from level k to $k-1$:

$$u_j^{k-1} = u_{2j}^k, \quad j = 1, 2, \ldots, J_{k-1}. \tag{3.1}$$

(See Figure 3.1 and (2.2).) Note that $\mathbf{u}^{k-1}$ is formed from $\mathbf{u}^k$ by decimation, i.e., discarding values of $\mathbf{u}^k$ with odd indices. Some authors refer to this step as subsampling.

Next, we want to go back up to level k. The even-point values at level k are determined by simply reversing the projection (3.1):

$$u_{2j}^k = u_j^{k-1}, \quad j = 1, 2, \ldots, J_{k-1}. \tag{3.2}$$

But how can one get the odd-point values u_{2j-1}^k at level k from data at level $k-1$? Harten [Ha93, Ha95] used interpolation, and we use linear interpolation as an example. (Interpolation formulas are discussed in detail in section 5.) Let $x_{j-1/2}^{k-1}$ be a fictitious

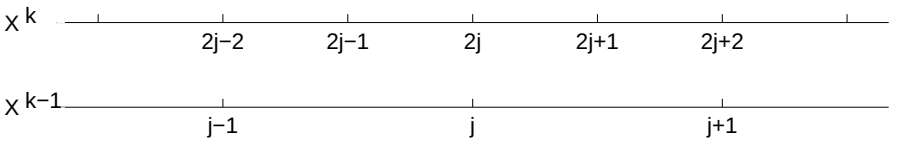


FIG. 3.1. *Two generic grid levels.*

grid point in the middle of the subinterval $[x_{j-1}^{k-1}, x_j^{k-1}]$. The point is called fictitious because it is not a point of the grid X^{k-1}; however, it is clear that

$$x_{j-1/2}^{k-1} = x_{2j-1}^{k}. \tag{3.3}$$

(See Figure 3.1.) By linear interpolation at level $k-1$, one obtains

$$\tilde{u}_{j-1/2}^{k-1} = \frac{1}{2}(u_j^{k-1} + u_{j-1}^{k-1}), \quad j = 1, 2, \ldots, J_{k-1}, \tag{3.4}$$

where $\tilde{u}_{j-1/2}^{k-1}$ is the (linearly) interpolated value at the fictitious point $x_{j-1/2}^{k-1}$. The projection (3.1) and (3.3) are used to rewrite (3.4) as

$$\tilde{u}_{2j-1}^{k} = \frac{1}{2}(u_{2j}^{k} + u_{2j-2}^{k}), \quad j = 1, 2, \ldots, J_{k-1}, \tag{3.5}$$

where $\tilde{u}_{2j-1}^{k}$ denotes a linearly interpolated value of u_{2j-1}^{k}. However, in general, (3.5) is not exact and will not have perfect reconstruction.

An obvious way to fix this deficiency is the following. While we are still at level k (in the decomposition), we have the exact odd-point values u_{2j-1}^{k}. Before we decimate the odd-point values at level k we compute interpolation *errors* or *residuals* at the odd grid points of X^k and denote the result by r_j^{k-1}:

$$r_j^{k-1} = u_{2j-1}^{k} - \tilde{u}_{2j-1}^{k}, \quad j = 1, 2, \ldots, J_{k-1}. \tag{3.6}$$

For linear interpolation $\tilde{u}_{2j-1}^{k}$ is given by (3.5). If we save the r_j^{k-1}'s, then when we go up from level $k-1$ to k (in the reconstruction), we can compute the exact value at the odd points by

$$u_{2j-1}^{k} = \tilde{u}_{2j-1}^{k} + r_j^{k-1} \quad j = 1, 2, \ldots, J_{k-1}. \tag{3.7}$$

The superscript $k-1$ of r_j^{k-1} indicates that the interpolation error or residual is stored at level $k-1$. It is used to reconstruct the exact values at the odd-point values when going from level $k-1$ to level k (see the next algorithm given below). The r_j^k's are sometimes referred to as the scale coefficients at the kth level of resolution.

Before writing down the multiresolution algorithm we introduce notation for the interpolation function. For polynomial interpolation of arbitrary order, the function $I^k(x_{2j-1}^k; u_{2j}^k)$ interpolates odd-point values u_{2j-1}^k (at odd grid points x_{2j-1}^k) using even-point values u_{2j}^k. For linear interpolation (3.5) we write

$$\tilde{u}_{2j-1}^{k} = I^k(x_{2j-1}^k; u_{2j}^k) = \frac{1}{2}(u_{2j}^{k} + u_{2j-2}^{k}), \quad j = 1, 2, \ldots, J_{k-1}. \tag{3.8}$$

As a higher order accurate example, cubic interpolation is given by

$$\tilde{u}^k_{2j-1} = I^k(x^k_{2j-1}; u^k_{2j}) = \frac{9}{16}(u^k_{2j} + u^k_{2j-2}) - \frac{1}{16}(u^k_{2j+2} + u^k_{2j-4}). \tag{3.9}$$

Finally, using (3.1), (3.6) and (3.2), (3.7) together with the interpolation function, e.g., (3.8), one can write an interpolatory multiresolution algorithm in pseudocode:

decomposition

For $k = m, m-1, \ldots, n+1$

For $j = 1, 2, \ldots, J_{k-1}$

$$\begin{aligned} u^{k-1}_j &= u^k_{2j}, \\ r^{k-1}_j &= u^k_{2j-1} - I^k(x^k_{2j-1}; u^k_{2j}), \end{aligned} \tag{3.10a}$$

End

End

reconstruction

For $k = n+1, n+2, \ldots, m$

For $j = 1, 2, \ldots, J_{k-1}$

$$\begin{aligned} u^k_{2j} &= u^{k-1}_j, \\ u^k_{2j-1} &= I^k(x^k_{2j-1}; u^k_{2j}) + r^{k-1}_j, \end{aligned} \tag{3.10b}$$

End

End

Note that the reconstruction step (3.10b) is an obvious reversal or inverse of the decomposition step (3.10a). The number of levels we go down in the decomposition step is $m-n$. The minimum value of the index n is determined by the support of the interpolation formula. For linear interpolation it is $n \geq 1$, and for cubic interpolation it is $n \geq 3$. If for cubic interpolation one picks $n < 3$, then there is a periodic wraparound of the basis function.

In the reconstruction algorithm (3.10b) the interpolation is done at level k. One could also do the interpolation at level $k-1$ as given, for example, by (3.4). The choice is determined by ease of coding or notational convenience as in the vector-matrix form given below. The function $I^{k-1}(x^{k-1}_{j-1/2}; u^{k-1}_j)$ interpolates half-integer-point values $u^{k-1}_{j-1/2}$ (at half-integer grid points $x^{k-1}_{j-1/2}$) using integer point values u^{k-1}_j. For example, linear interpolation (3.4) is denoted by

$$\tilde{u}^{k-1}_{j-1/2} = I^{k-1}(x^{k-1}_{j-1/2}; u^{k-1}_j) = \frac{1}{2}(u^{k-1}_j + u^{k-1}_{j-1}), \quad j = 1, 2, \ldots, J_{k-1}. \tag{3.11}$$

The fact that

$$I^k(x^k_{2j-1}; u^k_{2j}) = I^{k-1}(x^{k-1}_{j-1/2}; u^{k-1}_j)$$

follows directly from the projection (3.1) and the identity (3.3). Consequently, one can write the reconstruction algorithm (3.10b) as

reconstruction

For $k = n+1, n+2, \ldots, m$

For $j = 1, 2, \ldots, J_{k-1}$

$$\begin{aligned} u^k_{2j} &= u^{k-1}_j, \\ u^k_{2j-1} &= I^{k-1}(x^{k-1}_{j-1/2}; u^{k-1}_j) + r^{k-1}_j, \end{aligned} \tag{3.12}$$

End

End

3.2. Biorthogonality and perfect reconstruction. For analysis and exposition purposes we follow the notation of Strang [St89] and rewrite the multiresolution algorithm (3.10a), (3.12) in vector-matrix form as

decomposition

For $k = m, m-1, \ldots, n+1$

$$\begin{aligned} \mathbf{u}^{k-1} &= \mathbf{L}\mathbf{u}^k \\ \mathbf{r}^{k-1} &= \mathbf{H}\mathbf{u}^k \end{aligned} \tag{3.13a}$$

End

reconstruction

For $k = n+1, n+2, \ldots, m$

$$\mathbf{u}^k = \mathbf{L}^s\mathbf{u}^{k-1} + \mathbf{H}^s\mathbf{r}^{k-1} \tag{3.13b}$$

End

The length (dimension) of the column vectors $\mathbf{u}^k$ and $\mathbf{r}^k$ is 2^k (see (2.4)). The dimensions of the rectangular matrices $\mathbf{L}$ and $\mathbf{H}$ are $2^{k-1} \times 2^k$, and the dimensions of the matrices $\mathbf{L}^s$ and $\mathbf{H}^s$ are $2^k \times 2^{k-1}$. For the case of linear interpolation, the form of the matrices $\mathbf{L}$, $\mathbf{H}$, $\mathbf{L}^s$, and $\mathbf{H}^s$ is given at the end of section 7 as a special case of a vector multiresolution algorithm. The vector-matrix form (3.13) is not generally the recommended computational form, but it is useful for analytical exposition. A multiresolution algorithm must possess the perfect reconstruction property, i.e., if one starts at an arbitrary level k and goes down one level to $k-1$ and goes back to level k, then the original k-level data must be reproduced exactly. Using (3.13), one can write the perfect reconstruction property in matrix form as

$$\begin{bmatrix} \mathbf{L} \\ \mathbf{H} \end{bmatrix} \begin{bmatrix} \mathbf{L}^s & \mathbf{H}^s \end{bmatrix} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix}, \tag{3.14}$$

and one obtains the following identities:

$$\mathbf{L}\mathbf{L}^s = \mathbf{I}, \tag{3.15a}$$

$$\mathbf{L}\mathbf{H}^s = \mathbf{0}, \tag{3.15b}$$

$$\mathbf{H}\mathbf{L}^s = \mathbf{0}, \tag{3.15c}$$

$$\mathbf{H}\mathbf{H}^s = \mathbf{I}. \tag{3.15d}$$

Note that (3.14) has the form $\mathbf{A}\mathbf{A}^{-1} = \mathbf{I}$, where $\mathbf{A}$ is a square matrix, and consequently, we can rewrite (3.14) as

$$\begin{bmatrix}\mathbf{L}^s & \mathbf{H}^s\end{bmatrix}\begin{bmatrix}\mathbf{L}\\ \mathbf{H}\end{bmatrix} = \mathbf{I}. \tag{3.16}$$

From this equation we obtain an additional identity

$$\mathbf{L}^s\mathbf{L} + \mathbf{H}^s\mathbf{H} = \mathbf{I}, \tag{3.17}$$

which is not an identity independent from (3.15).

Perfect reconstruction requires that identities (3.15) be satisfied. A discrete multiresolution analysis is said to be *biorthogonal* if identities (3.15) are satisfied. For example, (3.15a) and (3.15b) impose the condition that the ith row of $\mathbf{L}$ is orthogonal to the jth column of $\mathbf{L}^s$ $(i \neq j)$ and all columns of $\mathbf{H}^s$. If $\mathbf{L}^s$ and $\mathbf{H}^s$ are transposes of $\mathbf{L}$ and $\mathbf{H}$, then the multiresolution analysis is said to be *orthogonal* (see Strang [St89]). The interpolatory multiresolution analysis of this paper leads to a biorthogonal multiresolution analysis.

3.3. Pyramid scheme and scale decomposition. In the following discussion it is convenient to write vectors as row vectors. To avoid awkward superscripts such as $(\mathbf{u}^k)^T$, we use an accent prime to indicate a vector transpose:

$$\acute{\mathbf{u}}^k = (\mathbf{u}^k)^T.$$

From (3.13) it is clear that a knowledge of $[\acute{\mathbf{u}}^{k-1}, \acute{\mathbf{r}}^{k-1}]$ is equivalent to a knowledge of $[\acute{\mathbf{u}}^k]$:

$$[\acute{\mathbf{u}}^k] \rightleftharpoons [\acute{\mathbf{u}}^{k-1}, \acute{\mathbf{r}}^{k-1}]. \tag{3.18}$$

Hence one can represent the multiresolution algorithm as a pyramid scheme

$$[\acute{\mathbf{u}}^m] \rightleftharpoons [\acute{\mathbf{u}}^{m-1}, \acute{\mathbf{r}}^{m-1}] \rightleftharpoons [\acute{\mathbf{u}}^{m-2}, \acute{\mathbf{r}}^{m-2}, \acute{\mathbf{r}}^{m-1}] \rightleftharpoons \cdots \rightleftharpoons [\acute{\mathbf{u}}^n, \acute{\mathbf{r}}^n, \ldots, \acute{\mathbf{r}}^k, \ldots, \acute{\mathbf{r}}^{m-2}, \acute{\mathbf{r}}^{m-1}]$$

$$= [\acute{\mathbf{d}}^m]. \tag{3.19}$$

The rightmost vector of (3.19) is defined to be the decomposition vector

$$[\acute{\mathbf{d}}^m] = [\acute{\mathbf{u}}^n, \acute{\mathbf{r}}^n, \ldots, \acute{\mathbf{r}}^k, \ldots, \acute{\mathbf{r}}^{m-2}, \acute{\mathbf{r}}^{m-1}], \tag{3.20}$$

where $\mathbf{d}^m$ is the multiresolution representation of $\mathbf{u}^m$. In (3.19) sequencing left to right $\rightharpoonup$ is the decomposition and sequencing to right to left $\leftharpoondown$ is the reconstruction. It is important to note that every vector $[\,\cdot\,]$ of (3.19) has the same length, namely 2^m. Consequently, the most convenient and compact way to code an interpolatory multiresolution algorithm is to start with an input vector $\mathbf{u}^m$ and overwrite $\mathbf{u}^m$ at every step of the decomposition (or reconstruction) as indicated in (3.19). Note that only the *left* two subvectors of the vector are changed from step to step in this pyramid scheme.

Relation (3.19) shows there is a one-to-one transformation between $\mathbf{u}^m$ and $\mathbf{d}^m$ which we denote by

$$\mathbf{d}^m = \mathbf{W}^{-1}\mathbf{u}^m, \tag{3.21a}$$

$$\mathbf{u}^m = \mathbf{W}\mathbf{d}^m, \tag{3.21b}$$

where $\mathbf{W}$ is a $2^m \times 2^m$ matrix. In component form (3.21b) is

$$\mathbf{u}^m = \sum_{\ell=1}^{2^m} d_\ell^m \mathbf{w}_\ell, \tag{3.22}$$

where the d_ℓ^m's are the elements of $\mathbf{d}^m$. The basis vector $\mathbf{w}_\ell$ denotes the ℓth column of $\mathbf{W}$.

By using the decomposition vector $\mathbf{d}^m$ defined by (3.20), one can rewrite (3.21b) as

$$\mathbf{u}^m = \hat{\mathbf{u}}^{(n)} + \mathbf{u}^{(n)} + \mathbf{u}^{(n+1)} + \cdots + \mathbf{u}^{(k)} + \cdots + \mathbf{u}^{(m-1)}, \tag{3.23a}$$

where

$$\hat{\mathbf{u}}^{(n)} = \sum_{\ell=1}^{2^n} u_\ell^n \mathbf{w}_\ell, \tag{3.23b}$$

$$\mathbf{u}^{(k)} = \sum_{\ell=1}^{2^k} r_\ell^k \mathbf{w}_{2^k+\ell}. \tag{3.23c}$$

(The notation $\mathbf{u}^{(k)}$ should not be confused with $\mathbf{u}^k$.) In (3.23c) the r_ℓ^k's are the elements of the subvector $\mathbf{r}^k$ of the decomposition vector $\mathbf{d}^m$ given by (3.20). For $n \le k \le m-1$ the *detail* $\mathbf{u}^{(k)}$ is a linear combination of 2^k multiresolution basis vectors of scale 2^{-k}. For compact interpolation (section 5), the basis vectors of scale 2^{-k} are *nonoverlapping* (see Appendix B.2.a.). It is apparent that a multiresolution decomposition splits $\mathbf{u}^m$ into components at different *scales* indexed by k. The vector $\hat{\mathbf{u}}^{(n)}$ is a linear combination of *overlapping* multiresolution basis vectors (see (B.2b)). The u_ℓ^n's are the elements of the subvector $\mathbf{u}^n$ of the decomposition vector $\mathbf{d}^m$.

For any multiresolution scheme it is instructive to plot the discrete basis vectors $\mathbf{w}_\ell$ of the expansion (3.22) for particular values of n and m. This can easily be done as described in Appendix B. The basis vectors corresponding to linear interpolation are plotted in Figure B.1 for $m = 3$. In Appendix B.2 we also examine the discrete basis vectors as determined by the dilation and translation of a piecewise polynomial basis function evaluated on a discrete grid.

In a compact interpolatory multiresolution algorithm the *unit interpolation function* plays the role of the scaling function. In Appendix A we define the unit interpolation function and derive the dilation equation which it satisfies for linear interpolation.

An auxiliary question relates to subinterval function representation. Given the vector of discrete function values (2.6) and an interpolating polynomial (not necessarily compact), how does one *fill in* between the discrete function values by an iterative interpolation method? A "filled-in" function is the subinterval function representation (or subgrid resolution) resulting from a particular interpolation formula. The fill-in algorithm for scalar multiresolution analysis is given in Appendix C, and the generalization for a vector multiresolution analysis is given in Appendix D.

4. Accuracy of a scalar multiresolution algorithm. Let the function (2.7) be a polynomial, i.e.,

$$u(x) = x^q, \qquad 0 \le x \le 1, \tag{4.1}$$

where q is a nonnegative integer. On the grid points corresponding to the grid vector

$$\mathbf{x}^m = [x_1, x_2, \ldots, 1]^T, \quad x_j = j/2^m, \tag{4.2}$$

discrete values of the polynomial (4.1) are given by

$$\mathbf{u}^m = [x_1^q, x_2^q, \ldots, 1]^T. \tag{4.3}$$

From decomposition step (3.10a) the interpolation error or residual for $k = m$ is

$$r_j^{m-1} = u_{2j-1}^m - I^m(x_{2j-1}^m; u_{2j}^m), \quad j = 1, 2, \ldots, J_{m-1}, \tag{4.4}$$

where u_{2j}^m is the $2j$th component of (4.3). The number p of vanishing residuals of a multiresolution algorithm is defined by

$$\left[r_j^{m-1} = 0, \quad j = 1, 2, \ldots, J_{m-1}\right], \qquad q = 0, 1, \ldots, p-1 \tag{4.5a}$$

and

$$\left[r_j^{m-1} \neq 0, \quad j = 1, 2, \ldots, J_{m-1}\right], \qquad q = p. \tag{4.5b}$$

The order of accuracy is defined to be p. As an example, for linear interpolation (3.8) the number of vanishing residuals is $p = 2$ and the order of accuracy is 2. For cubic interpolation (3.9) the number of vanishing residuals is $p = 4$ and the order of accuracy is 4.

It is often convenient to define accuracy using vector-matrix notation. Denote the vector (4.3) by

$$[\mathbf{x}]^q = [x_1^q, x_2^q, \ldots, 1]^T. \tag{4.6}$$

The order of accuracy of a multiresolution algorithm is the number of vanishing moments of the error or residual operator $\mathbf{H}$ of (3.13a), i.e., if

$$\mathbf{H}[\mathbf{x}]^q = \mathbf{0}, \quad q = 0, 1, \ldots, p-1, \tag{4.7a}$$

and

$$\mathbf{H}[\mathbf{x}]^p \neq \mathbf{0}, \tag{4.7b}$$

then the order of accuracy is p. For periodic boundary conditions (2.5), the vectors $\mathbf{H}[\mathbf{x}]^q$, $q < p$, will have nonzero elements near the *ends* of the vector arrays since periodicity forces a jump in the function (4.1), except for $q = 0$. After proper adjustment of the scalar interpolation formula near the ends of the unit interval to account for nonperiodic boundary conditions, (4.7) will be satisfied exactly.

For an interpolatory multiresolution algorithm it is easy to analytically check the accuracy since the truncation error of the interpolation formula is the residual in the decomposition step. The truncation error for linear interpolation is given by (5.3) in the next section, and it is obvious that the error is zero if $u(x)$ is a linear function, and consequently, $p = 2$.

5. Compact interpolation formulas. Let $u(x)$ be a continuous periodic function on the interval $[0,1]$, i.e., $u(x) \in C[0,1]$. The discrete mesh or grid X^m defined by (2.1) is a uniform partition of $[0,1]$ by the points $0 = x_0 < x_1 < \cdots < x_j < \cdots < x_{J_m} = 1$. Let $\tilde{u}(x)$ be a piecewise polynomial which coincides with $u(x)$ at the grid points X^m called the interpolation points. The function $\tilde{u}(x)$ is a piecewise polynomial of degree $2\ell-1$ if in each subinterval $[x_{j-1}, x_j]$ it is a polynomial of degree $2\ell-1$.

In this paper a piecewise interpolating polynomial is said to be compact if it interpolates function values on the subinterval $[x_{j-1}, x_j]$ using only point values and (possibly) one or more derivatives of $u(x)$ at the end points x_{j-1} and x_j. If only point values are used, we use the terminology *scalar* compact interpolation. If point values and one or more derivative values are used, we use the terminology *vector* compact interpolation since more than one variable must be assigned to each grid point. Although in the multiresolution algorithms of this paper we assume uniform grid spacing, compact interpolation formulas are applicable for arbitrary spacing $h_j = x_j - x_{j-1}$.

To summarize, in the construction of a vector compact interpolation formula the function value and $(\ell-1)$ derivatives are given at each end point of the subinterval $[x_{j-1}, x_j]$. Consequently, we match 2ℓ conditions, and the interpolating piecewise polynomial has odd degree $2\ell-1$ with $\ell-1$ continuous derivatives.

The simplest example of a compact formula is (scalar) linear interpolation:

$$\tilde{u}_j(x) = a_j(x - x_{j-1}) + b_j, \quad x_{j-1} \le x \le x_j, \quad j = 0, 1, \ldots, J_m - 1, \tag{5.1a}$$

where

$$b_j = u(x_{j-1}), \tag{5.1b}$$

$$a_j = \frac{1}{h}[u(x_j) - u(x_{j-1})], \tag{5.1c}$$

$$h = x_j - x_{j-1}. \tag{5.1d}$$

In this case, $\tilde{u}(x)$ is a polynomial of degree 1 and is a straight line defined by (5.1). Evaluation of (5.1) at the midpoint of the subinterval $x_j - h/2$ yields

$$\tilde{u}_j(x_j - h/2) = \frac{1}{2}[u(x_j) + u(x_{j-1})]. \tag{5.2}$$

If $u(x) \in C^{(2)}[0,1]$, the truncation error of the midpoint interpolation (5.2) is found by a Taylor series expansion:

$$u(x_j - h/2) - \tilde{u}_j(x_j - h/2) = -\frac{1}{8}\left.\frac{d^2u}{dx^2}\right|_{x_j - h/2} h^2 + O(h^3). \tag{5.3}$$

The interpolation formula (5.1) is exact if $u(x)$ is a first degree (linear) polynomial.

An example of a noncompact interpolation formula is Newton's cubic interpolation formula

$$\begin{aligned}\tilde{u}_j(x) &= d_j + c_j(x - x_{j-2}) + b_j(x - x_{j-2})(x - x_{j-1}) \\ &\quad + a_j(x - x_{j-2})(x - x_{j-1})(x - x_j), \qquad x_{j-1} \le x \le x_j.\end{aligned} \tag{5.4}$$

Here function values $u(x_{j-2})$ and $u(x_{j+1})$ are required in addition to the local values $u(x_{j-1})$ and $u(x_j)$ to evaluate the coefficients a_j, b_j, c_j, and d_j. We also note that cubic spline interpolation is not compact since it requires nonlocal data.

Linear interpolation (5.1) is the only compact formula using only function values, i.e., the only *scalar* compact formula. In addition to low order approximation, it has the drawback of discontinuous derivatives at the interpolation points. We can obtain higher order accurate compact formulas and at the same time obtain smoother approximations by using Hermite (vector) interpolation where one matches derivatives as well as function values at the interpolation points. If $u(x) \in C^{(2\ell)}[0,1]$, the truncation error is of the order 2ℓ, i.e., $O(h^{2\ell})$, and the interpolating polynomial converges to $u(x)$ with order 2ℓ as $h \to 0$. Note that Hermite interpolation is exact if $u(x)$ is a polynomial of degree $2\ell - 1$ or less.

As an example, we consider compact Hermite interpolation which matches function values and first derivative values at the ends of the subinterval $[x_{j-1}, x_j]$. In the following, the first derivative of $u(x)$ is denoted by $u'(x)$. Here we assume that $u(x) \in C^{(1)}[0,1]$. Since there are four conditions $u(x_{j-1})$, $u'(x_{j-1})$, $u(x_j)$, and $u'(x_j)$ to match, it is clear that the interpolating polynomial is a cubic on each subinterval $[x_{j-1}, x_j]$:

$$\tilde{u}_j(x) = a_j(x - x_{j-1})^3 + b_j(x - x_{j-1})^2 + c_j(x - x_{j-1}) + d_j, \quad x_{j-1} \le x \le x_j. \tag{5.5}$$

The first derivative is

$$\tilde{u}_j'(x) = 3a_j(x - x_{j-1})^2 + 2b_j(x - x_{j-1}) + c_j, \quad x_{j-1} \le x \le x_j. \tag{5.6}$$

We enforce the conditions that the function $\tilde{u}_j(x)$ and its derivative $\tilde{u}_j'(x)$ agree with the exact values of $u(x)$ and its derivative at the ends of the subinterval $[x_{j-1}, x_j]$:

$$u(x_{j-1}) = d_j, \tag{5.7a}$$
$$u(x_j) = a_j h^3 + b_j h^2 + c_j h + d_j, \tag{5.7b}$$
$$u'(x_{j-1}) = c_j, \tag{5.7c}$$
$$u'(x_j) = 3a_j h^2 + 2b_j h + c_j. \tag{5.7d}$$

By solving the above equations for the coefficients, one obtains

$$d_j = u(x_{j-1}), \tag{5.8a}$$
$$c_j = u'(x_{j-1}), \tag{5.8b}$$
$$b_j = \frac{3}{h^2}[u(x_j) - u(x_{j-1})] - \frac{1}{h}[u'(x_j) + 2u'(x_{j-1})], \tag{5.8c}$$
$$a_j = -\frac{2}{h^3}[u(x_j) - u(x_{j-1})] + \frac{1}{h^2}[u'(x_j) + u'(x_{j-1})]. \tag{5.8d}$$

Evaluation of (5.5) at the midpoint of the subinterval $x_j - h/2$ yields

$$\tilde{u}_j(x_j - h/2) = \frac{1}{2}[u(x_j) + u(x_{j-1})] - \frac{h}{8}[u'(x_j) - u'(x_{j-1})], \tag{5.9a}$$

and evaluation of (5.6) at the midpoint of the subinterval $x_j - h/2$ yields

$$\tilde{u}_j'(x_j - h/2) = \frac{3}{2h}[u(x_j) - u(x_{j-1})] - \frac{1}{4}[u'(x_j) + u'(x_{j-1})]. \tag{5.9b}$$

If $u(x) \in C^{(4)}[0,1]$, the truncation error of the midpoint interpolation (5.9a) is found by Taylor series expansion:

$$u(x_j - h/2) - \tilde{u}_j(x_j - h/2) = \frac{1}{384} \left.\frac{d^4u}{dx^4}\right|_{x_j - h/2} h^4 + O(h^5). \tag{5.10a}$$

Furthermore, if $u(x) \in C^{(5)}[0,1]$, the truncation error of the midpoint interpolation (5.9b) is found by Taylor series expansion:

$$u'(x_j - h/2) - \tilde{u}'_j(x_j - h/2) = \frac{1}{1920} \left.\frac{d^5u}{dx^5}\right|_{x_j - h/2} h^4 + O(h^5). \tag{5.10b}$$

The error terms shown above are consistent with the fact that Hermite interpolation formulas (5.5) and (5.6) are exact if $u(x)$ is a polynomial of degree 3 or less.

In Appendix E of [Wa97] we give two additional compact Hermite interpolation formulas. In particular, we derive compact interpolation formulas which match function values and first and second derivatives at the ends of the subinterval $[x_{j-1}, x_j]$. In addition, we derive compact interpolation formulas which match function values and second derivatives at the ends of the subinterval $[x_{j-1}, x_j]$.

In the multiresolution algorithm one only needs the interpolation formulas (5.9) at the subinterval midpoints (see section 6). However, if one wants to derive analytic formulas for the multiresolution analysis basis functions, then the Hermite interpolating polynomials (5.5) and (5.6) are required. If a compact interpolation formula is used in an interpolatory multiresolution scheme, the resulting multiple basis functions are piecewise polynomials (two pieces) of odd degree $2\ell - 1$ with $\ell - 1$ derivatives. Analytical formulas for the basis functions for several cases are given in Appendix E. (In Appendix E the basis functions are referred to as unit interpolation functions. See also the first paragraph of Appendix A.)

6. Vector multiresolution algorithm. In this section we generalize the scalar multiresolution algorithm of section 3 to vector multiresolution for the special case where a function u and its derivative u' are given on the mesh X^m. In many applications, only point values are given and the requisite derivatives may need to be computed by finite difference formulas. We consider this topic in section 8.

Point values and derivative values at even-numbered grid points are projected from level k to $k-1$:

$$u_j^{k-1} = u_{2j}^k, \quad j = 1, 2, \ldots, J_{k-1} \tag{6.1a}$$

$$u_j^{'k-1} = u'k_{2j}, \quad j = 1, 2, \ldots, J_{k-1}. \tag{6.1b}$$

Recalling the interpolation formulas (5.9) it is convenient to remove the explicit dependence on the spacing h_m by defining $v(x_j^m)$:

$$v(x_j^m) = h_m u'(x_j^m). \tag{6.2}$$

From (2.1) the spacings on two successive levels are related by $h_{k-1} = 2h_k$, and hence one can rewrite (6.1b) as

$$v_j^{k-1} = 2v_{2j}^k, \quad j = 1, 2, \ldots, J_{k-1}. \tag{6.3}$$

On the uniform grid X^{k-1} defined by (2.1), the midpoint interpolation formulas (5.9) are

$$\tilde{u}^{k-1}_{j-1/2} = I^{k-1}_u(x^{k-1}_{j-1/2}; [u^{k-1}_j, v^{k-1}_j]) = \frac{1}{2}(u^{k-1}_j + u^{k-1}_{j-1}) - \frac{1}{8}(v^{k-1}_j - v^{k-1}_{j-1}), \tag{6.4a}$$

$$\tilde{v}^{k-1}_{j-1/2} = I^{k-1}_v(x^{k-1}_{j-1/2}; [u^{k-1}_j, v^{k-1}_j]) = \frac{3}{2}(u^{k-1}_j - u^{k-1}_{j-1}) - \frac{1}{4}(v^{k-1}_j + v^{k-1}_{j-1}). \tag{6.4b}$$

The projections (6.1a) and (6.3) are used to rewrite (6.4) on grid X^k:

$$\tilde{u}^k_{2j-1} = I^k_u(x^k_{2j-1}; [u^k_{2j}, v^k_{2j}]) = \frac{1}{2}(u^k_{2j} + u^k_{2j-2}) - \frac{1}{4}(v^k_{2j} - v^k_{2j-2}), \tag{6.5a}$$

$$\tilde{v}^k_{2j-1} = I^k_v(x^k_{2j-1}; [u^k_{2j}, v^k_{2j}]) = \frac{3}{4}(u^k_{2j} - u^k_{2j-2}) - \frac{1}{4}(v^k_{2j} + v^k_{2j-2}). \tag{6.5b}$$

Here the function $I^k_u(x^k_{2j-1}; [u^k_{2j}, v^k_{2j}])$ interpolates odd-point values u^k_{2j-1} (at odd grid points x^k_{2j-1}) using even-point values $[u^k_{2j}, v^k_{2j}]$. Likewise $I^k_v(x^k_{2j-1}; [u^k_{2j}, v^k_{2j}])$ interpolates odd derivative values v^k_{2j-1} (at odd grid points x^k_{2j-1}) using even-point values $[u^k_{2j}, v^k_{2j}]$. An obvious generalization of the scalar multiresolution algorithm (3.10) is

decomposition

For $k = m, m-1, \ldots, n+1$
  For $j = 1, 2, \ldots, J_{k-1}$

$$u^{k-1}_j = u^k_{2j},$$
$$v^{k-1}_j = 2v^k_{2j},$$

$$(r_u)^{k-1}_j = u^k_{2j-1} - I^k_u(x^k_{2j-1}; [u^k_{2j}, v^k_{2j}]), \tag{6.6a}$$
$$(r_v)^{k-1}_j = v^k_{2j-1} - I^k_v(x^k_{2j-1}; [u^k_{2j}, v^k_{2j}]),$$

  End
End

reconstruction

For $k = n+1, n+2, \ldots, m$
  For $j = 1, 2, \ldots, J_{k-1}$

$$u^k_{2j} = u^{k-1}_j,$$
$$v^k_{2j} = \frac{1}{2}v^{k-1}_j,$$

$$u^k_{2j-1} = I^k_u(x^k_{2j-1}; [u^k_{2j}, v^k_{2j}]) + (r_u)^{k-1}_j, \tag{6.6b}$$
$$v^k_{2j-1} = I^k_v(x^k_{2j-1}; [u^k_{2j}, v^k_{2j}]) + (r_v)^{k-1}_j,$$

  End
End

Alternatively, the reconstruction algorithm (6.6b) can be written as

reconstruction

For $k = n+1, n+2, \ldots, m$
For $j = 1, 2, \ldots, J_{k-1}$

$$u_{2j}^k = u_j^{k-1},$$
$$v_{2j}^k = \frac{1}{2} v_j^{k-1},$$

(6.6c)
$$u_{2j-1}^k = I_u^{k-1}(x_{j-1/2}^{k-1}; [u_j^{k-1}, v_j^{k-1}]) + (r_u)_j^{k-1},$$
$$v_{2j-1}^k = \frac{1}{2} I_v^{k-1}(x_{j-1/2}^{k-1}; [u_j^{k-1}, v_j^{k-1}]) + (r_v)_j^{k-1},$$

End
End

In the generalization of the next section it will be convenient to introduce vector-matrix notation. The vector of point values at the finest level $k = m$ is defined by (2.6):

$$\mathbf{u}^m = [u_1^m, u_2^m, \ldots, u_{J_m}^m]^T, \qquad J_m = 2^m, \tag{6.7a}$$

where

$$u_j^m = u(x_j^m), \quad j = 1, 2, \ldots J_m. \tag{6.7b}$$

The vector of scaled derivatives at the finest level is denoted by

$$\mathbf{v}^m = h_m[u_1'^m, u_2'^m, \ldots, u_{J_m}'^m]^T = [v_1^m, v_2^m, \ldots, v_{J_m}^m]^T, \qquad J_m = 2^m, \tag{6.8a}$$

where

$$v_j^m = h_m \frac{d}{dx} u(x_j^m) = h_m u_j'^m. \tag{6.8b}$$

The input array is represented in matrix form as

$$\boldsymbol{\mathcal{U}}^m = [\mathbf{u}^m, \mathbf{v}^m], \tag{6.9}$$

where $\boldsymbol{\mathcal{U}}^m$ is a $J_m \times 2$ matrix. The vector multiresolution algorithm can be represented as a pyramid scheme analogous to (3.19). The output array at the end of the decomposition is

$$\boldsymbol{\mathcal{D}}^m = [\mathbf{d}_u^m, \mathbf{d}_v^m]. \tag{6.10}$$

Here $\boldsymbol{\mathcal{D}}^m$ is the multiresolution decomposition of $\boldsymbol{\mathcal{U}}^m$. As in the scalar case, a compact way to code the multiresolution pyramid algorithm is to start with the input matrix $\boldsymbol{\mathcal{U}}^m$ (or $\boldsymbol{\mathcal{D}}^m$) and overwrite $\boldsymbol{\mathcal{U}}^m$ (or $\boldsymbol{\mathcal{D}}^m$) at every step of the decomposition (or reconstruction) to obtain $\boldsymbol{\mathcal{D}}^m$ (or $\boldsymbol{\mathcal{U}}^m$).

In a vector compact interpolatory multiresolution algorithm, the unit interpolation functions play the role of the scaling functions. In Appendix A we define the unit interpolation functions and derive the vector dilation equation which they satisfy.

7. General vector multiresolution algorithm. We now want to generalize the algorithm of the previous section to a vector multiresolution algorithm for any combination of u and one or more derivatives of u. For example, one could have u, u', and u'', or u and u''.

7.1. Notation and pyramid scheme. At the finest level $k = m$ let

$$\boldsymbol{\mathcal{U}}^m = [\mathbf{u}_1^m, \mathbf{u}_2^m, \dots, \mathbf{u}_\iota^m] \tag{7.1}$$

denote a $J_m \times \iota$ matrix. The dimension of each column vector $\mathbf{u}_i^m$ is J_m. For the multiresolution of the previous section $\iota = 2$ and

$$\mathbf{u}_1^m = \mathbf{u}^m \quad \text{and} \quad \mathbf{u}_2^m = \mathbf{v}^m, \tag{7.2}$$

where $\mathbf{u}^m$ and $\mathbf{v}^m$ are defined by (6.7) and (6.8).

The multiresolution decomposition of $\boldsymbol{\mathcal{U}}^m$ is denoted by

$$\boldsymbol{\mathcal{D}}^m = [\mathbf{d}_1^m, \mathbf{d}_2^m, \dots, \mathbf{d}_\iota^m]. \tag{7.3}$$

At level k the matrices

$$\boldsymbol{\mathcal{U}}^k = [\mathbf{u}_1^k, \mathbf{u}_2^k, \dots, \mathbf{u}_\iota^k], \tag{7.4a}$$

$$\boldsymbol{\mathcal{R}}^k = [\mathbf{r}_1^k, \mathbf{r}_2^k, \dots, \mathbf{r}_\iota^k] \tag{7.4b}$$

have dimension $J_k \times \iota$. The matrix $\boldsymbol{\mathcal{R}}^k$ is the array of residual values at the kth level of resolution. For a vector multiresolution algorithm, the analogue of (3.18) is

$$[\,\acute{\boldsymbol{\mathcal{U}}}^k] \rightleftharpoons [\,\acute{\boldsymbol{\mathcal{U}}}^{k-1}, \acute{\boldsymbol{\mathcal{R}}}^{k-1}] \tag{7.5}$$

with an obvious generalization of the pyramid scheme (3.19).

$$[\,\acute{\boldsymbol{\mathcal{U}}}^m] \rightleftharpoons [\,\acute{\boldsymbol{\mathcal{U}}}^{m-1}, \acute{\boldsymbol{\mathcal{R}}}^{m-1}] \rightleftharpoons [\,\acute{\boldsymbol{\mathcal{U}}}^{m-2}, \acute{\boldsymbol{\mathcal{R}}}^{m-2}, \acute{\boldsymbol{\mathcal{R}}}^{m-1}] \rightleftharpoons \cdots$$

$$\rightleftharpoons [\,\acute{\boldsymbol{\mathcal{U}}}^n, \acute{\boldsymbol{\mathcal{R}}}^n, \dots, \acute{\boldsymbol{\mathcal{R}}}^k, \dots, \acute{\boldsymbol{\mathcal{R}}}^{m-2}, \acute{\boldsymbol{\mathcal{R}}}^{m-1}] = [\,\acute{\boldsymbol{\mathcal{D}}}^m]. \tag{7.6}$$

In (7.5) and (7.6) the accent prime denotes a matrix transpose.

The jth rows of $\boldsymbol{\mathcal{U}}^k$ and $\boldsymbol{\mathcal{R}}^k$ are denoted by

$$\boldsymbol{\mathcal{U}}_j^k = [u_{1,j}^k, u_{2,j}^k, \dots, u_{\iota,j}^k], \tag{7.7a}$$

$$\boldsymbol{\mathcal{R}}_j^k = [r_{1,j}^k, r_{2,j}^k, \dots, r_{\iota,j}^k], \tag{7.7b}$$

and hence

$$\boldsymbol{\mathcal{U}}^k = \begin{bmatrix} \boldsymbol{\mathcal{U}}_1^k \\ \boldsymbol{\mathcal{U}}_2^k \\ \cdot \\ \cdot \\ \cdot \\ \boldsymbol{\mathcal{U}}_{J_k-1}^k \\ \boldsymbol{\mathcal{U}}_{J_k}^k \end{bmatrix}, \tag{7.8a}$$

$$\boldsymbol{\mathcal{R}}^k = \begin{bmatrix} \boldsymbol{\mathcal{R}}^k_1 \\ \boldsymbol{\mathcal{R}}^k_2 \\ \cdot \\ \cdot \\ \cdot \\ \boldsymbol{\mathcal{R}}^k_{J_k-1} \\ \boldsymbol{\mathcal{R}}^k_{J_k} \end{bmatrix}. \tag{7.8b}$$

We also need a symbolic notation for the requisite interpolation formulas. A row vector of interpolated values is

$$\boldsymbol{\mathcal{I}}^k[x^k_{2j-1};\boldsymbol{\mathcal{U}}^k_{2j}] = [I^k_1(x^k_{2j-1};\boldsymbol{\mathcal{U}}^k_{2j}), I^k_2(x^k_{2j-1};\boldsymbol{\mathcal{U}}^k_{2j}), \dots, I^k_\iota(x^k_{2j-1};\boldsymbol{\mathcal{U}}^k_{2j})]. \tag{7.9}$$

Here $\boldsymbol{\mathcal{I}}^k[x^k_{2j-1};\boldsymbol{\mathcal{U}}^k_{2j}]$ interpolates odd-point values $\boldsymbol{\mathcal{U}}^k_{2j-1}$ (at odd grid points x^k_{2j-1}) using even-point values $\boldsymbol{\mathcal{U}}^k_{2j}$. For example, from the previous section, (6.5) is written as

$$\boldsymbol{\mathcal{I}}^k[x^k_{2j-1};\boldsymbol{\mathcal{U}}^k_{2j}] = [I^k_1(x^k_{2j-1};\boldsymbol{\mathcal{U}}^k_{2j}), I^k_2(x^k_{2j-1};\boldsymbol{\mathcal{U}}^k_{2j})]$$

$$= \Big[\frac{1}{2}(u^k_{2j} + u^k_{2j-2}) - \frac{1}{4}(v^k_{2j} - v^k_{2j-2}), \frac{3}{4}(u^k_{2j} - u^k_{2j-2}) - \frac{1}{4}(v^k_{2j} + v^k_{2j-2})\Big]. \tag{7.10}$$

Finally, in the projection step we need to allow for the possibility of multiplying (or dividing) each column of $\boldsymbol{\mathcal{U}}^k$ by a different constant value. Let

$$\boldsymbol{\alpha} = [\alpha_1, \alpha_2, \dots, \alpha_\iota] \tag{7.11}$$

and define

$$\boldsymbol{\mathcal{P}}_{*\boldsymbol{\alpha}}(\boldsymbol{\mathcal{U}}^k_{2j}) = [\alpha_1 u^k_{1,2j}, \alpha_2 u^k_{2,2j}, \dots, \alpha_\iota u^k_{\iota,2j}] \tag{7.12a}$$

$$\boldsymbol{\mathcal{P}}_{/\boldsymbol{\alpha}}(\boldsymbol{\mathcal{U}}^{k-1}_j) = [u^{k-1}_{1,j}/\alpha_1, u^{k-1}_{2,j}/\alpha_2, \dots, u^{k-1}_{\iota,j}/\alpha_\iota]. \tag{7.12b}$$

In MATLAB notation

$$\boldsymbol{\mathcal{P}}_{*\boldsymbol{\alpha}}(\boldsymbol{\mathcal{U}}^k_{2j}) = \boldsymbol{\mathcal{U}}^k_{2j} .* \boldsymbol{\alpha}, \qquad \boldsymbol{\mathcal{P}}_{/\boldsymbol{\alpha}}(\boldsymbol{\mathcal{U}}^k_j) = \boldsymbol{\mathcal{U}}^k_j ./ \boldsymbol{\alpha}. \tag{7.13}$$

7.2. Computational algorithm. Using the above vector-matrix notation, the decomposition and reconstruction algorithms are written as

decomposition

For $k = m, m-1, \dots, n+1$
  For $j = 1, 2, \dots, J_{k-1}$

$$\begin{aligned} \boldsymbol{\mathcal{U}}^{k-1}_j &= \boldsymbol{\mathcal{P}}_{*\boldsymbol{\alpha}}(\,\boldsymbol{\mathcal{U}}^k_{2j}), \\ \boldsymbol{\mathcal{R}}^{k-1}_j &= \boldsymbol{\mathcal{U}}^k_{2j-1} - \boldsymbol{\mathcal{I}}^k[x^k_{2j-1};\boldsymbol{\mathcal{U}}^k_{2j}], \end{aligned} \tag{7.14a}$$

  End
End

reconstruction

For $k = n+1, n+2, \ldots, m$
  For $j = 1, 2, \ldots, J_{k-1}$

$$\begin{aligned} \boldsymbol{\mathcal{U}}_{2j}^{k} &= \boldsymbol{\mathcal{P}}_{/\boldsymbol{\alpha}}(\boldsymbol{\mathcal{U}}_{j}^{k-1}), \\ \boldsymbol{\mathcal{U}}_{2j-1}^{k} &= \boldsymbol{\mathcal{I}}^{k}[x_{2j-1}^{k}; \boldsymbol{\mathcal{U}}_{2j}^{k}] + \boldsymbol{\mathcal{R}}_{j}^{k-1}, \end{aligned} \tag{7.14b}$$

  End
End

Alternatively, the reconstruction algorithm can be rewritten as

reconstruction

For $k = n+1, n+2, \ldots, m$
  For $j = 1, 2, \ldots, J_{k-1}$

$$\begin{aligned} \boldsymbol{\mathcal{U}}_{2j}^{k} &= \boldsymbol{\mathcal{P}}_{/\boldsymbol{\alpha}}(\boldsymbol{\mathcal{U}}_{j}^{k-1}), \\ \boldsymbol{\mathcal{U}}_{2j-1}^{k} &= \boldsymbol{\mathcal{P}}_{/\boldsymbol{\alpha}}(\boldsymbol{\mathcal{I}}^{k-1}[x_{j-1/2}^{k-1}; \boldsymbol{\mathcal{U}}_{j}^{k-1}]) + \boldsymbol{\mathcal{R}}_{j}^{k-1}, \end{aligned} \tag{7.14c}$$

  End
End

7.3. Vector-matrix form of algorithm. For analysis and exposition purposes the decomposition and reconstruction algorithms (7.14a) and (7.14c) can also be written in vector-matrix form as

decomposition

For $k = m, m-1, \ldots, n+1$
  For $j = 1, 2, \ldots, J_{k-1}$

$$\begin{aligned} \acute{\boldsymbol{\mathcal{U}}}_{j}^{k-1} &= \mathbf{C}_0\, \acute{\boldsymbol{\mathcal{U}}}_{2j}^{k}, \\ \acute{\boldsymbol{\mathcal{R}}}_{j}^{k-1} &= \mathbf{D}_0\, \acute{\boldsymbol{\mathcal{U}}}_{2j-2}^{k} + \mathbf{D}_1\, \acute{\boldsymbol{\mathcal{U}}}_{2j-1}^{k} + \mathbf{D}_2\, \acute{\boldsymbol{\mathcal{U}}}_{2j}^{k}, \end{aligned} \tag{7.15a}$$

  End
End

reconstruction

For $k = n+1, n+2, \ldots, m$
  For $j = 1, 2, \ldots, J_{k-1}$

$$\begin{aligned} \acute{\boldsymbol{\mathcal{U}}}_{2j}^{k} &= \mathbf{F}_1\, \acute{\boldsymbol{\mathcal{U}}}_{j}^{k-1}, \\ \acute{\boldsymbol{\mathcal{U}}}_{2j-1}^{k} &= \mathbf{F}_2\, \acute{\boldsymbol{\mathcal{U}}}_{j-1}^{k-1} + \mathbf{F}_0\, \acute{\boldsymbol{\mathcal{U}}}_{j}^{k-1} + \mathbf{E}_0\, \acute{\boldsymbol{\mathcal{R}}}_{j}^{k-1}, \end{aligned} \tag{7.15b}$$

  End
End,

where

$$\acute{\boldsymbol{\mathcal{U}}}_j^k = \begin{bmatrix} u_{1,j}^k \\ u_{2,j}^k \\ \vdots \\ u_{\iota,j}^k \end{bmatrix}, \tag{7.16a}$$

$$\acute{\boldsymbol{\mathcal{R}}}_j^k = \begin{bmatrix} r_{1,j}^k \\ r_{2,j}^k \\ \vdots \\ r_{\iota,j}^k \end{bmatrix} \tag{7.16b}$$

are column vectors (see (7.7)). The square matrices $\mathbf{C}_0$, $\mathbf{D}_j$, $\mathbf{F}_j$, and $\mathbf{E}_0$ have dimension $\iota \times \iota$. For an interpolatory multiresolution algorithm, the reconstruction step (7.15b) is an obvious reversal or inverse of the decomposition step, and hence

$$\mathbf{F}_0 = -\mathbf{D}_2\mathbf{C}_0^{-1}, \quad \mathbf{F}_1 = \mathbf{C}_0^{-1}, \quad \mathbf{F}_2 = -\mathbf{D}_0\mathbf{C}_0^{-1}. \tag{7.17}$$

For the special case of section 6 where a function and its derivative are given on the mesh X^m, the matrices are given by

$$\mathbf{C}_0 = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}, \tag{7.18a}$$

$$\mathbf{D}_0 = \begin{bmatrix} -1/2 & -1/4 \\ 3/4 & 1/4 \end{bmatrix}, \tag{7.18b}$$

$$\mathbf{D}_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \tag{7.18c}$$

$$\mathbf{D}_2 = \begin{bmatrix} -1/2 & 1/4 \\ -3/4 & 1/4 \end{bmatrix}, \tag{7.18d}$$

$$\mathbf{F}_1 = \mathbf{C}_0^{-1} = \begin{bmatrix} 1 & 0 \\ 0 & 1/2 \end{bmatrix}, \tag{7.18e}$$

$$\mathbf{E}_0 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \tag{7.18f}$$

$$\mathbf{F}_0 = -\mathbf{D}_2\mathbf{C}_0^{-1} = \begin{bmatrix} 1/2 & -1/8 \\ 3/4 & -1/8 \end{bmatrix}, \tag{7.18g}$$

$$\mathbf{F}_2 = -\mathbf{D}_0\mathbf{C}_0^{-1} = \begin{bmatrix} 1/2 & 1/8 \\ -3/4 & -1/8 \end{bmatrix}. \tag{7.18h}$$

For comparison with the notation for multiwavelets used by Strang and Strela [St94] it is convenient to have the decomposition and reconstruction algorithms in matrix notation:

decomposition

For $k = m, m-1, \ldots, n+1$

$$\begin{aligned} \mathbf{U}^{k-1} &= \mathbf{L}\mathbf{U}^{k}, \\ \mathbf{R}^{k-1} &= \mathbf{H}\mathbf{U}^{k}, \end{aligned} \tag{7.19a}$$

End

reconstruction

For $k = n+1, n+2, \ldots, m$

$$\mathbf{U}^{k} = \mathbf{L}^{s}\mathbf{U}^{k-1} + \mathbf{H}^{s}\mathbf{R}^{k-1} \tag{7.19b}$$

End,

where

$$\mathbf{U}^{k} = \begin{bmatrix} \acute{\boldsymbol{u}}_{1}^{k} \\ \acute{\boldsymbol{u}}_{2}^{k} \\ \vdots \\ \acute{\boldsymbol{u}}_{2^k}^{k} \end{bmatrix}; \qquad \mathbf{R}^{k} = \begin{bmatrix} \acute{\boldsymbol{\mathcal{R}}}_{1}^{k} \\ \acute{\boldsymbol{\mathcal{R}}}_{2}^{k} \\ \vdots \\ \acute{\boldsymbol{\mathcal{R}}}_{2^k}^{k} \end{bmatrix}. \tag{7.20}$$

The length (dimension) of the column vectors $\mathbf{U}^k$ and $\mathbf{R}^k$ is $\iota \times 2^k$. The block rectangular matrices $\mathbf{L}$ and $\mathbf{H}$ have block dimension $2^{k-1} \times 2^k$ with subblock size $\iota \times \iota$, and block matrices $\mathbf{L}^s$ and $\mathbf{H}^s$ have block dimension $2^k \times 2^{k-1}$. To indicate the form of the block matrices we write out the matrices for the special case $k = 3$ with periodic boundary conditions:

$$\mathbf{L} = \begin{bmatrix} 0 & \mathbf{C}_0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \mathbf{C}_0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \mathbf{C}_0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{C}_0 \end{bmatrix}, \tag{7.21a}$$

$$\mathbf{H} = \begin{bmatrix} \mathbf{D}_1 & \mathbf{D}_2 & 0 & 0 & 0 & 0 & 0 & \mathbf{D}_0 \\ 0 & \mathbf{D}_0 & \mathbf{D}_1 & \mathbf{D}_2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \mathbf{D}_0 & \mathbf{D}_1 & \mathbf{D}_2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \mathbf{D}_0 & \mathbf{D}_1 & \mathbf{D}_2 \end{bmatrix}, \tag{7.21b}$$

$$\mathbf{L}^{s} = \begin{bmatrix} \mathbf{F}_0 & 0 & 0 & \mathbf{F}_2 \\ \mathbf{F}_1 & 0 & 0 & 0 \\ \mathbf{F}_2 & \mathbf{F}_0 & 0 & 0 \\ 0 & \mathbf{F}_1 & 0 & 0 \\ 0 & \mathbf{F}_2 & \mathbf{F}_0 & 0 \\ 0 & 0 & \mathbf{F}_1 & 0 \\ 0 & 0 & \mathbf{F}_2 & \mathbf{F}_0 \\ 0 & 0 & 0 & \mathbf{F}_1 \end{bmatrix}, \tag{7.21c}$$

$$\mathbf{H}^s = \begin{bmatrix} \mathbf{E}_0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & \mathbf{E}_0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \mathbf{E}_0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \mathbf{E}_0 \\ 0 & 0 & 0 & 0 \end{bmatrix}. \tag{7.21d}$$

The rectangular matrices satisfy the identities (3.15). For orthogonal multiwavelets, one has $\mathbf{L}^s = \mathbf{L}'$ and $\mathbf{H}^s = \mathbf{H}'$, which is clearly not the case for the biorthogonal multiwavelets of this paper.

For the compact scalar algorithm (linear interpolation) of section 3, the submatrices appearing in the above matrices are scalar entries. The appropriate entry is the first element of each of the matrices (7.18):

$$C_0 = 1, \quad D_0 = -1/2, \quad D_1 = 1, \quad D_2 = -1/2, \tag{7.22a}$$

$$F_0 = 1/2, \quad F_1 = 1, \quad F_2 = 1/2, \quad E_0 = 1. \tag{7.22b}$$

7.4. Vector interpolation formulas and basis functions. To implement a vector multiresolution algorithm for any particular case, one needs the appropriate compact Hermite interpolation formulas. In Appendix F of [Wa97] we consider the case of function values and first and second derivative values and the case of function values and second derivative values.

Analytical formulas for the basis functions corresponding to several vector multiresolution algorithms based on compact Hermite interpolation are given in Appendix E of [Wa97].

The fill-in algorithm for vector multiresolution is described in Appendix D of the present paper. A "filled-in" function and one or more derivatives is the subinterval function representation (or subgrid resolution) resulting from a particular compact Hermite interpolation formula.

8. Point value implementation of a vector multiresolution algorithm. In this section we consider an implementation of a vector multiresolution algorithm if only point values are given. If the point values and first derivative values are both given on the finest grid X^m, then one can apply the algorithm of section 7 directly and form $J_m \times 2$ array

$$\boldsymbol{\mathcal{U}}^m = [\mathbf{u}^m, \mathbf{v}^m] \tag{8.1}$$

defined by (6.9). In the notation of section 7, we rewrite (8.1) as

$$\boldsymbol{\mathcal{U}}^m = [\mathbf{u}_1^m, \mathbf{u}_2^m]. \tag{8.2}$$

8.1. Preprocessing. If only point values are given on the finest grid, i.e.,

$$\mathbf{u}^m = \mathbf{u}_1^m = [u_1^m, u_2^m, \ldots, u_{J_m}^m]^T, \qquad J_m = 2^m, \tag{8.3}$$

we require a preprocessing step. The requisite derivatives at each point on the finest grid are computed using the point values (8.3) and a numerical differentiation formula.

A difference quotient consistent with the accuracy and compactness of the compact Hermite interpolation formulas (5.9) is the Hermite or Padé formula

$$u'(x_{j-1}) + 4u'(x_j) + u'(x_{j+1}) = -\frac{3}{h}[u(x_{j-1}) - u(x_{j+1})]$$

$$+\frac{1}{30}u^{(5)}(\eta)\ h^4, \quad x_{j-1} < \eta < x_{j+1}, \tag{8.4}$$

where $h = x_{j+1} - x_j$ for uniform spacing. In writing (8.4) we assume $u(x) \in C^{(5)}[0,1]$. A Hermite or Padé difference approximation is obtained from (8.4) by dropping the error term and replacing $u(x_j)$ and $u'(x_j)$ by the discrete values u_j and u'_j, i.e.,

$$u'_{j-1} + 4u'_j + u'_{j+1} = -\frac{3}{h}[u_{j-1} - u_{j+1}]. \tag{8.5}$$

Note that (8.5) is consistent with (5.9b) rewritten to coincide with the grid points.

Following this procedure would introduce a derivative value for each point value and double the size of the input array, but it would add no additional information. To retain the size of the input vector array (8.3) we keep point values at even grid points and compute the derivatives numerically at the even grid points by using odd-point values. The Hermite formula (8.4) is modified to obtain

$$u'(x_{j-2}) + 22u'(x_j) + u'(x_{j+2}) = -\frac{12}{h}[u(x_{j-1}) - u(x_{j+1})]$$

$$-\frac{17}{15}u^{(5)}(\eta)\ h^4, \quad x_{j-2} < \eta < x_{j+2}, \tag{8.6}$$

where $h = x_{j+1} - x_j$. It is obvious that derivatives computed from the above formula are exact if $u(x)$ is a quartic polynomial. On the even grid points of the discrete grid X^m, the formula (8.6) becomes the approximation

$$v_{j-2} + 22v_j + v_{j+2} = -24[u_{j-1} - u_{j+1}], \quad j = 2, 4, 6, \ldots, J_m, \tag{8.7}$$

where $v_j = 2hu'_j$. In the definition of v_j the factor $2h$ appears rather than h since the derivatives are only computed on the even grid points and the equivalent spacing is $2h$. The right-hand side of (8.7) is evaluated using odd-point values because if only even points appear on the right-hand side, one cannot recover the odd-point values at the completion of the reconstruction step of the multiresolution algorithm (see below). For periodic boundary conditions, (8.7) leads to a circulant tridiagonal Toeplitz matrix that must be "inverted" to solve for the even-point values v_j. The transformed input array is given by (8.2), where

$$\mathbf{u}_1^m = [u_2^m, u_4^m, \ldots, u_{J_m}^m]^T, \qquad J_m = 2^m \tag{8.8a}$$

$$\mathbf{u}_2^m = [v_2^m, v_4^m, \ldots, v_{J_m}^m]^T, \qquad J_m = 2^m \tag{8.8b}$$

and $v_j = 2h_m u'_j$. The modified input array size is $J_{m-1} \times 2$ and has the same number of elements as the input array (8.3).

8.2. Postprocessing. At the completion of the reconstruction step of the multiresolution algorithm, a postprocessing step is required to recover point values at the odd grid points by rewriting (8.7):

$$[u_{j-1} - u_{j+1}] = -\frac{1}{24}[v_{j-2} + 22v_j + v_{j+2}], \qquad j = 2, 4, 6, \ldots, J_m. \tag{8.9}$$

For periodic boundary conditions, one cannot solve the inverse transformation (8.9) to find the odd-point values because the resulting circulant bidiagonal Toeplitz matrix is singular. This problem always occurs with centered difference approximations and periodic boundary conditions. The singularity can be avoided by the artifice of *saving* the odd-point value u_1 and using it in the postprocessing step. The matrix is then a pure bidiagonal Toeplitz matrix, and the linear system is easily solved by a sweep across the grid.

By choosing an uncentered difference approximation, one avoids the singular matrix problem resulting from periodic boundary conditions. As an example, derivatives at the even grid points (as in the preprocessing step) can be computed from an explicit unsymmetric fourth-order difference approximation:

$$v_j = 2hu'_j = \frac{1}{6}[-u_{j-3} + 6u_{j-2} - 18u_{j-1} + 10u_j + 3u_{j+1}], \qquad j = 2, 4, 6, \ldots, J_m. \tag{8.10}$$

Derivatives computed from this difference approximation are exact if $u(x)$ is a quartic polynomial. The transformed input array is (8.2) with (8.8b) computed using (8.10). To recover the odd-point values from (8.8b), one rewrites (8.10) as

$$u_{j-2} + 18u_j - 3u_{j+2} = -6v_{j+1} + 6u_{j-1} + 10u_{j+1}, \quad j = 1, 3, \ldots, J_m - 1. \tag{8.11}$$

For periodic boundary conditions, (8.11) leads to a nonsingular circulant tridiagonal Toeplitz matrix that must be "inverted" to solve for the odd-point values.

9. Accuracy of a vector multiresolution algorithm. The definition of accuracy for a vector multiresolution algorithm is an obvious generalization of the scalar case of section 4. For example, the input array for point values and derivatives of the polynomial (4.1) of section 4 is

$$\boldsymbol{\mathcal{U}}^m = [\mathbf{u}_1^m, \mathbf{u}_2^m], \tag{9.1a}$$

where

$$\mathbf{u}_1^m = [x_1^q, x_2^q, \ldots, 1]^T, \ h_m = 1/2^m, \tag{9.1b}$$

$$\mathbf{u}_2^m = h_m[qx_1^{q-1}, qx_2^{q-1}, \ldots, q]^T, \ h_m = 1/2^m. \tag{9.1c}$$

If the point value input vector is the polynomial array (9.1b), then clearly the scaled (by h) derivative input vector is (9.1c).

From the decomposition step (7.14a) the residual vector for $k = m$ is

$$\boldsymbol{\mathcal{R}}_j^{m-1} = \boldsymbol{\mathcal{U}}_{2j-1}^m - \boldsymbol{\mathcal{I}}^m(x_{2j-1}^m; \boldsymbol{\mathcal{U}}_{2j}^m), \quad j = 1, \ldots, J_{m-1}, \tag{9.2}$$

where $\boldsymbol{\mathcal{U}}_{2j}^m$ is the $2j$th row of (9.1a). The number p of vanishing residuals of a multiwavelet algorithm is defined by

$$\left[\boldsymbol{\mathcal{R}}_j^{m-1} = \mathbf{0}, \quad j = 1, 2, \ldots, J_{m-1}\right], \qquad q = 0, 1, \ldots, p-1 \tag{9.3a}$$

and

$$\big[\boldsymbol{\mathcal{R}}_j^{m-1} \neq \mathbf{0}, \quad j = 1, 2, \ldots, J_{m-1}\big], \qquad q = p. \tag{9.3b}$$

The order of accuracy is defined to be p. Recall from section 5 that for compact interpolation, the interpolating polynomial has odd degree $2\ell - 1$. Hence the number of vanishing residuals is $p = 2\ell$. As an example, for point values and derivatives using Hermite interpolation (5.9), the number of vanishing residuals is $p = 2\ell = 4$ and the order of accuracy is 4.

For a vector interpolatory multiresolution algorithm one can easily check the accuracy since the truncation errors of the interpolation formulas are the residuals in the decomposition step. The truncation error for Hermite interpolation (5.9) is given by (5.10). It is obvious that the error is zero if $u(x)$ is a cubic polynomial and is not zero if $u(x)$ is a quartic polynomial, and consequently, $p = 4$.

As a consistency check on the programming of the decomposition step (7.14a), one can verify the accuracy numerically by checking for the proper number of vanishing moments. After one decomposition step (7.6) one has

$$[\,\acute{\boldsymbol{\mathcal{U}}}^{m}] \rightharpoonup [\,\acute{\boldsymbol{\mathcal{U}}}^{m-1}, \acute{\boldsymbol{\mathcal{R}}}^{m-1}]. \tag{9.4}$$

The array $\boldsymbol{\mathcal{R}}^{m-1}$ is the *bottom* half of the transpose of array (9.4). The second column of $\boldsymbol{\mathcal{R}}^{m-1}$ has an extra vanishing moment since the Hermite derivative interpolation formula (5.9b) is exact for a quartic polynomial (see (5.10b)). One can also verify the accuracy of the preprocessing step of section 8 by computing (8.8b) numerically using the Padé approximation (8.7) with $u_j^m = x_j^q$ and checking again for the proper number of vanishing moments.

10. Data compression. One of the primary applications of a multiresolution analysis is lossy data compression. In many data analysis problems, most of the residual coefficients of the decomposition are negligible and can be set to zero, and upon reconstruction an accurate approximation of the original data will be retained. First, we consider a scalar multiresolution algorithm.

10.1. Scalar compression. For the residuals r_j^k, we define the following truncation operator for $1 \leq j \leq J_k$ and $n \leq k \leq m-1$:

$$\hat{r}_j^k = \begin{cases} r_j^k & \text{if} \quad |r_j^k| > \epsilon_k, \\ 0 & \text{if} \quad |r_j^k| \leq \epsilon_k, \end{cases} \tag{10.1}$$

where ϵ_k is the cutoff tolerance. The decomposition vector is defined by (3.20):

$$[\acute{\mathbf{d}}^m] = [\acute{\mathbf{u}}^n, \acute{\mathbf{r}}^n, \ldots, \acute{\mathbf{r}}^k, \ldots, \acute{\mathbf{r}}^{m-2}, \acute{\mathbf{r}}^{m-1}]. \tag{10.2}$$

On applying the truncation operator (10.1) to each subvector $\mathbf{r}^k$ of (10.2), one obtains the truncated decomposition vector

$$\hat{\mathbf{d}}^m = \begin{bmatrix} \mathbf{u}^n \\ \hat{\mathbf{r}}^n \\ \vdots \\ \hat{\mathbf{r}}^k \\ \vdots \\ \hat{\mathbf{r}}^{m-2} \\ \hat{\mathbf{r}}^{m-1} \end{bmatrix}, \tag{10.3}$$

where the truncated subvectors $\hat{\mathbf{r}}^k$ are given by (10.1). Note that the subvector $\mathbf{u}^n$ is not truncated. Let the number of nonzero elements of $\hat{\mathbf{d}}^m$ be denoted by T_m. The compression ratio is defined by

$$C_r = \text{compression ratio} = \frac{J_m}{T_m}. \tag{10.4}$$

The cutoff tolerance ϵ_k is scaled according to the level k by the formula [Ha95]

$$\epsilon_k = 2^{(k-m)}\epsilon, \tag{10.5}$$

where ϵ is a constant. This scaling formula results in a smaller cutoff tolerance at the coarser scales as compared to the finer scales. If the compression ratio is specified, then ϵ can be determined so that (10.3) has the desired number of nonzero elements.

In this section we assume the compression ratio is specified. A procedure for truncating the decomposition vector when the cutoff tolerance depends on k is as follows. We first replace each subvector $\mathbf{r}^k$ of the decomposition vector $\mathbf{d}^m$ by $2^{(m-k)}\mathbf{r}^k$ and let the *scaled* decomposition vector be denoted by $\mathbf{d}_s^m$. Now the cutoff tolerance for each subvector $\mathbf{r}_s^k$ will be the same, namely ϵ (see (10.5)). Next we take the absolute value of each entry $d_s^m(j)$ of $\mathbf{d}_s^m$ and sort the array in ascending order (the *top* vector $\mathbf{u}^n$ is excluded during this process). Let

$$j_{max} = [J_m(1 - 1/C_r)] + length(\mathbf{u}^n), \tag{10.6}$$

where $[\,\cdot\,]$ denotes the largest integer function. Then ϵ is given by the element

$$d_s^m(j_{max}) = \epsilon. \tag{10.7}$$

Now we go back to the array $\mathbf{d}_s^m$ and set every element of each subvector $\mathbf{r}_s^k$ to zero if it is below the above tolerance ϵ. Finally, we form the truncated vector $\hat{\mathbf{d}}^m$ by multiplying each subvector $\hat{\mathbf{r}}_s^k$ by $2^{(k-m)}$ to obtain $\hat{\mathbf{r}}_k$. In practical applications the scaling and rescaling by 2^{k-m} is avoided by including a scaling factor in the multiresolution algorithm.

Let $\hat{\mathbf{u}}^m$ denote the reconstructed vector corresponding to truncated decomposition vector $\hat{\mathbf{d}}^m$. Harten [Ha95] has shown that one achieves a given error tolerance in the L_1 norm

$$||\mathbf{u}^m - \hat{\mathbf{u}}^m||_1 = O(\epsilon) \tag{10.8}$$

by truncating the residual vectors $\mathbf{r}^k$ with truncation levels ϵ_k determined by (10.1) and (10.5).

10.2. Vector compression. Next we consider the general multiwavelet case. The multiresolution decomposition of the matrix $\boldsymbol{\mathcal{U}}^m$ is given by (7.6).

$$[\,\acute{\boldsymbol{\mathcal{D}}}^m] = [\,\acute{\boldsymbol{\mathcal{U}}}^n, \acute{\boldsymbol{\mathcal{R}}}^n, \ldots, \acute{\boldsymbol{\mathcal{R}}}^k, \ldots, \acute{\boldsymbol{\mathcal{R}}}^{m-2}, \acute{\boldsymbol{\mathcal{R}}}^{m-1}]. \tag{10.9}$$

The jth row of the residual matrix $\boldsymbol{\mathcal{R}}^k$ is

$$\boldsymbol{\mathcal{R}}_j^k = [r_{1,j}^k, r_{2,j}^k, \ldots, r_{\iota,j}^k]. \tag{10.10}$$

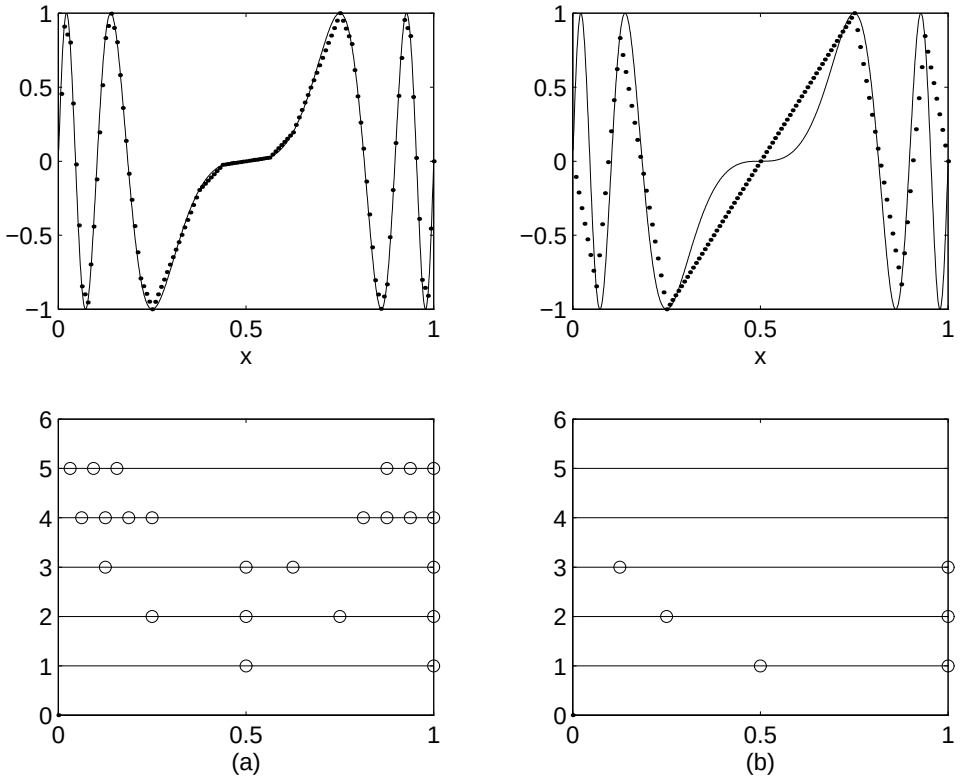


FIG. 10.1. *Compression of a periodic chirp for* $m = 7, n = 1$ *(scalar algorithm). The top figures show reconstructed compressed data, and the bottom figures show nonzero values of compressed decomposition vectors: (a) compression ratio of* $5:1$*, (b) compression ratio of* $16:1$.

For the residual $\boldsymbol{\mathcal{R}}_j^k$ row (10.10) we define the following truncation operator for $1 \le j \le J_k$ and $n \le k \le m-1$:

$$\hat{r}_{i,j}^k = \begin{cases} r_{i,j}^k & \text{if} \quad |r_{i,j}^k| > \epsilon_k \quad \text{for any } i, \qquad 1 \le i \le \iota, \\ 0 & \text{if} \quad |r_{i,j}^k| \le \epsilon_k, \quad \text{for every } i, \quad 1 \le i \le \iota. \end{cases} \tag{10.11}$$

On applying the truncation operator (10.11) to each submatrix $\boldsymbol{\mathcal{R}}^k$, one obtains the truncated decomposition matrix

$$\widehat{\boldsymbol{\mathcal{D}}}^m = \begin{bmatrix} \boldsymbol{\mathcal{U}}^n \\ \widehat{\boldsymbol{\mathcal{R}}}^n \\ \vdots \\ \widehat{\boldsymbol{\mathcal{R}}}^k \\ \vdots \\ \widehat{\boldsymbol{\mathcal{R}}}^{m-2} \\ \widehat{\boldsymbol{\mathcal{R}}}^{m-1} \end{bmatrix}, \tag{10.12}$$

where the truncated subvectors $\widehat{\boldsymbol{\mathcal{R}}}^k$ are given by (10.11).

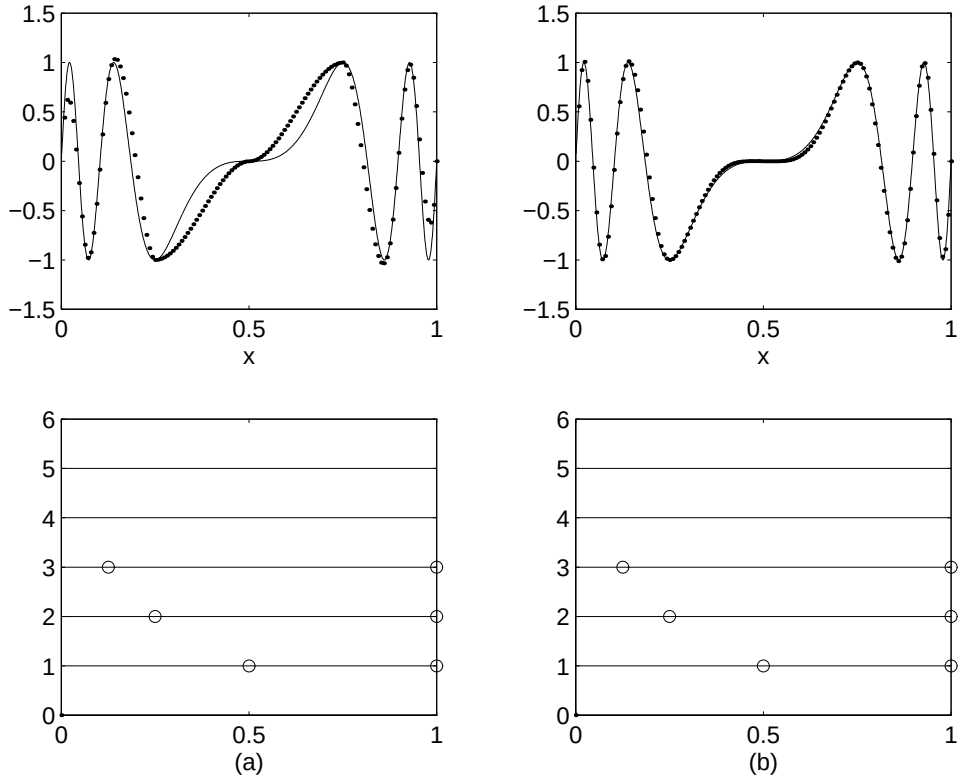


FIG. 10.2. *Compression of a periodic chirp for* $m = 7, n = 1$*, and a compression ratio of* $16 : 1$ *(vector algorithm): (a) function and its derivative are given on the mesh, (b) function and its first and second derivatives are given on the mesh.*

According to the truncation operator (10.11), at a given scale k for fixed j, if any one of the residual components $r_{i,j}^k$ satisfies the inequality $|r_{i,j}^k| > \epsilon$, then $\hat{r}_{i,j}^k = r_{i,j}^k$ for $1 \le i \le \iota$, i.e., the entire row is retained. Let the number of nontruncated rows of decomposition matrix (10.12) be denoted by T_m. The compression ratio is defined by (10.4).

10.3. Computational examples. For our first computational example, we choose a periodic *chirp* with quadratically increasing frequency

$$u(x) = \sin(\alpha\pi(x - 1/2)^3), \quad 0 \le x \le 1, \tag{10.13}$$

where α is a specified parameter. The function (10.13) for $\alpha = 32$ is plotted as a solid curve in the upper part of Figure 10.1(a).

We apply the discrete scalar multiresolution algorithm (3.10) using linear interpolation (3.8) to the input data vector defined by (2.6):

$$\mathbf{u}^m = [u_1^m, u_2^m, \ldots, u_{J_m}^m]^T, \qquad J_m = 2^m. \tag{10.14}$$

We pick $m = 7$, $n = 1$, and a compression ratio of 5:1. Hence the number of subintervals on the finest grid is $J_m = 2^m = 128$. The reconstructed (compressed) discrete data are shown in the upper plot of Figure 10.1(a) by dots. The bottom part of Figure 10.1(a) is a plot in the (space-scale) $x - k$ plane. The open circles plotted in the figure

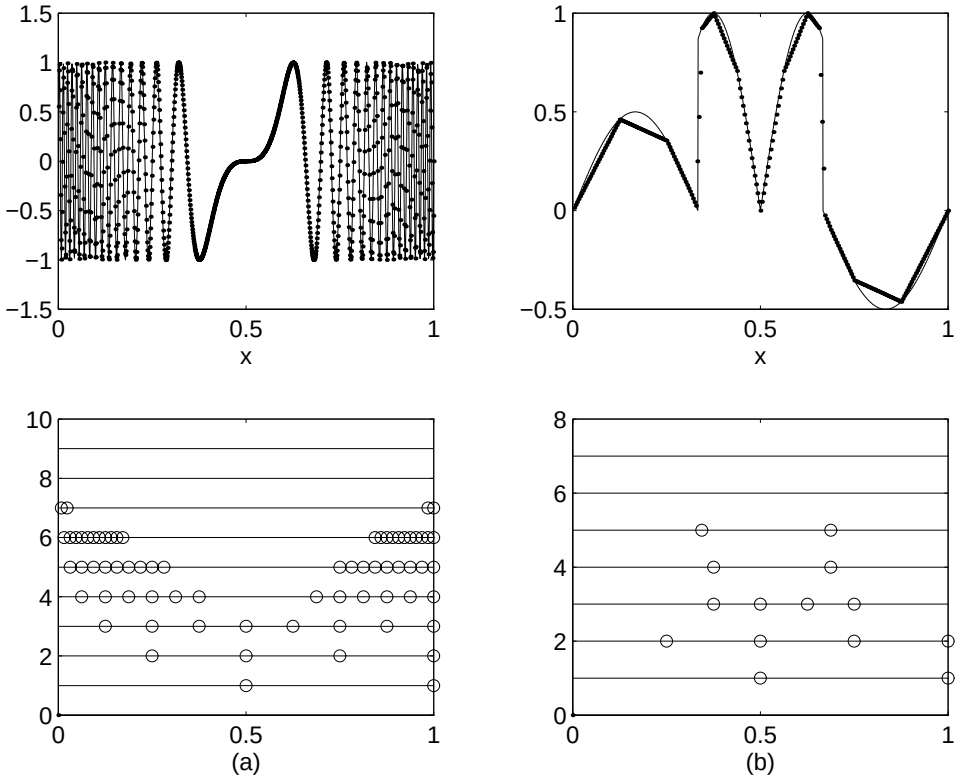


FIG. 10.3. *(a) Compression of a periodic chirp for* $m = 10, n = 1$*, and a compression ratio of* $16:1$ *(vector algorithm). (b) Compression of a function with a jump discontinuity for* $m = 8, n = 1$*, and a compression ratio of* $16:1$ *(scalar algorithm).*

indicate the nonzero values of each truncated subvector $\hat{\mathbf{r}}^k$ plotted as a function of grid X^k defined by (2.1). The scale or level index k is indicated by the integers on the left vertical axis. For example, at level $k = 5$ there are six nonzero residual values in the truncated subvector $\hat{\mathbf{r}}^5$. The open circles form a *spike* at the right end $x = 1$ where the function (10.13) has its steepest gradient. Circles do not appear at the left end $x = 0$ because of the assumption of periodicity.

All of the subsequent figures in this section have the same form as Figure 10.1(a). The upper part of each figure compares the reconstructed compressed data with the exact function, and the lower part of each figure is an $x - k$ plane plot indicating the location of the nonzero values of each truncated subvector $\hat{\mathbf{r}}^k$. For a vector multiresolution algorithm the lower part of each figure indicates the location of the nontruncated rows of each truncated subvector $\widehat{\boldsymbol{\mathcal{R}}}^k$.

We next compare the compression properties of the scalar multiresolution algorithm with two vector multiresolution algorithms using Hermite interpolation. Again we pick $m = 7$, $n = 1$, but with a compression ratio of 16:1. The reconstructed compressed data shown in Figure 10.1(b) for the scalar algorithm (3.10) using linear interpolation are clearly inaccurate for this higher compression ratio. Figure 10.2(a) shows the reconstructed compressed data obtained by applying the vector multiresolution algorithm (7.14) or (6.6) for the case where the function (10.13) and its derivative

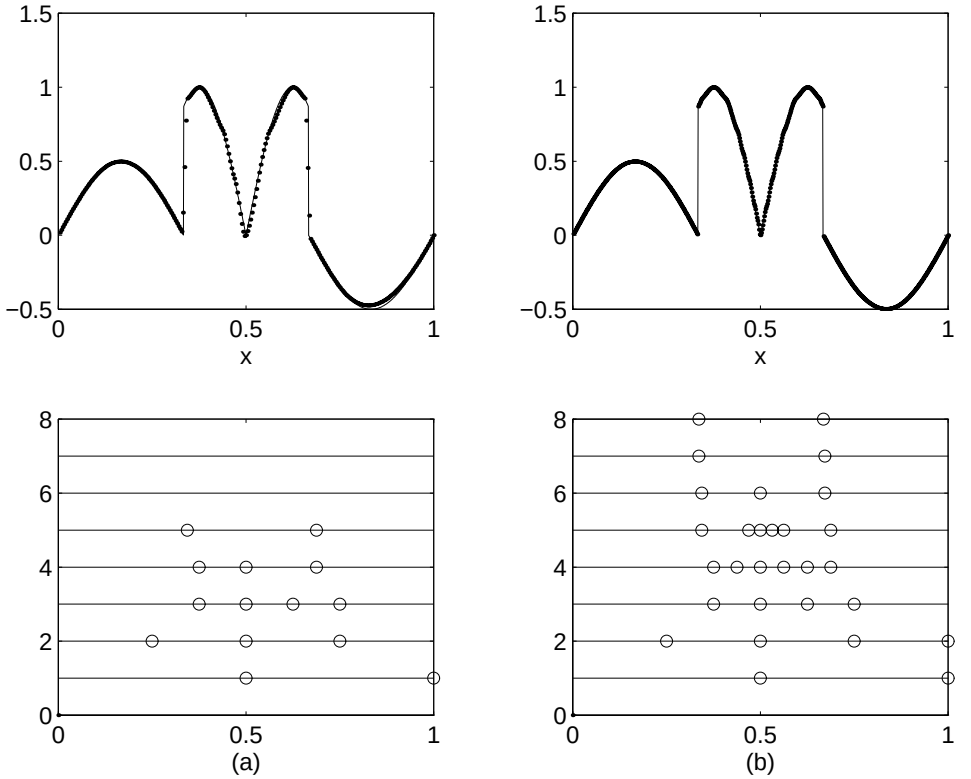


FIG. 10.4. *Compression of a function with a jump discontinuity with a compression ratio of* 16 : 1 *(vector algorithm): (a)* $m = 8$ *and* $n = 1$*, (b)* $m = 9$ *and* $n = 1$*.*

are given on the mesh X^m. In Figure 10.2(b) we plot the reconstructed compressed data obtained by applying the vector multiresolution algorithm (7.14) for the case where the function and its first *and second* derivatives are given on the mesh X^m.

It should be noted that accuracy of the compressed data for any particular case, i.e., the linear case shown in Figure 10.1(b), can be improved by either increasing m or decreasing the compression ratio. Furthermore, for smooth data the number of nonnegligible residual coefficients decreases as the order of the Hermite interpolation formula increases. On comparing Figure 10.1(b) with Figure 10.2(b) the superiority of the vector multiresolution algorithm is apparent. Although not shown in Figure 10.2(b), first and second derivative data are also compressed. On the other hand, three times as much data is retained in the vector algorithm as compared to the scalar algorithm used in Figure 10.1(b). The amount of compressed data retained is comparable in Figure 10.1(a) and Figure 10.2(b).

By increasing the α parameter of the function (10.13) to 256, one obtains the curve shown by the solid curve in Figure 10.3(a). Again we apply the vector multiresolution algorithm for the case where the function and its first and second derivatives are specified. The compression results for $m = 10$ are illustrated in Figure 10.3(a). If one increases the frequency parameter α for a fixed compression ratio, the size of the grid J_m must be increased proportionately to maintain a specified accuracy level.

In our second example, we consider a periodic function with a discontinuous

derivative and two jump discontinuities:

$$u(x) = \begin{cases} \sin(3\pi x)/2 & \text{for} \quad 0 \le x \le 1/3, \\ |\sin(2\pi(2x-1))| & \text{for} \quad 1/3 < x < 2/3, \\ -\sin(3\pi x)/2 & \text{for} \quad 2/3 \le x \le 1. \end{cases} \tag{10.15}$$

This function is plotted as a solid curve in the upper part of Figure 10.3(b). We choose $m = 8$, $n = 1$, and a compression ratio of 16:1. The compression results for the scalar multiresolution algorithm are shown in Figure 10.3(b). In Figure 10.4(a) the compression results for a vector multiresolution algorithm (function and derivative) are shown. On the basis of numerical tests (not shown), we conclude that the accuracy of the Hermite interpolation formulas is severely degraded by the presence of the discontinuities. Higher order interpolation formulas using function values and its first and second derivatives yield reconstructed compressed data that are virtually identical to compression results of Figure 10.4(a).

In Figure 10.4(b) we repeat the computation of Figure 10.4(a) but with $m = 9$, i.e., twice as many points. The reconstructed compressed data now resolve the discontinuous derivative and the two jump discontinuities. At the finest scales there is a *spike* at $x = 1/3$ and $x = 2/3$ in the $x - k$ plane indicating the presence of the discontinuities. At the coarser scales, there is a *spike* at $x = 1/2$ indicating the presence of a discontinuous derivative. This example illustrates the feature recognition properties of a multiresolution analysis.

11. Concluding remarks. A vector interpolatory multiresolution algorithm using Hermite interpolation has been developed. These biorthogonal bases (multiwavelets) have several advantages. The basis functions have the same compactness (two subintervals) regardless of the order of accuracy. For a uniform grid the basis functions are symmetric or skew-symmetric piecewise polynomials (which can easily be derived analytically). Although this paper has been restricted to uniform grids and periodic data, compact interpolation formulas do not depend on uniform spacing, so the multiwavelets can be extended to nonperiodic nonuniform grids on a finite domain. The multiwavelets considered in this paper are well suited for lossy data compression of relatively smooth data and for feature recognition. For data with jump discontinuities, supercompact multiwavelets [Be96] are more suitable.

Appendix A. Dilation equation for a multiresolution analysis. Neither the development nor the application of an interpolatory multiresolution algorithm requires a knowledge of the basis vectors or functions—they are implicitly generated in the multiresolution algorithm. Likewise, implementation of an interpolatory multiresolution algorithm does not require explicit knowledge of the dilation (scaling) equation or wavelet equation. However, in practice one needs to select basis functions (which are derived from the scaling functions) appropriate for the particular application, and naturally, one would like to know what the scaling functions look like. In a compact interpolatory multiresolution algorithm the *unit interpolation function* plays the role of the scaling function. In this appendix we define the unit interpolation function and derive the dilation equation which it satisfies. Analytic formulas for the unit interpolation functions for particular cases are derived in Appendix E. The unit interpolation functions can also be computed numerically as described in Appendix C.

A.1. Unit interpolation function. The multiresolution algorithms of this paper are discrete, i.e., they operate on vectors and return vectors. Given an input

vector, the decomposition algorithm finds the coefficients of the basis vectors described in Appendix B. There is, however, a natural extension of the basis vectors to functions, i.e., a "continuous approximation" to the basis vectors. The extension to basis functions is intimately related to the *unit interpolation function* which we denote by $\phi_i^k(x)$. For a scalar multiresolution analysis the unit interpolation function is defined by the following construction: (1) place a grid X^k on the unit interval, (2) choose the value of $\phi_i^k(x)$ to be unity at x_j^k and zero at all other grid points, and (3) determine the values of $\phi_i^k(x)$ between the grid points from the compact interpolation polynomial. For periodic boundary conditions, $\phi_i^k(x)$ is determined by a single (scaling) function $w(x)$ by dilation and translation.

In the case of Hermite interpolation (where the interpolation polynomial requires more than one value at each grid point), there will be multiple unit interpolation functions, one for each required value. For example, if function values and first derivatives are used, there will be two unit interpolation functions which we denote by ${\phi_1}_i^k(x)$ and ${\phi_2}_i^k(x)$. Examples of unit interpolation functions are given in Appendix E. The unit interpolation functions for compact interpolation are equivalent to the scaling functions for finite element multiwavelets [Stre95].

For compact interpolation the unit interpolation functions have special importance since, as we shall see, they are the scaling functions (continuous approximation to the basis vectors) for the multiresolution analysis. This equivalence between the unit interpolation functions and the scaling functions is not valid for noncompact interpolation (e.g., the cubic interpolation (5.4)). Scaling functions satisfy a dilation equation and generate the wavelets. In the following section we show that for scalar compact interpolation, the unit interpolation function is the scaling function. In section A.4 we show that for compact Hermite interpolation the multiple unit interpolation functions are the multiple scaling functions.

A.2. Scalar dilation equation. In this section we derive the dilation equation from the discrete compact (linear interpolation) multiresolution algorithm (3.10a) and (3.12):

decomposition

$$u_j^{k-1} = u_{2j}^k, \tag{A.1a}$$

$$r_j^{k-1} = u_{2j-1}^k - \frac{1}{2}(u_{2j}^k + u_{2j-2}^k), \tag{A.1b}$$

reconstruction

$$u_{2j}^k = u_j^{k-1}, \tag{A.2a}$$

$$u_{2j-1}^k = \frac{1}{2}(u_j^{k-1} + u_{j-1}^{k-1}) + r_j^{k-1}. \tag{A.2b}$$

We begin by expressing a piecewise linear function $u(x)$ on the unit interval (with periodic boundary conditions) as a series of unit interpolation functions on grid X^{k-1}

$$u(x) = \sum_{i=1,2,\ldots}^{J_{k-1}} u_i^{k-1}\phi_i^{k-1}(x). \tag{A.3}$$

Since $u(x)$ is piecewise linear on grid X^{k-1}, it can be represented exactly as a series of unit interpolation functions on grid X^k, i.e., the residual is zero ($r_j^{k-1} = 0$). Therefore, $u(x)$ on grid X^k is

$$\begin{aligned} u(x) &= \sum_{i=1,2,\ldots}^{J_k} u_i^k \phi_i^k(x) \\ &= \sum_{i=1,3,\ldots}^{J_k-1} u_i^k \phi_i^k(x) + \sum_{i=2,4,\ldots}^{J_k} u_i^k \phi_i^k(x) \\ &= \sum_{i=1,2,\ldots}^{J_{k-1}} u_{2i-1}^k \phi_{2i-1}^k(x) + \sum_{i=1,2,\ldots}^{J_{k-1}} u_{2i}^k \phi_{2i}^k(x). \end{aligned} \tag{A.4}$$

From (A.2b) (with the residual set to zero) we obtain

$$u_{2j-1}^k = \frac{1}{2}(u_j^{k-1} + u_{j-1}^{k-1}). \tag{A.5}$$

Next we use (A.2a) and (A.5) to rewrite (A.4), i.e.,

(A.6)

$$\begin{aligned} u(x) &= \sum_{i=1,2,\ldots}^{J_{k-1}} \frac{1}{2}(u_i^{k-1} + u_{i-1}^{k-1})\phi_{2i-1}^k(x) + \sum_{i=1,2,\ldots}^{J_{k-1}} u_i^{k-1}\phi_{2i}^k(x) \\ &= \sum_{1=1,2,\ldots}^{J_{k-1}} \frac{1}{2}u_i^{k-1}\phi_{2i-1}^k(x) + \sum_{i=0,1,\ldots}^{J_{k-1}-1} \frac{1}{2}u_i^{k-1}\phi_{2i+1}^k(x) + \sum_{i=1,2,\ldots}^{J_{k-1}} u_i^{k-1}\phi_{2i}^k(x) \\ &= \sum_{i=1,2,\ldots}^{J_{k-1}} u_i^{k-1}\left[\frac{1}{2}\phi_{2i-1}^k(x) + \frac{1}{2}\phi_{2i}^k(x) + \phi_{2i+1}^k(x)\right], \end{aligned}$$

where in the last line of (A.6) we have used the assumed periodicity of the u_j^k's. Finally, from a comparison of (A.3) and (A.6) we conclude

$$\phi_i^{k-1}(x) = \frac{1}{2}\phi_{2i-1}^k(x) + \phi_{2i}^k(x) + \frac{1}{2}\phi_{2i+1}^k(x), \tag{A.7}$$

which is the dilation equation for the multiresolution analysis scaling function.

For linear interpolation the unit interpolation function is derived by following the three steps of section A.1 above. For simplicity we pick $k = 1$, the grid on the unit interval is X^1 with $h_1 = 1/2$, and the unit perturbation is at the midpoint $x_1^1 = 1/2$. From Appendix E.1 one finds

$$\phi_1^1(x) = w(x) = 1 - |2x - 1|, \qquad 0 \le x \le 1, \tag{A.8}$$

where $w(x)$ is the hat function plotted in Figure A.1.

For periodic boundary conditions, $\phi_i^k(x)$ is determined from a single scaling function $w(x)$ by dilation and translation:

$$\phi_i^k(x) = w(2^{k-1}x - (i-1)/2). \tag{A.9}$$

To obtain the dilation equation in a more conventional form, we set $k = 2,\ i = 1$ in (A.7) and use (A.9) to obtain

$$w(x) = \frac{1}{2}w(2x) + w(2x - 1/2) + \frac{1}{2}w(2x - 1), \qquad 0 \le x \le 1. \tag{A.10}$$

By construction the solution is the hat function (A.8).

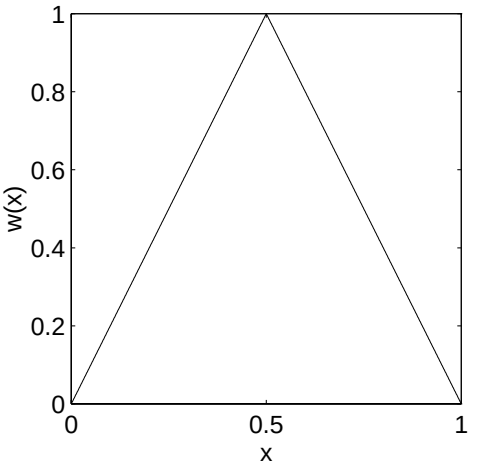


FIG. A.1. *Hat function.*

A.3. Scalar wavelets. In the previous section we assumed that there was no error in the decomposition, i.e., $u(x)$ is represented exactly on both grids X^k and X^{k-1}. If this is not the case and $u(x)$ is represented exactly only on the finer grid, then the reconstruction from the coarser grid to the finer grid results in an error, i.e., instead of (A.4) we write

$$\begin{aligned} u(x) &= \sum_{i=1,2,\ldots}^{J_k} (u_i^k + \epsilon_i^k)\phi_i^k(x) \\ &= \sum_{i=1,2,\ldots}^{J_{k-1}} (u_{2i}^k + \epsilon_{2i}^k)\phi_{2i}^k(x) + \sum_{i=1,2,\ldots}^{J_{k-1}} (u_{2i-1}^k + \epsilon_{2i-1}^k)\phi_{2i-1}^k(x). \end{aligned} \tag{A.11}$$

The error in $u(x)$, denoted by $\epsilon_u^k(x)$, is

$$\epsilon_u^k(x) = \sum_{i=1,2,\ldots}^{J_{k-1}} \epsilon_{2i}^k \phi_{2i}^k(x) + \sum_{i=1,2,\ldots}^{J_{k-1}} \epsilon_{2i-1}^k \phi_{2i-1}^k(x). \tag{A.12}$$

From (A.1a) we see that the even-numbered u's are projected and the error is zero. The error in the odd-numbered u's is r_j^{k-1}, i.e.,

$$\epsilon_u^k(x) = \sum_{i=1,2,\ldots}^{J_{k-1}} r_i^{k-1} \phi_{2i-1}^k(x). \tag{A.13}$$

The functions for the expansion of the error (or detail) are the wavelets, ψ_i^k,

$$\epsilon_u^k(x) = \sum_{i=1,2,\ldots}^{J_{k-1}} r_i^{k-1} \psi_i^{k-1}(x). \tag{A.14}$$

Therefore, for the compact linear algorithm the wavelet is simply the dilated and translated scaling function

$$\psi_i^{k-1}(x) = \phi_{2i-1}^k(x). \tag{A.15}$$

The wavelet is given by inserting (A.9) into (A.15):

$$\psi_i^{k-1}(x) = w(2^{k-1}x - (i-1)). \tag{A.16}$$

A.4. Vector dilation equation. In the derivation of the scalar dilation and wavelet equations we took advantage of the fact that (with the compact unit interpolation functions) a piecewise linear function on grid X^{k-1} can be represented exactly on grid X^k. That is, (A.3) and (A.4) are both valid when $w(x)$ is the unit interpolation function. For higher order (than linear) polynomials $u(x)$, we could try the noncompact cubic interpolation (5.4). We would find, however, that expansions (A.3) and (A.4) with $\phi(x)$ the unit interpolation function are not valid, i.e., they do not produce the same function $u(x)$ and the unit interpolation functions are not the scaling functions.

If we use the higher order compact (vector) interpolation, we can, with slight modification, repeat the analysis of sections A.2 and A.3 and show that the unit interpolation functions (there are more than one in the vector case) are the scaling functions. We demonstrate matching function values and first derivatives for the special case of Hermite interpolation, but the analysis is easily generalized to higher order compact interpolation.

We follow the notation of section 6, i.e., the weighted first derivative is denoted by v (see (6.2)). On the grid X^{k-1} we expand the piecewise cubic polynomial $u(x)$ in a double series of unit interpolation functions

$$u(x) = \sum_{i=1,2,\ldots}^{J_{k-1}} u_i^{k-1}\phi_{1_i}^{k-1}(x) + \sum_{i=1,2,\ldots}^{J_{k-1}} v_i^{k-1}\phi_{2_i}^{k-1}(x), \tag{A.17}$$

where $\phi_{1_i}^{k}(x)$ and $\phi_{2_i}^{k}(x)$ correspond to the unit interpolation functions for point value and for derivative, respectively. On the grid X^k we can write the same piecewise cubic polynomial $u(x)$

$$u(x) = \sum_{i=1,2,\ldots}^{J_k} u_i^{k}\phi_{1_i}^{k}(x) + \sum_{i=1,2,\ldots}^{J_k} v_i^{k}\phi_{2_i}^{k}(x). \tag{A.18}$$

For notational simplicity we will use the vector notation (7.7a)

$$\boldsymbol{\mathcal{U}}_j^k = [u_j^k \; v_j^k] \tag{A.19}$$

and introduce the vector form of the unit interpolation functions, i.e.,

$$\boldsymbol{\phi}_i^k = \begin{bmatrix} \phi_{1_i}^{k}(x) \\ \phi_{2_i}^{k}(x) \end{bmatrix}. \tag{A.20}$$

Note that $\boldsymbol{\mathcal{U}}$ is a row vector and $\boldsymbol{\phi}$ is a column vector. Equations (A.17) and (A.18) become

$$u(x) = \sum_{i=1,2,\ldots}^{J_{k-1}} \boldsymbol{\mathcal{U}}_i^{k-1}\boldsymbol{\phi}_i^{k-1}(x) \tag{A.21}$$

and

$$
\begin{aligned}
u(x) &= \sum_{i=1,2,\ldots}^{J_k} \boldsymbol{\mathcal{U}}_i^k \boldsymbol{\phi}_i^k(x) \\
&= \sum_{i=1,3,\ldots}^{J_k-1} \boldsymbol{\mathcal{U}}_i^k \boldsymbol{\phi}_i^k(x) + \sum_{i=2,4,\ldots}^{J_k} \boldsymbol{\mathcal{U}}_i^k(x) \boldsymbol{\phi}_i^k(x) \qquad \text{(A.22)} \\
&= \sum_{i=1,2,\ldots}^{J_{k-1}} \boldsymbol{\mathcal{U}}_{2i-1}^k \boldsymbol{\phi}_{2i-1}^k(x) + \sum_{i=1,2,\ldots}^{J_{k-1}} \boldsymbol{\mathcal{U}}_{2i}^k \boldsymbol{\phi}_{2i}^k(x).
\end{aligned}
$$

From the reconstruction algorithm (7.15b) (with the residual set to zero)

$$\boldsymbol{\mathcal{U}}_{2i}^k = \boldsymbol{\mathcal{U}}_i^{k-1} \mathbf{F}_1{}', \tag{A.23}$$

$$\boldsymbol{\mathcal{U}}_{2i-1}^k = \boldsymbol{\mathcal{U}}_{i-1}^{k-1} \mathbf{F}_2{}' + \boldsymbol{\mathcal{U}}_i^{k-1} \mathbf{F}_0{}'. \tag{A.24}$$

If we substitute from (A.23) and (A.24) into (A.22), rearrange terms as we did in (A.6), and use the assumed periodicity of the $\boldsymbol{\mathcal{U}}_i^k$'s, then (A.22) can be rewritten as

$$u(x) = \sum_{i=1,2,\ldots}^{J_{k-1}} \boldsymbol{\mathcal{U}}_i^{k-1} [\mathbf{F}_0{}' \boldsymbol{\phi}_{2i-1}^k(x) + \mathbf{F}_1{}' \boldsymbol{\phi}_{2i}^k(x) + \mathbf{F}_2{}' \boldsymbol{\phi}_{2i+1}^k(x)]. \tag{A.25}$$

A comparison of (A.21) and (A.25) leads to the vector dilation equation

$$\boldsymbol{\phi}_i^{k-1} = \mathbf{F}_0{}' \boldsymbol{\phi}_{2i-1}^k(x) + \mathbf{F}_1{}' \boldsymbol{\phi}_{2i}^k(x) + \mathbf{F}_2{}' \boldsymbol{\phi}_{2i+1}^k(x). \tag{A.26}$$

The vector analog of (A.8) is obtained from (A.20):

$$\boldsymbol{\phi}_1^1(x) = \begin{bmatrix} \phi_1{}_1^1(x) \\ \phi_2{}_1^1(x) \end{bmatrix} = \begin{bmatrix} w_1(x) \\ w_2(x) \end{bmatrix}, \tag{A.27}$$

where $w_1(x)$ is the unit interpolation function (E.8) and $w_2(x)$ is the unit interpolation function (E.11) (see Appendix E). The vector unit interpolation function (A.20) is given explicitly by

$$\boldsymbol{\phi}_i^k(x) = \begin{bmatrix} \phi_1{}_i^k(x) \\ \phi_2{}_i^k(x) \end{bmatrix} = \begin{bmatrix} w_1(2^{k-1}x - (i-1)/2) \\ w_2(2^{k-1}x - (i-1)/2) \end{bmatrix}. \tag{A.28}$$

A.5. Vector wavelets. The vector wavelet equation for the compact vector interpolation is completely analogous to the compact scalar wavelet equation. The error equation is the vector form of (A.14), i.e.,

$$\epsilon_u^k(x) = \sum_{i=1,2,\ldots}^{J_{k-1}} \boldsymbol{\mathcal{R}}_i^{k-1} \boldsymbol{\psi}_i^{k-1}(x), \tag{A.29}$$

where the wavelet is the dilated and translated scaling function

$$\boldsymbol{\psi}_i^{k-1}(x) = \boldsymbol{\phi}_{2i-1}^k(x) = \begin{bmatrix} w_1(2^{k-1}x - (i-1)) \\ w_2(2^{k-1}x - (i-1)) \end{bmatrix}. \tag{A.30}$$

Appendix B. Basis vectors for a scalar multiresolution algorithm. For any interpolatory multiresolution scheme it is instructive to plot the discrete basis vectors $\mathbf{w}_\ell$ of the expansion (3.22) for particular values of n and m. The basis vectors can easily be computed numerically from the reconstruction algorithm or analytically from the scaling function for compact interpolation.

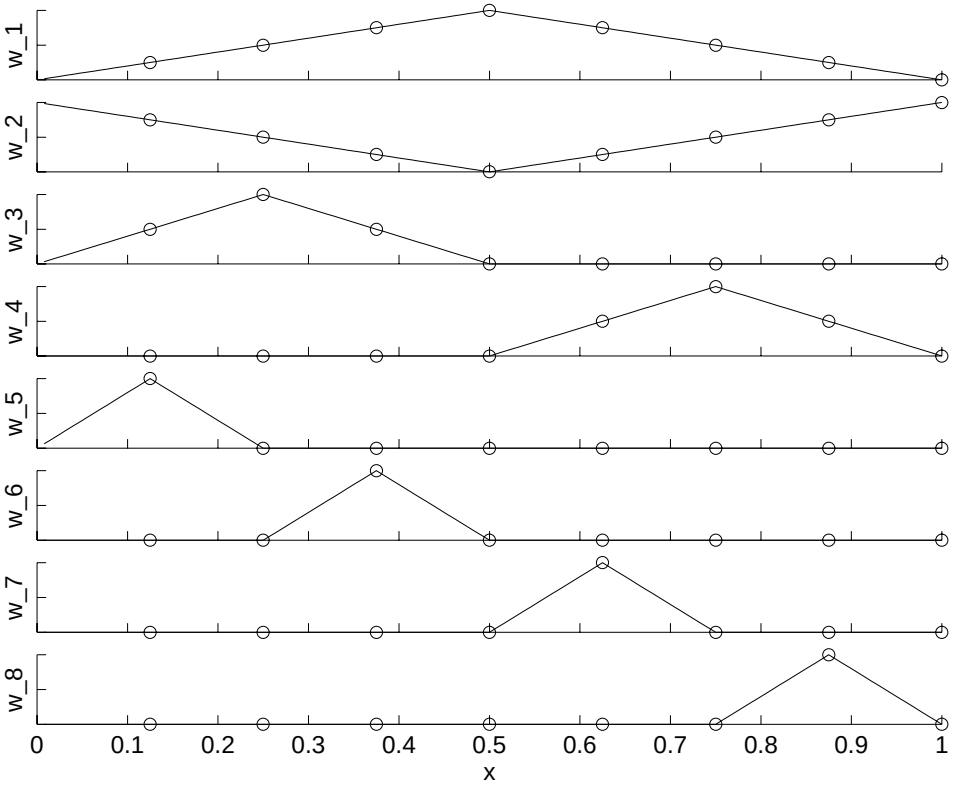


FIG. B.1. *Discrete and continuous basis functions.*

B.1. Numerical method. We first consider a direct numerical calculation. Let ϵ_ℓ^m denote a unit vector, i.e., the ℓth column of the $2^m \times 2^m$ identity matrix. (The ℓth component of ϵ_ℓ^m is unity and all other components are zero.) To numerically compute the discrete basis vector $\mathbf{w}_\ell$, set the decomposition vector $\mathbf{d}^m$ defined by (3.20) to ϵ_ℓ^m in the reconstruction pyramid algorithm (3.19) with the $\leftarrow$ symbol. Recall that the pseudocode for reconstruction is given by (3.10b). The output vector $\mathbf{u}^m$ is $\mathbf{w}_\ell$.

As an example, consider linear interpolation (3.11) on a uniform grid with periodic boundary conditions. The discrete basis vectors $\mathbf{w}_\ell$ are computed by setting $\mathbf{d}^m = \epsilon_\ell^m$ and applying the reconstruction algorithm (3.10b). For $n = 1$ and $m = 3$ the elements of the vectors $\mathbf{w}_\ell$ versus the points of the grid vector (2.3) are plotted as open circles in Figure B.1 for $1 \le \ell \le 8$. The elements of the vector $\mathbf{w}_1$ fall on the hat function plotted in Figure A.1.

An alternative numerical method of computing the basis vectors using the fill-in algorithm is discussed in Appendix C.

B.2. Analytical method. In the following discussion the unit interpolation function (scaling function) is the hat function defined by (A.8). Although $w(x)$ is a continuous function, the discrete basis vector $\mathbf{w}_1$ is obtained by evaluating $w(x)$ on the discrete grid

$$\mathbf{x}^m = [x_1^m, x_2^m, \ldots, 1]^T, \quad \text{where} \quad x_j^m = j/2^m. \tag{B.1}$$

Under dilation and translation, the scaling function $w(x)$ evaluated on the discrete grid (B.1) forms the basis vectors at all scales indicated by (3.23). In the following we assume $w(x) = 0$ for $x \not\subset [0, 1]$, i.e., the basis function $w(x)$ has compact support.

B.2.a. Basis vectors of $\mathbf{u}^{(k)}$. In this section we consider the special case of the vectors $\mathbf{w}_\ell$ in expansion (3.23c) of $\mathbf{u}^{(k)}$. In general, a wavelet comes from the scaling function by taking "differences" [St89], i.e., the wavelet is a linear combination of translates of the scaling function at the next finer level. In the case of a uniform grid, periodic boundary conditions, compact interpolation, and projection of the even-point values, an interpolatory multiresolution approach leads to a wavelet which is a single translate of the scaling function at the next finer level. Consequently, the scaling function and the wavelet have the same functional form (see (A.16)). Hence the family of wavelets is determined from the *single* unit interpolation function (scaling function) by the operations of dilation and translation:

$$w_\ell(x) = w(2^k x - i), \qquad 0 \le x \le 1, \tag{B.2a}$$

where the dilation index k and the translation index i are nonnegative integers satisfying:

$$n \le k \le m - 1, \tag{B.2b}$$

$$0 \le i < 2^k, \tag{B.2c}$$

$$\ell = 2^k + i + 1. \tag{B.2d}$$

If ℓ is specified, the indices k and i are unique. The smallest value of ℓ in (B.2d) is $\ell = 2^n + 1$. For $w(x)$ given by the hat function (A.8), continuous wavelets $w_\ell(x)$ are plotted in Figure B.1 for $3 \le \ell \le 8$. Smaller values of ℓ, which are treated as a special case in section B.2.b, define $w_\ell(x)$ in terms of a family of scaling functions.

The wavelet scale (or width) is given by

$$1/2^k, \quad \text{where} \quad k \quad \text{is the dilation index,} \tag{B.3}$$

and the wavelet location (left end) is

$$i/2^k, \quad \text{where} \quad i \quad \text{is the translation index.} \tag{B.4}$$

For compact interpolation, the support of the family of wavelets (B.2) is the interval

$$I_\ell = [i/2^k, (i + 1)/2^k]. \tag{B.5}$$

For a fixed value of k, the intervals are disjoint if the left end point is omitted.

For $2^n + 1 \le \ell \le 2^m$, the components $w_{\ell j}$, $1 \le j \le 2^m$ of the discrete wavelet vector $\mathbf{w}_\ell$ are found by evaluating (B.2) on the grid $\mathbf{x}^m$:

$$w_{\ell j} = \begin{cases} w(2^k x_j^m - i) & \text{for} \quad \dfrac{i}{2^k} \le x_j^m \le \dfrac{i+1}{2^k}, \\ 0 & \text{otherwise,} \end{cases} \tag{B.6}$$

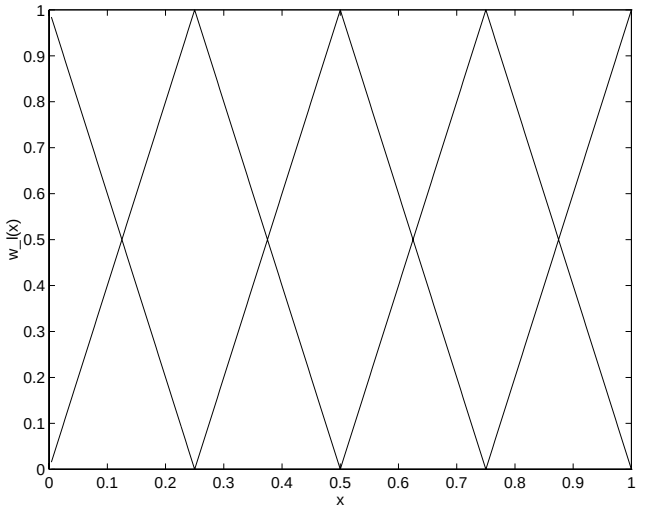


FIG. B.2. *Overlapping scaling functions for* $n = 2$.

where the relations between the indices k, i, ℓ are given by (B.2b), (B.2c), and (B.2d).

B.2.b. Basis vectors of $\hat{\mathbf{u}}^{(n)}$. The smallest value of ℓ in (B.2d) is $\ell = 2^n + 1$. For $1 \le \ell \le 2^n$, the family of scaling functions $w_\ell(x)$ is given by

$$w_\ell(x) = w(2^{n-1}x - i/2), \quad 0 \le x \le 1, \quad i = \ell - 1. \tag{B.7}$$

(See (A.9).) For $w(x)$ given by the hat function (A.8), continuous scaling functions $w_\ell(x)$ are plotted in Figure B.1 for $1 \le \ell \le 2$. The support of the family of scaling functions (B.7) is the interval

$$I_\ell = [i/2^n, (i+2)/2^n]. \tag{B.8}$$

It is clear that the intervals are not disjoint, i.e., the intervals overlap.

For $1 \le \ell \le 2^n$, the components $w_{\ell j}$, $1 \le j \le 2^m$ of the discrete scaling vector $\mathbf{w}_\ell$ are found by evaluating (B.7) on the grid $\mathbf{x}^m$:

$$w_{\ell j} = \begin{cases} w(2^{n-1}x_j^m - i/2) & \text{for} \quad \dfrac{i}{2^n} \le x_j^m \le \dfrac{i+2}{2^n}, \\ 0 & \text{otherwise}, \end{cases} \tag{B.9}$$

where $i = \ell - 1$.

Recall that we are considering the special case of the vectors $\mathbf{w}_\ell$ in the expansion (3.23b) of $\hat{\mathbf{u}}^{(n)}$. This corresponds to the lowest level of a discrete multiresolution algorithm and, as noted above, the scaling functions for an interpolatory multiresolution algorithm overlap at the lowest level $k = n$. For example, if we choose $n = 2$, then the four functions given by (B.7)

$$w_\ell(x) = w(2x - i/2), \quad i = 0, 1, 2, 3, \quad 0 \le x \le 1 \tag{B.10}$$

overlap as illustrated in Figure B.2. In writing (B.10) we have assumed that $w(x)$ is the hat function (A.8) with $w(x) = 0$ for $|x| > 1$ and that $w(2x)$, $0 \le x \le 1$ has been periodically extended to all x.

For the family of basis functions plotted in Figure B.1 the integer n is unity and the first two basis functions are scaling functions:

$$w_1(x) = w(x), \qquad w_2(x) = w(x - 1/2), \qquad 0 \le x \le 1. \tag{B.11}$$

It should be mentioned that the vector multiresolution algorithms of sections 6 and 7 have several piecewise polynomial scaling functions with compact support. Analytical formulas for particular multiple scaling functions are given in Appendix E.

Appendix C. Fill-in algorithm for scalar multiresolution. In the nested sequence of uniform grids (2.1) we assumed that $k = m$ corresponds to the *finest* grid for which a discrete vector $\mathbf{u}^m$ is defined. In this section we assume that $k = n$ corresponds to the *finest* grid for which a discrete vector $\mathbf{u}^n$ is defined, i.e.,

$$\mathbf{x}^n = [x_1^n, x_2^n, \dots, 1]^T, \quad \text{where} \quad x_j^n = j/2^n \tag{C.1}$$

on which an input vector

$$\mathbf{u}^n = [u_1^n, u_2^n, \dots, u_{J_n}^n]^T, \quad J_n = 2^n \tag{C.2}$$

is defined. Hence the nested sequence of uniform grids is $\{X^k\}_{k=n,n+1,\dots}$, where X^k is defined by (2.1). If one starts on the grid X^n and applies the reconstruction algorithm (3.12) with the error or residual term set to zero and with increasing values of k, then one obtains an iterative method of interpolation or *fill-in* between function values (C.2) on the grid X^n. The filled-in function is the subinterval function representation (or subgrid resolution).

For the scalar multiresolution algorithm of section 3 the fill-in algorithm is

fillin

For $k = n + 1, n + 2, \dots, max$
  For $j = 1, 2, \dots, J_{k-1}$

$$\begin{aligned} u_{2j}^k &= u_j^{k-1}, \\ u_{2j-1}^k &= I^{k-1}(x_{j-1/2}^{k-1}; u_j^{k-1}), \end{aligned} \tag{C.3}$$

  End
End

At each level k, additional function values, denoted by $\tilde{u}(x_{2j-1}^k)$, are generated at the dyadic grid points

$$X^k - X^{k-1} = \{x_{2j-1}^k\}_{j=1}^{J_{k-1}}. \tag{C.4}$$

In numerical computations, there is a truncated upper limit to the index k of (C.3), which is denoted by max. The number of grid points in the filled-in function is $2^{\max}$. The fill-in between the point values (C.2) is accomplished by the iterative interpolation process (C.3). This method of interpolation is due to Dubuc [Du86] and has been extended by Deslauriers and Dubuc [De89]. In the $\lim_{max\to\infty}$ the iterative process converges to a uniformly continuous function $\tilde{u}(x)$ [Du86].

The fill-in algorithm (C.3) can be recast in vector-matrix form as

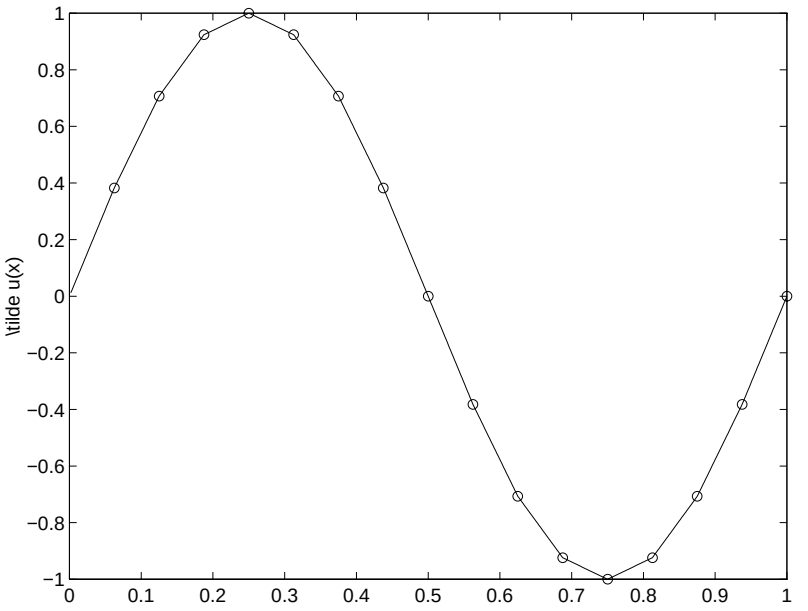


FIG. C.1. *Fill-in function for* $\sin(2\pi x)$.

fillin

For $k = n+1, n+2, \ldots, max$

$$\mathbf{u}^k = \mathbf{L}^s \mathbf{u}^{k-1} \tag{C.5}$$

End

The accuracy of the filled-in function $\tilde{u}(x)$ when compared to the function $u(x)$, whose point values were prescribed on the grid X^n, is determined by the interpolation formula used in the multiresolution algorithm. For example, the scalar algorithm (C.3) using linear interpolation (3.11) yields a piecewise linear function $\tilde{u}(x)$ between the original grid points X^n. The filled-in function for $\sin(2\pi x)$, $0 \le x \le 1$ with $n = 4$ is shown in Figure C.1. The open circles are the exact values of $\sin(2\pi x)$ on the grid (C.1).

The scaling function for a scalar interpolatory multiresolution algorithm, which we denote by $w(x)$, is easily computed numerically from the iterative *fill-in* algorithm. The scaling function is equivalent to the fundamental function defined by Dubuc [Du86]. For *compact* interpolation the support of scaling function is two intervals of the grid X^n. On the unit interval with $n = 1$ we write $[0,1] = [0,1/2] \cup [1/2,1]$. We assume that the scaling function $w(x)$ is zero at both ends of the unit interval $[0,1]$ and there is a unit perturbation at the center point $x = 1/2$:

$$w(0) = w(1) = 0, \qquad w(1/2) = 1. \tag{C.6}$$

Since the scaling function is zero on both ends of the unit interval, it is convenient to assume periodic boundary conditions when applying the fill-in algorithm. For (C.3) with $n = 1$, the *input* array corresponding to (C.2) is

$$\mathbf{w}^1 = [1,0]^T \quad \text{with} \quad \mathbf{x}^1 = [1/2,1]^T, \tag{C.7}$$

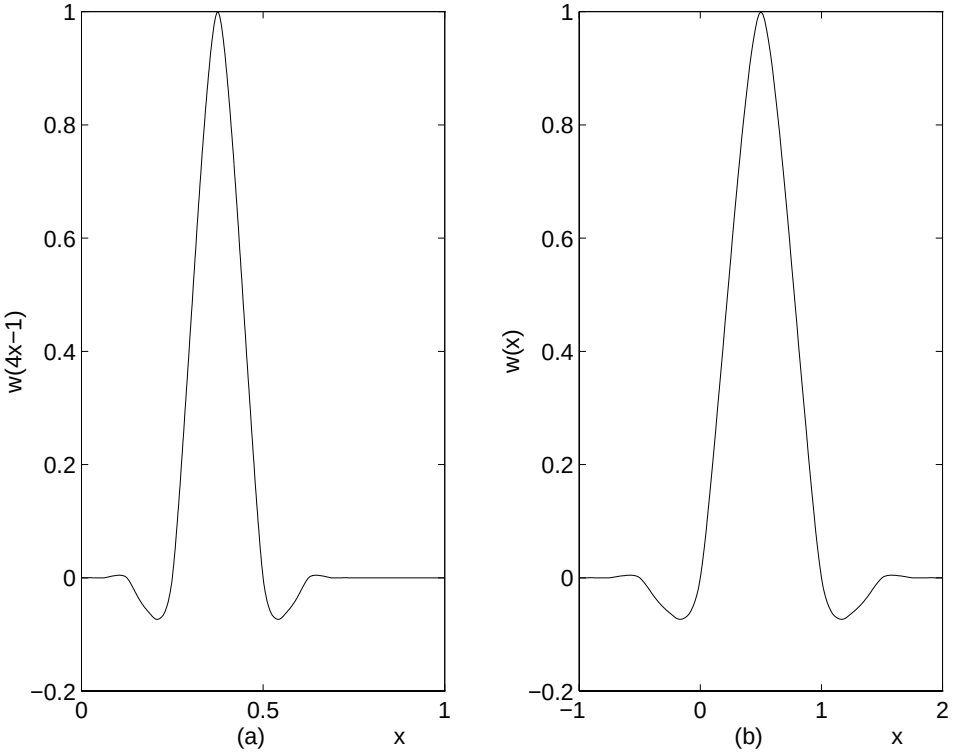


FIG. C.2. *Scaling function for noncompact cubic interpolation: (a) $w(4x-1)$, (b) $w(x)$.*

and the *output* array of the fill-in algorithm is a *table* of numerical values of $w(x)$ expressed in vector form

$$\mathbf{w}^{max} = \mathbf{tab}(w) = [w(x_1^{max}), w(x_2^{max}), \ldots, w(1)]^T \tag{C.8}$$

on the grid

$$\mathbf{x}^{max} = \mathbf{tab}(x) = [x_1^{max}, x_2^{max}, \ldots, 1]^T, \qquad x_j^{max} = j/2^{max}. \tag{C.9}$$

(The column vector notation $\mathbf{tab}(w)$ can be found, for example, in Dahlquist and Björck [Da74].) The dimensions of $\mathbf{tab}(w)$ and $\mathbf{tab}(x)$ are both 2^{max}. If one chooses the integer max large enough, then a point plot of the elements of $\mathbf{tab}(w)$ versus the points of the mesh $\mathbf{tab}(x)$ is so dense as to appear continuous. For any integer $max \geq n+1$, one has $\mathbf{tab}(w) \in \mathbf{R}^m$, $m = 2^{max}$, and in the $\lim_{max\to\infty}$ the iterative fill-in process converges to a continuous function $w(x) \in C[0,1]$.

For a scalar interpolatory multiresolution algorithm, linear interpolation is the only compact interpolation. If $max = 3$, then $2^{max} = 8$ and $\mathbf{tab}(w)$ has 8 values shown as open circles in the top plot of Figure B.1. For 2^{max} very large, a plot of $\mathbf{tab}(w)$ gives the hat function (Figure A.1). Since the interpolation is compact, the scaling function is the unit interpolation function.

For noncompact interpolation, the numerical computation of the scaling function is slightly more complicated since the support of the scaling function is larger than

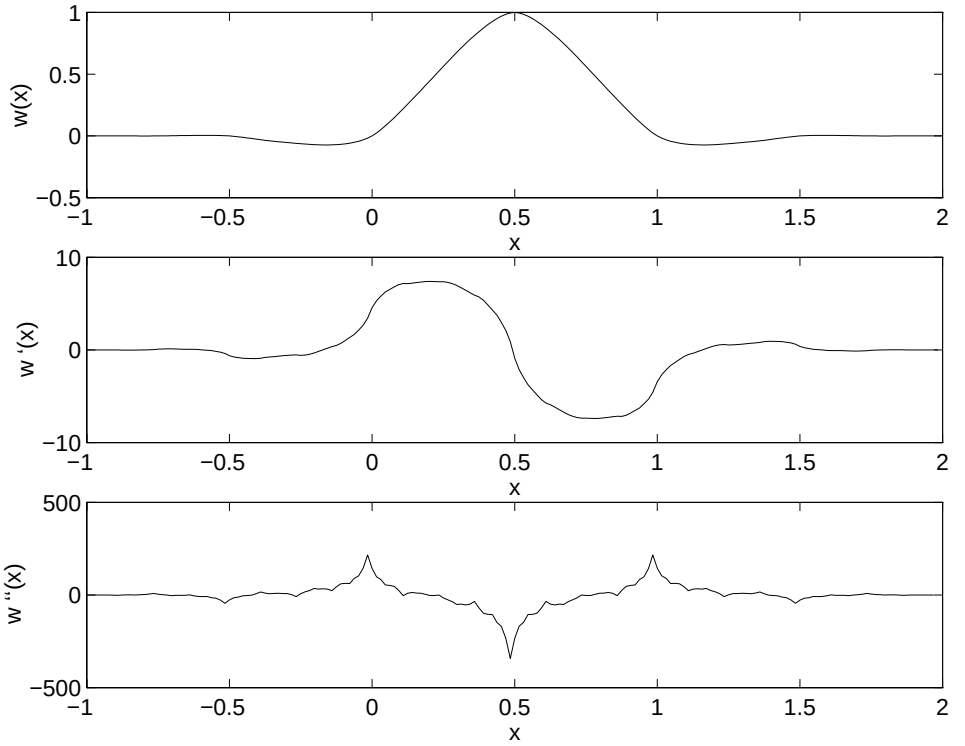


FIG. C.3.

two intervals. Furthermore, the scaling function is not a piecewise polynomial; in fact, it is not piecewise analytic (and it is not the unit interpolation function).

For noncompact cubic interpolation

$$\tilde{u}^{k-1}_{j-1/2} = I^{k-1}(x^{k-1}_{j-1/2}; u^{k-1}_j) = \frac{9}{16}(u^{k-1}_j + u^{k-1}_{j-1}) - \frac{1}{16}(u^{k-1}_{j+1} + u^{k-1}_{j-2}) \tag{C.10}$$

the support of the interpolation formula is three increments:

$$[x_{j-2}, x_{j-1}] \cup [x_{j-1}, x_j] \cup [x_j, x_{j+1}]. \tag{C.11}$$

Consequently, the support of the scaling function for cubic interpolation is six intervals. On the interval $[0, 1]$ with periodic boundary conditions, the index n must be at least 3 in the input vector (C.2) for the scaling function to *fit* in the unit interval with no periodic *wraparound*. For $n = 3$, we pick the input vector

$$\mathbf{w}^3 = [0, 0, 1, 0, 0, 0, 0, 0]^T, \quad \text{on the grid} \quad \mathbf{x}^3 = [1/8, 2/8, 3/8, \ldots, 1]^T. \tag{C.12}$$

There is a unit perturbation at $x = 3/8$, and there are three intervals of length $1/8$ to the left of $x = 3/8$ and five intervals to the right. Application of the fill-in algorithm (C.3) with the input vector (C.12) yields the scaling function (B.2a):

$$w_6(x) = w(4x - 1), \qquad 0 \le x \le 1. \tag{C.13}$$

This function is plotted in Figure C.2(a). On the unit interval the function (C.13) vanishes outside the interval $(0, 3/4)$ since the support is six intervals of length $1/8$. In

Figure C.2(b) we plot the scaling function $w(x)$ on the interval $(-1, 2)$. In the absence of any finite boundary effects, the function $w(x)$ vanishes outside the interval $(-1, 2)$. Note that the support of the scaling function for cubic interpolation is three times the support of the (hat) scaling function for linear interpolation plotted in Figure A.1.

For a *noncompact* scheme such as cubic interpolation (3.9), the scaling function (see, e.g., Figure C.2(b)) looks smooth but in fact is not even locally analytic. The *noncompact* scaling function $w(x)$ is continuously differentiable, and the function $w'(x)$ is *close* to being differentiable (see [Du86] for details). For cubic interpolation the scaling function computed by the fill-in algorithm and the first two derivatives computed by finite differences are shown in Figure C.3. At the end of his paper Dubuc [Du86] lists some open problems related to the scaling (or fundamental) function plotted in Figure C.2(b), e.g., "Is it possible that $w''(x)$ does not exist at any point x of $(-1, 2)$?".

Finally, one should note that, in the special case where the input function (C.2) is the point discretization of a polynomial of degree 3 or less, then the filled-in function $\tilde{u}(x)$ for cubic interpolation will be exact for appropriately modified nonperiodic boundary conditions.

Appendix D. Fill-in algorithm for vector multiresolution. The fill-in algorithm for vector multiresolution is an obvious generalization of the scalar algorithm given in Appendix C. If one starts at the finest grid X^n and applies the reconstruction algorithm (7.14b) with the error or residual term set to zero and with increasing values of k, then one obtains an iterative method of interpolation or the *fill-in* between function values and derivatives defined on the grid X^n. For the vector multiresolution algorithm (7.14) the fill-in algorithm is

fillin

For $k = n+1, n+2, \ldots, max$
  For $j = 1, 2, \ldots, J_{k-1}$

$$\begin{aligned} \boldsymbol{\mathcal{U}}^k_{2j} &= \boldsymbol{\mathcal{P}}_{/\boldsymbol{\alpha}}(\boldsymbol{\mathcal{U}}^{k-1}_j), \\ \boldsymbol{\mathcal{U}}^k_{2j-1} &= \boldsymbol{\mathcal{I}}^k[x^k_{2j-1}; \boldsymbol{\mathcal{U}}^k_{2j}], \end{aligned} \tag{D.1}$$

  End
End

At each level k, additional values $\tilde{\boldsymbol{\mathcal{U}}}(x^k_{2j-1})$ are generated at the dyadic grid points

$$X^k - X^{k-1} = \{x^k_{2j-1}\}_{j=1}^{J_{k-1}}. \tag{D.2}$$

The number of grid points in the filled-in function is 2^{max}.

The accuracy of the filled-in function is determined by the interpolation formula used in a vector multiresolution algorithm. Consider the vector multiresolution of section 6 where function values and derivatives are used. The cubic Hermite interpolation formula (6.4) yields a (cubic) filled-in function $\tilde{u}(x)$ between the original grid points X^n. As an example, we compute the filled-in function for $\sin(2\pi x)$, $0 \le x \le 1$ with $n = 2$. There are only four grid points and four function values (and four derivative values) on the given grid as indicated by the open circles of Figure D.1. In regards to plotting accuracy, there is no discernible difference between the exact and filled-in sine wave as illustrated in Figure D.1. One also obtains the filled-in derivative

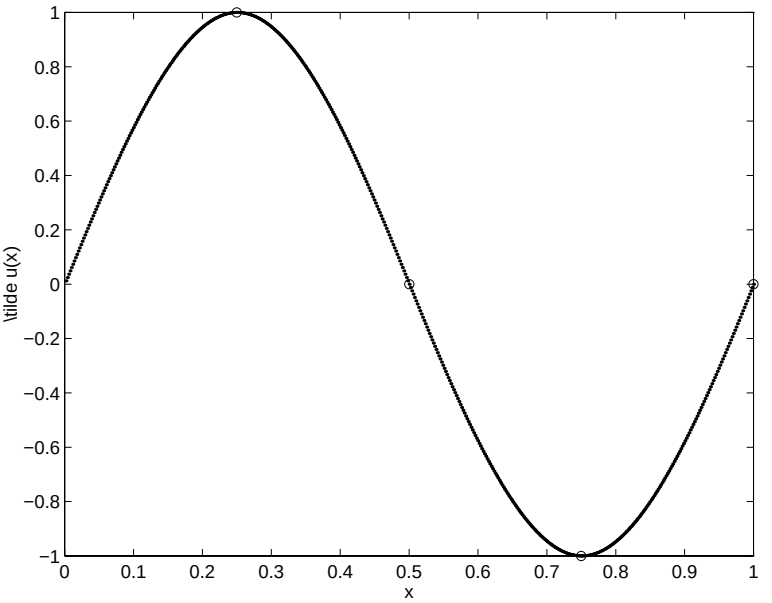


FIG. D.1. *Fill-in function for* $\sin(2\pi x)$.

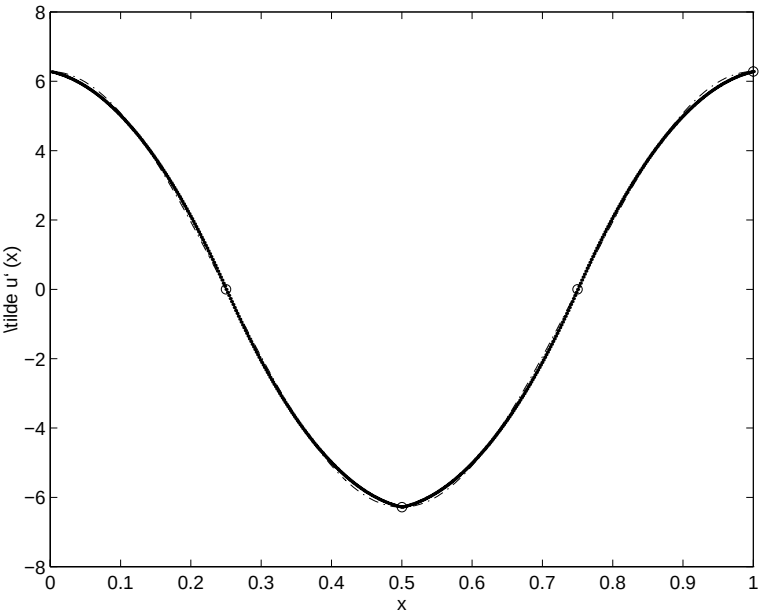


FIG. D.2. *Fill-in function for* $\sin'(2\pi x)$.

shown in Figure D.2. The filled-in derivative $\tilde{u}'(x)$ is plotted as a solid line, and the exact derivative is plotted as a dash-dot line. The filled-in derivative is a quadratic polynomial on each subinterval of the original grid.

Appendix E. Analytic formulas for the unit interpolation functions. The support of an interpolation formula is defined to be the number of subintervals spanned by the interpolation formula. Recall that in deriving a vector compact interpolation formula, the function value and $(\ell-1)$ derivatives are given at each end point of the subinterval $[x_{j-1}, x_j]$. Consequently, the support of a compact interpolation formula is one interval. The support of a unit interpolation function for an interpolatory multiresolution algorithm is equal to twice the support of the interpolation formula. Consequently, since we are utilizing compact interpolation, the support of the vector multiresolution unit interpolation functions is two intervals. The resulting unit interpolation functions are piecewise polynomials (two pieces) of odd degree $2\ell - 1$ with $\ell - 1$ continuous derivatives.

As previously noted, neither the development nor the implementation of an interpolatory multiresolution algorithm requires an explicit knowledge of the unit interpolation functions. However, in practice one needs to select unit interpolation functions appropriate for the particular application, and naturally, one would like to know what unit interpolation functions the multiresolution algorithm is producing. In this appendix we derive analytical formulas for the unit interpolation functions for several cases. The derivation follows the construction outlined in the first paragraph of section A.1. The unit interpolation functions of this appendix are called finite elements by Strela and Strang [Stre95]. They note that local interpolation distinguishes finite elements.

E.1. Linear interpolation. We divide the unit interval into two equal subintervals, write $[0, 1] = [0, 1/2] \cup [1/2, 1]$, and construct the unit interpolation function for linear interpolation (5.1). We assume that the unit interpolation function $w(x)$ is zero at both ends of the interval $[0, 1]$ and that there is a unit perturbation at the center point $w(1/2)$:

$$w(0) = 0, \tag{E.1a}$$

$$w(1) = 0, \tag{E.1b}$$

$$w(1/2) = 1. \tag{E.1c}$$

Using (E.1b) and (E.1c) evaluating the coefficients (5.1b) and (5.1c), and applying (5.1a) on the interval $[1/2, 1]$, one obtains the linear function

$$w(x) = -2(x - 1) \quad \text{for} \quad 1/2 \le x \le 1.$$

An analogous computation for the interval $[0, 1/2]$ yields the other half of a symmetric (about $x = 1/2$) piecewise linear unit interpolation function

$$w(x) = \begin{cases} 2x & \text{for} \quad 0 \le x \le 1/2 \\ -2(x-1) & \text{for} \quad 1/2 \le x \le 1. \end{cases} \tag{E.2}$$

The hat function (E.2) can also be written as (A.8) and is plotted in Figure A.1.

E.2. Hermite interpolation matching function values and first derivatives. Again the unit interval is divided into two equal subintervals. From the Hermite interpolation formula (5.5) we construct two independent unit interpolation functions. We assume that a unit interpolation function $w(x)$ and its derivative $w'(x)$ are zero at the ends of the interval $[0, 1]$:

$$w(0) = w'(0) = 0, \tag{E.3a}$$

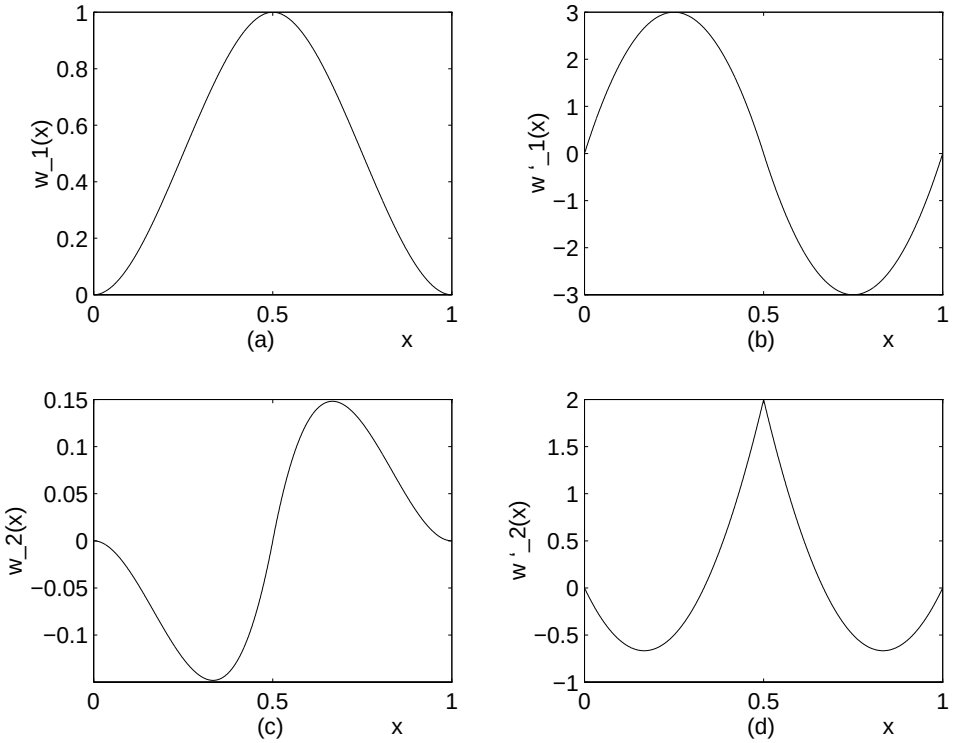


FIG. E.1. *Unit interpolation functions (and derivatives) for Hermite interpolation matching function values and first derivatives.*

$$w(1) = w'(1) = 0. \tag{E.3b}$$

For the center point $x = 1/2$ there are two cases:

$$\text{case a:} \quad w(1/2) \neq 0, \quad \text{and} \quad w'(1/2) = 0 \tag{E.4}$$

and

$$\text{case b:} \quad w(1/2) = 0, \quad \text{and} \quad w'(1/2) \neq 0. \tag{E.5}$$

For case (a) we assume there is a unit perturbation at the center point $x = 1/2$,

$$w(1/2) = 1, \tag{E.6}$$

and interpolate on the interval $[1/2, 1]$. Using the conditions (E.3b), (E.4), and (E.6), evaluating the coefficients (5.8), and applying (5.5), one obtains the cubic polynomial

$$w_1(x) = 4(x-1)^2(4x-1) \quad \text{for} \quad 1/2 \leq x \leq 1. \tag{E.7}$$

An analogous computation for the interval $[0, 1/2]$ yields the other half of a symmetric Hermite unit interpolation function

$$w_1(x) = \begin{cases} -4x^2(4x-3) & \text{for} \quad 0 \leq x \leq 1/2 \\ 4(x-1)^2(4x-1) & \text{for} \quad 1/2 \leq x \leq 1. \end{cases} \tag{E.8}$$

The unit interpolation function (E.8) is plotted in Figure E.1(a). The derivative interpolation function corresponding to case (a) is the derivative of $w_1(x)$:

$$w_1'(x) = \begin{cases} -24x(2x-1) & \text{for} \quad 0 \le x \le 1/2 \\ 24(x-1)(2x-1) & \text{for} \quad 1/2 \le x \le 1. \end{cases} \tag{E.9}$$

This function is plotted in Figure E.1(b).

For case (b) we recall from section 6 that in defining an interpolatory multiresolution algorithm it was convenient to remove the explicit dependence on the spacing h and we let $v(x) = hu'(x)$ on the mesh points. Hence we assume a unit perturbation for $hw'(x)$ at the center point $x = 1/2$:

$$hw'(1/2) = 1 \quad \text{or} \quad w'(1/2) = 2 \tag{E.10}$$

since $h = 1/2$. There follows a skew-symmetric Hermite unit interpolation function

$$w_2(x) = \begin{cases} 4x^2(2x-1) & \text{for} \quad 0 \le x \le 1/2 \\ 4(x-1)^2(2x-1) & \text{for} \quad 1/2 \le x \le 1. \end{cases} \tag{E.11}$$

The function (E.11) is plotted in Figure E.1(c). The derivative interpolation function corresponding to case (b) is the derivative of $w_2(x)$:

$$w_2'(x) = \begin{cases} 8x(3x-1) & \text{for} \quad 0 \le x \le 1/2 \\ 8(x-1)(3x-2) & \text{for} \quad 1/2 \le x \le 1. \end{cases} \tag{E.12}$$

The function (E.12) is plotted in Figure E.1(d). In the discrete multiresolution algorithm, the actual derivative interpolation functions are $hw_1'(x)$ and $hw_2'(x)$. Analytical formulas and plots of the unit interpolation functions for the case of Hermite interpolation matching function values, first derivatives, and second derivatives are given in Appendix G.3 of [Wa97]. In addition, analytical formulas and plots of the unit interpolation functions for the case of Hermite interpolation matching function values and second derivatives are given in Appendix G.4 of [Wa97].

Acknowledgment. We are grateful to Dennis Jesperson for comments and suggestions that improved our manuscript.

REFERENCES

[Al97] A. Aldroubi, *Oblique and hierarchical multiwavelet bases*, Appl. Comput. Harmon. Anal., 4 (1997), pp. 231–263.

[Be96] R. Beam and R. Warming, *Multiresolution analysis and supercompact multiwavelets*, SIAM J. Sci. Comput., 22 (2000), pp. 1238–1268.

[Ca96] W. Cai and J. Z. Wang, *Adaptive multiresolution collocation methods for the initial boundary value problems of nonlinear PDEs*, SIAM J. Numer. Anal., 33 (1996), pp. 937–970.

[Da74] G. Dahlquist and Å. Björck, *Numerical Methods*, Prentice-Hall, Englewood Cliffs, NJ, 1974.

[De89] G. Deslauriers and S. Dubuc, *Symmetric iterative interpolation process*, Constr. Approx., 5 (1989), pp. 49–68.

[Do92] D. L. Donoho, *Interpolating Wavelet Transforms*, Technical Report 408, Department of Statistics, Stanford University, Stanford, CA, 1992.

[Du86] S. Dubuc, *Interpolation through an iterative scheme*, J. Math. Anal. Appl., 114 (1986), pp. 185–204.

[Ha93] A. Harten, *Discrete multi-resolution analysis and generalized wavelets*, Appl. Numer. Math., 12 (1993), pp. 153–192.

[Ha94] A. Harten, *Multiresolution Representation and Numerical Algorithms: A Brief Review*, ICASE Report No. 94-59, 1994.

[Ha95] A. Harten, *Multiresolution algorithms for the numerical solution of hyperbolic conservation laws*, Comm. Pure Appl. Math., 48 (1995), pp. 1305–1342.

[St89] G. Strang, *Wavelets and dilation equations: A brief introduction*, SIAM Rev., 31 (1989), pp. 614–627.

[St94] G. Strang and V. Strela, *Orthogonal multiwavelets with vanishing moments*, J. Optical Eng., 33 (1994), pp. 2104–2107.

[Stre95] V. Strela and G. Strang, *Finite element multiwavelets*, in Approximation Theory, Wavelets, and Applications, NATO Adv. Sci. Inst. Ser. C Math. Phys. Sci. 454, 1995, pp. 485–496.

[Sw95] W. Sweldens, *The Lifting Scheme: A New Philosophy in Biorthogonal Wavelet Constructions*, Proc. SPIE 2569, 1995, pp. 68–79.

[Sw96] W. Sweldens and P Schroder, *Building your own wavelets at home*, in Wavelets in Computer Graphics, ACM SIGGRAPH Course Notes, 1966.

[Wa97] R. Warming and R. Beam, *Discrete Multiresolution Analysis Using Hermite Interpolation: Biorthogonal Multiwavelets*, NASA TM-110434, 1997.

SIAM J. SCI. COMPUT.
Vol. 22, No. 4, pp. 1318–1332
© 2000 Society for Industrial and Applied Mathematics

ROBUST APPROXIMATE INVERSE PRECONDITIONING FOR THE CONJUGATE GRADIENT METHOD*

MICHELE BENZI†, JANE K. CULLUM‡, AND MIROSLAV TŮMA§

Abstract. We present a variant of the AINV factorized sparse approximate inverse algorithm which is applicable to any symmetric positive definite matrix. The new preconditioner is breakdown-free and, when used in conjunction with the conjugate gradient method, results in a reliable solver for highly ill-conditioned linear systems. We also investigate an alternative approach to a stable approximate inverse algorithm, based on the idea of diagonally compensated reduction of matrix entries. The results of numerical tests on challenging linear systems arising from finite element modeling of elasticity and diffusion problems are presented.

Key words. sparse linear systems, finite element matrices, preconditioned conjugate gradients, factorized sparse approximate inverses, incomplete conjugation, stabilized AINV, diagonally compensated reduction

AMS subject classifications. Primary, 65F10, 65N22, 65F50; Secondary, 15A06

PII. S1064827599356900

1. Introduction.

We consider the solution of sparse linear systems $Ax = b$, where A is a symmetric and positive definite (SPD) matrix, by the preconditioned conjugate gradient method. In the last few years there has been considerable interest in explicit preconditioning techniques based on directly approximating A^{-1} with a sparse matrix M; see, e.g., [7], [8], [16], [18], [23], [24], [27], [31], and the recent survey [10]. Sparse approximate inverses have been shown to result in good rates of convergence of the preconditioned iteration (comparable to those obtained with incomplete factorization methods) while being well suited for implementation on vector and parallel architectures; see, e.g., [6], [9], [12], [21].

Although the main motivation for the development of sparse approximate inverse preconditioners comes from parallel processing, it is becoming clear that these techniques are also of interest because of their robustness. Sparse approximate inverses are often applicable to difficult problems where other preconditioners may break down [4]. For instance, incomplete factorization preconditioners, while widely popular and fairly robust, are not always reliable, in that the incomplete factorization process may

*Received by the editors June 4, 1999; accepted for publication (in revised form) June 19, 2000; published electronically October 25, 2000. This work was performed by an employee of the U.S. Government or under U.S. Government contract. The U.S. Government retains a nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or allow others to do so, for U.S. Government purposes. Copyright is owned by SIAM to the extent not limited by these rights.

http://www.siam.org/journals/sisc/22-4/35690.html

†Computer Research and Applications Group (CIC-3), MS B256, Los Alamos National Laboratory, Los Alamos, New Mexico 87545. Current address: Department of Mathematics and Computer Science, Emory University, Atlanta, GA 30322 (benzi@mathcs.emory.edu). The work of the first author was supported in part by Department of Energy grant W-7405-ENG-36 with Los Alamos National Laboratory.

‡Computer Research and Applications Group (CIC-3), MS B256, Los Alamos National Laboratory, Los Alamos, New Mexico 87545 (cullum@lanl.gov). The work of the second author was supported by Department of Energy grant W-7405-ENG-36 and by Department of Energy Applied Mathematical Sciences Program grant KC-07-01-01.

§Institute of Computer Science, Academy of Sciences of the Czech Republic, Pod vodárenskou věží 2, 182 07 Prague 8, Czech Republic (tuma@cs.cas.cz). The work of the third author was supported by Grant Agency of the Czech Academy of Sciences grants 2030706 and 2030801.

suffer from various types of instability [17], [20]. Even in the SPD case, existence of the standard incomplete Cholesky (IC) factorization [34] is guaranteed only for special classes of matrices, such as H-matrices [33]. For general SPD matrices, breakdown in the IC process may occur due to exceedingly small or negative pivots. For this reason, variants of IC have been developed which are applicable to any SPD matrix without breakdowns; see [1], [25], [28], [37], [38]. However, some of these modifications are expensive, while others require diagonal perturbations which introduce additional parameters in the algorithm.

We are interested in parallel preconditioners that are widely applicable, reliable, and effective at reducing the number of iterations. The factorized sparse approximate inverse (FSAI) method developed by Kolotilina and Yeremin in [31] is one of the very few naturally parallel techniques that are also quite effective and highly stable for arbitrary SPD matrices. In this method, a lower triangular matrix G is computed as a sparse approximation to L^{-1}, where $A = LL^T$ is the Cholesky factorization of A. The algorithm does not require any information about L and works exclusively with A. Entries of G are computed as solutions of small "local" linear systems having principal submatrices of A as coefficient matrices. These are SPD and can be solved in a stable way by standard direct methods. Hence, no breakdowns are possible in computing G. Also, G is necessarily nonsingular, and the preconditioner $M := G^T G \approx A^{-1}$ is SPD and can be used with the conjugate gradient method. The standard form of this algorithm requires a prescribed sparsity pattern for G. Some ideas for determining sparsity patterns can be found in [16] and [26], but the effectiveness of these heuristics for factorized approximate inverses remains to be investigated.

A different factorized sparse approximate inverse preconditioner, based on incomplete conjugation (A-orthogonalization) of the unit basis vectors, is the AINV preconditioner [7], [8]. This method does not require the sparsity pattern of the approximate inverse factors to be specified in advance; rather, a good sparsity pattern is determined *dynamically* as the preconditioner is being computed. This is done by applying a drop tolerance to the computed entries of the inverse factors. As shown in [6], the calculation of the preconditioner can be parallelized using graph partitioning. However, the preconditioner may not be well defined for a general SPD matrix, due to breakdowns—see the next section for the definition of breakdown in the context of AINV. A sufficient condition for AINV to be breakdown-free is that A be an H-matrix [7]. We recall that M-matrices and diagonally dominant matrices are examples of H-matrices. If A is far from being an M-matrix or diagonally dominant, for example, if A has large positive off-diagonal entries, the preconditioner may not be defined or may fail to be positive definite. This is indeed the case for many problems arising from finite element modeling of structures and thin shells [5] and for certain diffusion problems involving highly distorted meshes [35].

In this paper we present a variant of the AINV preconditioner which is well defined (in exact arithmetic) for an arbitrary SPD matrix. We refer to the resulting preconditioner as the stabilized AINV, or SAINV, preconditioner. It is mathematically equivalent to the standard AINV algorithm when no dropping is applied.

The price to pay for the added robustness is the slightly higher cost of computing the preconditioner with respect to the standard AINV algorithm. However, for most problems the cost of forming the preconditioner is still reasonable. As we will see, matrix reorderings and scalings can be used to reduce costs.

We mention that breakdown-free variants of AINV have also been developed, independently, by Bridson [14] and by Kharchenko et al. [29]. Bridson's scheme is

different from ours. The variant of Kharchenko et al., on the other hand, is identical to ours. Their paper offers independent further evidence of the reliability of this approach.

Together with SAINV, we explore another approach to preventing breakdowns in the AINV process, based on the notion of *diagonally compensated reduction of positive off-diagonal entries*. This technique, introduced and analyzed by Axelsson and Kolotilina in [3], associates with any SPD matrix a Stieltjes matrix (that is, a symmetric and necessarily positive definite M-matrix) in a natural way. The standard AINV process can be applied to this new matrix, without breakdowns, resulting in an approximate inverse that can be used as a preconditioner for the original system.

The remainder of the paper is organized as follows. In section 2 we recall the standard AINV algorithm and review the techniques that have been used so far to handle breakdowns. In section 3 we introduce the stabilized AINV algorithm for SPD matrices, and in section 4 we describe the alternative approach based on diagonally compensated reduction. Section 5 is devoted to numerical experiments using challenging matrices from finite element modeling, including some experiments with other methods. Finally, in section 6 we make some concluding remarks and suggestions for further work.

2. The AINV algorithm. From now on, $A \in \mathbb{R}^{n\times n}$ is assumed to be SPD. The AINV algorithm [7] builds a factorized sparse approximate inverse of the form

$$M = ZD^{-1}Z^T \approx A^{-1}, \tag{2.1}$$

where Z is a unit upper triangular matrix and D is diagonal. The approximate inverse factor Z is a sparse approximation of the inverse of the L^T factor in the LDL^T decomposition of A. When the inverse factorization is performed exactly, the diagonal matrix D is the same in the two decompositions and contains the *pivots* down the main diagonal.

The AINV algorithm computes Z and D directly from A by means of an incomplete A-orthogonalization process applied to the unit basis vectors. In this process, small elements are dropped to preserve sparsity. The underlying assumption is that most entries in L^{-1} are small in magnitude. This is true for many problems of practical interest, particularly for discretizations of partial differential equations of elliptic type. Sparsity can also be achieved by combining the dropping of small entries with suitable sparse matrix orderings, such as nested dissection and minimum degree. These orderings are beneficial in that they result in smaller time and space requirements for forming and storing the preconditioner, while at the same time improving the quality of the preconditioner in a significant number of cases; see [11], [15], [22].

In order to describe the procedure, let a_i^T denote the ith row of A. Also, let e_i denote the ith unit basis vector. The basic A-orthogonalization procedure can be written as follows.

ALGORITHM 2.1.

(1) *Let* $z_i^{(0)} = e_i \quad (1 \le i \le n)$
(2) *For* $i = 1, 2, \ldots, n$ *do*
(3) *For* $j = i, i+1, \ldots, n$ *do*
(4) $p_j^{(i-1)} := a_i^T z_j^{(i-1)}$
(5) *End do*
(6) *if* $i = n$ *go to* (11)
(7) *For* $j = i+1, \ldots, n$ *do*

(8) $$z_j^{(i)} := z_j^{(i-1)} - \left(\frac{p_j^{(i-1)}}{p_i^{(i-1)}}\right) z_i^{(i-1)}$$

(9) *End do*

(10) *End do*

(11) *Let* $z_i := z_i^{(i-1)}$ *and* $p_i := p_i^{(i-1)}$, *for* $1 \le i \le n$. *Return* $Z = [z_1, z_2, \ldots, z_n]$ *and* $D = \mathrm{diag}(p_1, p_2, \ldots, p_n)$.

Sparsity is preserved by dropping off-diagonal entries in the z-vectors after the updates at step (8). The resulting (incomplete) algorithm is sometimes referred to as the *right-looking* AINV process. A left-looking variant also exists [9], which is sometimes advantageous. These algorithms can be extended in a straightforward manner to the nonsymmetric case [8].

A *breakdown* is defined as a negative or zero value of a pivot p_i. When no dropping is applied, $p_i = z_i^T A z_i > 0$. The incomplete procedure is well defined, i.e., no breakdown can occur, if A is an H-matrix (in the absence of round-off). When no breakdown occurs, the resulting sparse approximate inverse preconditioner is usually quite effective; see [9], [10] for comparisons between AINV and other preconditioners. In the general case, breakdowns can occur. Breakdowns have a crippling effect on the quality of the preconditioner. A negative p_i would result in an approximate inverse which is not positive definite; a zero pivot would force termination of the procedure, since step (8) cannot be carried out. In practice, exactly zero pivots are very unlikely to occur, but exceedingly small pivots can happen, resulting in uncontrolled growth of the entries of Z and extremely high fill-in.

In [7], a dynamic strategy to shift negative or numerically small pivots away from zero was proposed. Unfortunately, in the vast majority of cases the resulting sparse approximate inverse is not a good preconditioner. A somewhat better strategy is to adapt to AINV the a priori diagonal shift technique introduced by Manteuffel [33] for the IC factorization. Whenever a negative or exceedingly small pivot is encountered, the AINV process is terminated and reattempted on a new matrix $A' := A + \alpha \mathrm{diag}(A)$. Here $\mathrm{diag}(A)$ denotes the main diagonal of A, and $\alpha > 0$ denotes a parameter such that A' has a stable approximate inverse factorization, to be determined by trial and error. That such an α must exist is clear, since the AINV process cannot break down on a diagonally dominant matrix, and α can be chosen so large as to make A' diagonally dominant. However, the *optimal* value of α is usually much smaller than the one that makes A' diagonally dominant. Although this shifting strategy has proved successful in handling some difficult problems, it is expensive and frequently produces preconditioners of poor quality.

In the next section we propose a reformulation of the AINV algorithm that is applicable, without breakdowns, to any SPD matrix.

3. Stabilized AINV. First we need to take a close look at the mechanism of breakdown. We begin by writing down the explicit formula for the p_j's:

$$p_j^{(i-1)} = a_i^T z_j^{(i-1)} = \sum_{l=1}^{i-1} a_{il} z_{lj}^{(i-1)} + a_{ij} \qquad (i \le j \le n). \tag{3.1}$$

Here a_{ij} is the (i,j) entry of A, and $z_{lj}^{(i-1)}$ denotes the lth entry of vector $z_j^{(i-1)}$. Suppose now that a dropping rule is applied in the calculation of the z-vectors. The modified z-vectors will be denoted by $\bar{z}_j^{(i-1)}$, and the corresponding pivots are given

by

$$\bar{p}_i^{(i-1)} = \sum_{l=1}^{i-1} a_{il}\bar{z}_{li}^{(i-1)} + a_{ii} \qquad (1 \le i \le n). \tag{3.2}$$

(We explicitly stipulate that no dropping is applied to the diagonal entries of Z or to the diagonal pivots.) When A is an M-matrix, it is easily seen by induction that all the $\bar{z}_{lj}^{(i-1)}$ are nonnegative [7]. Because the off-diagonal entries of A are nonpositive, the action of dropping one entry in the z-vector cannot cause the (inexact) pivot $\bar{p}_i^{(i-1)}$ to decrease; it either remains unchanged, or it increases, depending on whether the corresponding coefficient a_{il} is zero or not. Because the exact pivots are positive, the AINV process cannot break down. Indeed, dropping has the effect of stabilizing the AINV process even further. This is in perfect analogy with the IC factorization for M-matrices [34].

Dropping an entry at step $(i-1)$ of the AINV process amounts to setting $\bar{z}_{lj}^{(i-1)} = 0$ for some l. Therefore, if either

$$a_{il} > 0 \qquad \text{and} \qquad z_{li}^{(i-1)} > 0$$

or

$$a_{il} < 0 \qquad \text{and} \qquad z_{li}^{(i-1)} < 0,$$

then the inexact pivot will be smaller than the exact pivot. In particular, if either one of a_{il} or $z_{li}^{(i-1)}$ is positive and large and the other is positive and not small, the inexact pivot can be significantly smaller than the exact one and can even become negative. In reality, the dropping rule will cause many of the $\bar{z}_{lj}^{(i-1)}$ to be zero, and the net change in the ith pivot will be the result of cumulative effects, some of which tend to increase and others to decrease the value of $\bar{p}_i^{(i-1)}$. Such effects may cancel each other out. Nevertheless, we have observed in numerical experiments performed on matrices with relatively large positive off-diagonal entries that the inexact pivots can become negative and, in fact, rather large in absolute value. The matrices for which this behavior has been observed were not artificial examples but came from real applications in structural analysis. It is no surprise that just setting these negative pivots equal to some arbitrary positive quantity resulted in preconditioners of very poor quality.

The way to avoid nonpositive pivots is simply to recall that in the exact A-orthogonalization process, the p_i's are the diagonal entries of matrix D which satisfies the matrix equation

$$Z^T A Z = D;$$

hence for $1 \le i \le n$

$$p_i = z_i^T A z_i > 0$$

since A is SPD and $z_i \neq 0$. (Recall that the ith entry of z_i is equal to 1.) In the exact process, the following equality holds:

$$p_i = z_i^T A z_i = a_i^T z_i. \tag{3.3}$$

This identity follows immediately from the fact that Z is unit upper triangular and AZ is lower triangular. Clearly, it is more economical to compute the pivots using the expression on the right-hand side of (3.3) rather than that in the middle. The same goes for the p_j's:

$$p_j = z_i^T A z_j = a_i^T z_j. \tag{3.4}$$

(For brevity, we have omitted the $(i-1)$ superscripts in the above formulas.) However, because of dropping and the resulting loss of A-orthogonality in the $\bar{z}$-vectors, such identities no longer hold in the inexact process, and for some matrices one can have

$$a_i^T \bar{z}_i \ll \bar{z}_i^T A \bar{z}_i$$

with the concomitant possibility of breakdowns.

These simple observations point to several reformulations of the AINV algorithm that are breakdown-free in exact arithmetic. The simplest thing to do is to use the standard AINV process (Algorithm 2.1 with dropping) as far as possible, and to switch to formula $\bar{p}_i = \bar{z}_i^T A \bar{z}_i$ whenever a negative or exceedingly small pivot shows up. Unfortunately, this simple fix is not good enough. It typically results in slow convergence of the PCG iteration, and for really hard problems there may be no convergence within a reasonable number of steps. The reason is that by the time a negative pivot occurs, the loss of information about the true inverse due to dropping has already been so great that no "local" trick can succeed in recovering a good preconditioner. A more global strategy is needed.

We found that a robust algorithm requires that the $\bar{p}_i$'s be computed using the quadratic form $\bar{z}_i^T A \bar{z}_i$ throughout the entire AINV process, for $i = 1, \ldots, n$. This still leaves the question of how to compute the $\bar{p}_j$'s with $i+1 \le j \le n$. One could either use the inexpensive formula (3.1) or the more expensive bilinear form (3.4). In the latter case, we have

$$\bar{p}_j = \bar{v}_i^T \bar{z}_j, \qquad \text{where} \qquad \bar{v}_i^T := \bar{z}_i^T A.$$

Because the vector $\bar{v}_i$ has already been computed as part of the calculation of $\bar{p}_i$, this approach is only slightly more expensive than using formula (3.1). The difference depends on how much more dense $\bar{v}_i^T$ is compared to a_i^T. It turns out that using the bilinear form (3.4) results in a preconditioner of much higher quality for nearly the same cost, so it definitely pays off to use the more expensive approach.

Hence, we obtain a reliable version of the AINV algorithm based on the following reformulation of Algorithm 2.1.

ALGORITHM 3.1.

(1) *Let* $z_i^{(0)} = e_i \quad (1 \le i \le n)$
(2) *For* $i = 1, 2, \ldots, n$ *do*
(3) $\quad v_i := A z_i^{(i-1)}$
(4) $\quad$ *For* $j = i, i+1, \ldots, n$ *do*
(5) $\quad\quad p_j^{(i-1)} := v_i^T z_j^{(i-1)}$
(6) $\quad$ *End do*
(7) $\quad$ *if* $i = n$ *go to* (12)
(8) $\quad$ *For* $j = i+1, \ldots, n$ *do*
(9) $\quad\quad z_j^{(i)} := z_j^{(i-1)} - \left(\frac{p_j^{(i-1)}}{p_i^{(i-1)}} \right) z_i^{(i-1)}$

(10) *End do*
(11) *End do*
(12) *Let* $z_i := z_i^{(i-1)}$ *and* $p_i := p_i^{(i-1)}$, *for* $1 \le i \le n$. *Return*
$Z = [z_1, z_2, \ldots, z_n]$ *and* $D = \mathrm{diag}(p_1, p_2, \ldots, p_n)$.

Obviously, Algorithms 2.1 and 3.1 are mathematically equivalent. However, the incomplete process obtained by dropping in the z-vectors after step (9) of Algorithm 3.1 leads to a reliable approximate inverse procedure. This algorithm, in exact arithmetic, is applicable to any SPD matrix without breakdowns. Of course, instabilities due to positive but extremely small pivots may occur in finite precision, and a thresholding technique may still be necessary to guard against such possibility. However, we have not met this situation in our tests, and the resulting preconditioner appears to be reliable in practice. This new preconditioner will be hereafter referred to as the SAINV (for stabilized AINV) preconditioner.

What about the cost of SAINV? At first sight it might look very expensive to compute the preconditioner on the basis of Algorithm 3.1, which requires the computation of n matrix-vector products $v_i = Az_i$. To see that this is not nearly as expensive as it looks, it should be noticed that in the incomplete process the $\bar{z}_i$ vectors are kept sparse through the use of dropping (and possibly through reorderings). Hence, the n matrix-vector products can be performed in *sparse-sparse mode*, to borrow the term used in [18]. Hence, computing $\bar{v}_i$ amounts to forming a linear combination of a few columns of A, namely, those that correspond to nonzero entries in $\bar{z}_i$. Also, because $\bar{z}_i$ is the ith column of an upper triangular matrix, it is clear that at step i only the first i columns of A enter the matrix-vector product. Assuming that the drop tolerance is chosen so that the final Z contains $\mathcal{O}(n)$ nonzeros, and assuming an even distribution of nonzeros across the columns of Z, the cost of computing all the $\bar{p}_j$'s in the form of sparse bilinear expressions involving A is linear in the dimension n of the problem. As we shall see in the section on numerical experiments, the run time for computing the SAINV preconditioner is only slightly higher than that for the standard AINV algorithm when the latter does not break down, and the run time tends to be dominated by the time required to perform the iteration phase.

In the next section we briefly describe an alternative, inexpensive approach to systematically avoid breakdowns in the AINV process, based on the diagonally compensated reduction of positive off-diagonal matrix entries.

4. Diagonally compensated reduction approximate inverse. The method of diagonally compensated reduction of positive off-diagonal entries, due to Axelsson [2] and Axelsson and Kolotilina [3], associates with any SPD matrix A an SPD M-matrix $\hat{A}$. In the simplest variant of this technique, $\hat{A}$ is obtained by setting to zero the positive off-diagonal entries a_{ij} of A (reduction), which are subsequently added to the corresponding diagonal entry a_{ii} (diagonal compensation). Formally, the idea is to split A as

$$A = B + R,$$

where R contains the off-diagonal positive entries of A, and to let

$$\hat{A} = B + \Delta,$$

where Δ is the diagonal matrix satisfying

$$\Delta e = Re,$$

where e denotes the vector of all ones. Thus,

$$\hat{A} = A + (\Delta - R).$$

Notice that the symmetric matrix $\Delta - R$ is a singular M-matrix, since it has nonpositive off-diagonal entries and zero row sums; see [13], page 147. In particular, $\Delta - R$ is symmetric positive semidefinite. In turn, this shows that $\hat{A}$ is SPD, because it is the sum of an SPD matrix and a positive semidefinite one. Because $\hat{A}$ has nonpositive off-diagonal entries, it must be an M-matrix [13].

The standard AINV process can be applied to $\hat{A}$ without breakdowns, and the corresponding approximate inverse $M \approx \hat{A}^{-1}$ can be used as a preconditioner for the original linear system $Ax = b$. It is hoped that M will be a good preconditioner for A, provided that A is not too far from being an M-matrix. A rough way to estimate how much a given A deviates from being an M-matrix is to compute the Frobenius norm $\|R\|_F$ of the matrix R which contains the positive off-diagonal entries of A. To make this quantity invariant under global scalings, we divide it by $\|A\|_F$. That is, we let

$$\eta = \frac{\|R\|_F}{\|A\|_F}.$$

Notice that $0 \leq \eta < 1$, with $\eta = 0$ if and only if A is an M-matrix. In the next section we will try to ascertain whether there is a correlation between the size of η and the quality of the preconditioner M obtained by applying the standard AINV approximate inverse algorithm to $\hat{A}$. See [25] for results on the use of diagonally compensated reduction in the context of IC factorizations.

5. Numerical experiments. The standard AINV preconditioner is often applicable, with good results, to matrices that do not satisfy the H-matrix condition [7]. There are, however, applications that lead to matrices for which the standard AINV approach is unstable and produces unreliable preconditioners. This is typically the case for matrices from structural engineering, including finite element modeling of elasticity and thin shell problems [5].

Another class of problems for which the standard AINV method fails due to breakdowns consists of diffusion equations discretized on highly distorted finite element meshes (e.g., Kershaw meshes). Such meshes arise frequently in codes developed at Los Alamos and elsewhere for the modeling of phenomena with complex physics, e.g., radiation diffusion [35].

In this section we present the results of a number of numerical tests performed on a selection of 16 matrices from the 2 areas mentioned above. Of these, 14 are from structural analysis and the remaining 2 are diffusion problems. Most of the matrices from structural analysis can be downloaded from the Matrix Market website [36]. The exceptions are the NASA examples, extracted from the University of Florida Sparse Matrix Collection [19], and the SMT matrix, which was provided by R. Kouhia of the Helsinki University of Technology. The two diffusion problems were extracted from a Los Alamos diffusion package, AUGUSTUS, developed by Mike Hall. These are three-dimensional, steady-state problems defined on the unit cube discretized on a highly skewed Kershaw mesh using the scheme described in [35].

Some basic information about the tests problems is provided in Table 1. For each matrix we provide the problem size n, the number of nonzeros in the lower triangular part nnz, the value of η as defined in the previous section, and the application area. In the last column we give the number of iterations and time (in seconds) required to solve

TABLE 1
Test problem information.

Matrix	n	nnz	η	Application	JCG (Its/Time)
BCSSTK13	2003	42943	0.474	Fluid flow	1406/20.4
BCSSTK14	1806	32630	0.325	Roof of Coliseum	409/4.79
BCSSTK15	3948	60882	0.096	Offshore platform	518/16.4
BCSSTK16	4884	147631	0.221	Dam	191/11.4
BCSSTK17	10974	219812	0.472	Pressure vessel	2522/287.
BCSSTK18	11948	80519	0.202	Power plant	1120/47.0
BCSSTK21	3600	15100	0.248	Clamped plate	559/4.81
BCSSTK25	15439	133840	0.003	Skyscraper	>10000/–
S1RMQ4M1	5489	143300	0.203	Cylindrical shell	692/37.1
S2RMQ4M1	5489	143300	0.293	Cylindrical shell	1529/80.0
S3RMQ4M1	5489	143300	0.302	Cylindrical shell	6884/359.
NASA2910	2910	88603	0.213	NASA structure	1350/51.1
NASA4704	4704	54730	0.248	NASA structure	4866/145.3
SMT	25710	1889447	0.423	Mounted transistor	1984/492.
AUGUSTUS5	134144	645028	0.235	Diffusion	842/136.
AUGUSTUS7	1060864	5187320	0.233	Diffusion	1540/2970.

the linear system using the conjugate gradient method with Jacobi preconditioning (JCG). The iteration was terminated when the 2-norm of the initial residual was reduced by at least 8 orders of magnitude, or when a maximum of 10,000 iterations was reached. The initial guess was the zero vector, and the right-hand side was constructed as $b = Ax$, where x is a vector with random entries, uniformly distributed in $(0, 1)$. Similar results were obtained for other choices of the right-hand side. All the runs were performed on a SUN Ultra 5 workstation, except for those with SMT and the AUGUSTUS diffusion problems, for which one processor of an SGI Origin 2000 was used. The codes were written in standard Fortran77 and compiled with the -O3 optimization option. As can be seen, several of these problems are rather difficult to solve and have substantial positive off-diagonal part R. The standard AINV algorithm is unstable on all these problems, except BCSSTK16, which is the easiest in our data set. The most challenging problems are BCSSTK25 and the thin shell S3RMQ4M1.

In the following tables we present a number of results obtained with various preconditioners. We provide the time for computing the preconditioner (P-time), the number of PCG iterations (Its), the time to perform the PCG iterations (It-time), the total time (Tot-time), and the density ρ of the preconditioner. This is defined as the ratio between the number of nonzeros in the approximate inverse factor and the number of nonzeros in the lower triangular part of A. The preconditioners considered are FSAI [31] and variants of the standard AINV algorithm and of SAINV. The variants correspond to various preliminary transformations operated on the coefficient matrix. These are symmetric diagonal scaling (J), reordering with the multiple minimum degree (MMD) algorithm [32], diagonally compensated reduction of positive off-diagonal entries (DCR), and combinations of these. Thus, for instance, J-DCR-MMD-AINV stands for the standard AINV algorithm applied to the matrix obtained from the diagonally compensated reduction of the original matrix after diagonal scaling and MMD reordering. For AINV and SAINV, the value $\tau = 10^{-1}$ of the drop tolerance is used throughout.

In Table 2 we present results obtained with the FSAI preconditioner [31]. This algorithm already incorporates a sophisticated scaling strategy; therefore, no diagonal scaling was applied. We found that MMD reordering had a beneficial effect on the

TABLE 2
Test results for MMD-FSAI preconditioner.

Matrix	P-time	Its	It-time	Tot-time
BCSSTK13	0.71	440	15.2	15.9
BCSSTK14	0.35	80	2.35	2.70
BCSSTK15	0.54	201	10.6	11.1
BCSSTK16	2.67	72	7.78	10.5
BCSSTK17	2.70	472	82.4	85.1
BCSSTK18	0.63	322	27.0	27.6
BCSSTK21	0.09	260	5.39	5.48
BCSSTK25	1.25	1136	145.	146.
S1RMQ4M1	1.99	233	24.1	26.1
S2RMQ4M1	2.01	293	30.5	32.5
S3RMQ4M1	2.03	446	48.0	50.0
NASA2910	2.51	230	15.6	18.1
NASA4704	0.42	1068	52.6	56.0
SMT	53.4	403	193.	246.
AUGUSTUS5	3.60	363	95.2	98.8
AUGUSTUS7	30.6	705	2042.	2073.

TABLE 3
Test results for BCSSTK16.

Preconditioner	P-time	Its	It-time	Tot-time	ρ
JCG	–	191	11.4	11.4	0.00
MMD-FSAI	2.67	72	7.78	10.5	1.00
AINV	0.79	105	6.33	7.12	0.12
MMD-AINV	1.01	99	5.93	6.94	0.13
J-AINV	0.89	101	6.09	6.98	0.12
J-MMD-AINV	0.99	102	6.02	7.01	0.12
DCR-AINV	0.49	150	8.76	9.25	0.05
DCR-MMD-AINV	0.54	147	8.39	8.93	0.06
J-DCR-AINV	0.51	149	8.63	9.14	0.05
J-DCR-MMD-AINV	0.54	149	8.46	9.00	0.05
SAINV	1.00	101	6.12	7.12	0.12
MMD-SAINV	1.36	96	5.68	7.04	0.13
J-SAINV	0.88	98	5.82	**6.70**	0.12
J-MMD-SAINV	1.05	98	5.72	6.77	0.12

quality of FSAI preconditioning for most problems, especially the larger ones, and this ordering was used in the experiments. The sparsity pattern for the approximate inverse factor G was the same as that of the lower triangular part of $P^T AP$, where P is the permutation matrix corresponding to the MMD ordering.

These results clearly demonstrate the robustness of FSAI. This algorithm is generally more reliable and efficient than JCG. Note, however, the relatively poor performance on BCSSTK21. We mention that the performance of FSAI can be significantly improved using a better choice for the sparsity pattern and by postprocessing the preconditioner; see, e.g., the suggestions in [30].

In Table 3 we present results for BCSSTK16 using a variety of preconditioners. We show these results because the standard AINV is stable for this matrix and we wish to compare it with the new variants. The fastest total timing is in boldface.

There are several observations worth making. We note that AINV and SAINV both produce very sparse and yet quite effective preconditioners. Because of such sparsity, the cost of SAINV is not much higher than that of AINV, particularly with

TABLE 4
Test results for BCSSTK18.

Preconditioner	P-time	Its	It-time	Tot-time	ρ
JCG	–	1120	47.0	47.0	0.00
MMD-FSAI	0.63	322	27.0	27.6	1.00
AINV	unst.	–	–	–	–
MMD-AINV	1.78	7068	501.	503.	1.15
J-AINV	0.47	743	46.6	47.1	0.73
J-MMD-AINV	0.96	1925	118.	119.	0.69
DCR-AINV	0.50	924	49.5	50.0	0.34
DCR-MMD-AINV	0.58	1006	53.3	53.9	0.34
J-DCR-AINV	0.49	663	36.3	36.8	0.35
J-DCR-MMD-AINV	0.56	613	32.2	32.8	0.34
SAINV	3.37	257	18.9	22.3	1.32
MMD-SAINV	2.83	354	24.6	27.4	0.99
J-SAINV	0.89	278	17.1	18.0	0.65
J-MMD-SAINV	1.05	261	15.5	**16.6**	0.58

symmetric scaling. On the other hand, diagonally compensated reduction (which is not needed here since AINV is stable), which also produces very sparse preconditioners, results in slower convergence and higher overall timings. These experiments suggest that SAINV can be superior to AINV even when AINV is stable, provided that the preconditioner is sufficiently sparse.

In Table 4 we show the results for a more difficult problem, BCSSTK18. These results are fairly typical, and this is why we discuss them in some detail. For this example, AINV is unstable. Applying symmetric scaling and reordering for sparsity improves the situation only slightly: convergence is extremely slow. Using diagonally compensated reduction with AINV is somewhat more effective, but worse than MMD-FSAI. We performed some experiments with a smaller value of the drop tolerance to see if a denser preconditioner would help, but this was not the case. The number of iterations was somewhat reduced but not the timings. Better results are obtained with SAINV, particularly in combination with scaling and MMD reordering. Notice that SAINV results in preconditioners which are fairly sparse, although not as much as for the previous example. The time for computing the approximate inverse with SAINV and its variants is still quite small.

In Table 5 we show the results of using AINV with diagonally compensated reduction. We see that this method is reliable but generally not very effective. On difficult matrices, such as BCSSTK18 and S3RMQ4M1, the performance is poor. In some cases, improved results can be obtained in combination with symmetric scalings and MMD reordering, but no clear trend emerges and it is difficult to make specific recommendations.

Table 6 contains results for the SAINV preconditioner. For most problems, the performance is better than that of DCR-AINV and comparable with that of MMD-FSAI. Note that MMD-FSAI performs better on the shell problems.

Finally, in Table 7 we show the results for SAINV used in conjunction with symmetric scaling and MMD reordering. While this combination is not always optimal, it was the best or nearly so in a majority of cases. Therefore we feel comfortable recommending to use this combination in practice. Note in particular that J-MMD-SAINV is on average a factor of three faster than Jacobi preconditioning. Also notice that this approach is better than using AINV with diagonally compensated reduc-

TABLE 5
Test results for DCR-AINV preconditioner.

Matrix	P-time	Its	It-time	Tot-time	ρ
BCSSTK13	0.11	1185	19.8	19.9	0.13
BCSSTK14	0.06	353	4.69	4.75	0.12
BCSSTK15	0.11	676	18.6	18.7	0.15
BCSSTK16	0.49	150	8.76	9.25	0.05
BCSSTK17	0.41	1531	150.	150.	0.12
BCSSTK18	0.50	924	49.5	50.0	0.34
BCSSTK21	0.03	285	3.39	3.42	0.66
BCSSTK25	33.8	1853	256.	290.	2.10
S1RMQ4M1	0.30	539	30.5	30.8	0.10
S2RMQ4M1	0.30	1619	92.4	92.7	0.10
S3RMQ4M1	0.30	7675	440.	440.	0.10
NASA2910	0.18	689	22.6	22.8	0.08
NASA4704	0.11	2935	79.4	79.5	0.26
SMT	5.55	1380	353.	358.	0.02
AUGUSTUS5	3.11	331	82.7	85.8	0.82
AUGUSTUS7	35.2	605	2733.	2761.	0.81

TABLE 6
Test results for SAINV preconditioner.

Matrix	P-time	Its	It-time	Tot-time	ρ
BCSSTK13	5.27	368	11.2	16.5	1.41
BCSSTK14	1.05	78	1.41	2.46	0.73
BCSSTK15	1.66	219	7.79	9.45	0.61
BCSSTK16	1.00	101	6.12	7.12	0.12
BCSSTK17	4.87	919	112.	117.	0.54
BCSSTK18	3.37	257	18.9	22.3	1.32
BCSSTK21	0.24	169	2.70	2.94	1.83
BCSSTK25	15.4	1848	286.	302.	2.20
S1RMQ4M1	8.04	267	23.8	31.8	1.06
S2RMQ4M1	13.8	521	60.4	74.2	1.91
S3RMQ4M1	36.4	4239	679.	715.	3.24
NASA2910	1.80	372	15.4	17.4	0.49
NASA4704	1.98	831	30.0	32.0	0.91
SMT	25.2	597	162.	188.	0.12
AUGUSTUS5	19.0	273	77.8	96.8	1.28
AUGUSTUS7	222.	515	1706.	1928.	1.28

tion in all cases, except AUGUSTUS5, where the results are comparable. We should

TABLE 7
Test results for J-MMD-SAINV preconditioner.

Matrix	P-time	Its	It-time	Tot-time	ρ
BCSSTK13	0.82	349	6.53	7.35	0.39
BCSSTK14	0.46	73	1.07	1.53	0.27
BCSSTK15	0.81	167	5.05	5.86	0.33
BCSSTK16	1.05	98	5.72	6.77	0.12
BCSSTK17	3.13	711	79.8	82.9	0.40
BCSSTK18	1.05	261	15.5	16.5	0.58
BCSSTK21	0.39	191	2.88	3.27	1.51
BCSSTK25	1.99	1512	151.	153.	0.57
S1RMQ4M1	1.26	248	15.0	16.3	0.20
S2RMQ4M1	1.42	528	32.7	34.1	0.25
S3RMQ4M1	1.43	1140	70.3	71.7	0.24
NASA2910	0.95	341	12.7	13.6	0.28
NASA4704	0.91	1176	38.3	39.2	0.62
SMT	10.3	546	148.	159.	0.11
AUGUSTUS5	12.0	281	75.6	87.6	1.10
AUGUSTUS7	220.	516	1551.	1771.	1.06

mention that for the matrices from structural analysis, which are fairly dense, the right-looking version of the SAINV code was found to be faster and it was the one used in the experiments. For the diffusion problems, which are comparatively sparse, the left-looking version is faster and was used for the runs presented here.

The experiments in this section suggest that SAINV is a robust and effective general purpose preconditioner for the conjugate gradient method, especially when used in combination with symmetric scalings and MMD ordering. The standard AINV preconditioner can be used in connection with diagonally compensated reduction, resulting in a preconditioner which is also reliable but not nearly as effective, with the possible exception of diffusion problems. Concerning this approach, we note that it is difficult to predict the performance on the basis of η. Some correlation is present, e.g., in the three shell problems, for which η increases with the difficulty of the problem. However, η is too coarse an indicator of performance to be useful in deciding when to use diagonally compensated reduction.

The SAINV preconditioner is also easy to use. As already mentioned, we have used the standard value $\tau = 10^{-1}$ for the drop tolerance in all the tests with AINV and SAINV. Much better results can be obtained in some cases with a different value of τ, but we deliberately avoided fine-tuning because we wanted to show that this is a good choice in general. Also, the performance of the algorithm is only moderately affected by the choice of τ, provided that it is not chosen too small or too large; see [12].

6. Conclusions. We have developed SAINV, a reliable version of the AINV factorized approximate inverse preconditioner for the conjugate gradient method. The algorithm is applicable to general SPD matrices, is easily parallelized, and performs well on challenging linear systems arising in finite element modeling of elasticity and diffusion problems. The new preconditioner is easy to use, as it does not require the user to specify a sparsity pattern for the approximate inverse but only the value of the drop tolerance τ. Typically, setting $\tau = 10^{-1}$ gives good results. While the setup phase is slightly more expensive than for the standard AINV preconditioner, the cost of computing the preconditioner is still reasonable and the overall algorithm is cost effective.

Our experiments indicate that the performance of SAINV can be significantly improved by applying some preliminary transformations to the matrix A, namely, symmetric diagonal scaling and MMD reordering. With these transformations the storage required for the SAINV preconditioner is often significantly less than that for the coefficient matrix A itself. We stress that the cost of these preprocessings is negligible compared to that of solving the linear system, and we recommend using them as the default options in practice.

We have also investigated the use of diagonally compensated reduction of positive off-diagonal matrix entries as an alternative, inexpensive means to stabilize the AINV preconditioner. While this approach gave good results on some of our test problems, it proved to be generally less effective than SAINV applied to the original matrix.

Acknowledgments. The first author would like to thank Sofia Benzi for postponing her birth until after the bulk of this work was completed. Parts of this paper were written while the third author was a visitor at Los Alamos National Laboratory; the hospitality and support of LANL are greatly appreciated. Thanks also to four anonymous referees for helpful comments.

REFERENCES

[1] M. A. Ajiz and A. Jennings, *A robust incomplete Choleski-conjugate gradient algorithm*, Internat. J. Numer. Methods Engrg., 20 (1984), pp. 949–966.
[2] O. Axelsson, *Iterative Solution Methods*, Cambridge University Press, Cambridge, 1994.
[3] O. Axelsson and L. Yu. Kolotilina, *Diagonally compensated reduction and related preconditioning methods*, Numer. Linear Algebra Appl., 1 (1994), pp. 155–177.
[4] S. T. Barnard, L. M. Bernardo, and H. D. Simon, *An MPI implementation of the SPAI preconditioner on the T3E*, Internat. J. High Perf. Comput. Applic., 13 (1999), pp. 107–123.
[5] M. Benzi, R. Kouhia, and M. Tůma, *An assessment of some preconditioning techniques in shell problems*, Comm. Numer. Methods Engrg., 14 (1998), pp. 897–906.
[6] M. Benzi, J. Marín, and M. Tůma, *A two-level parallel preconditioner based on sparse approximate inverses*, in Iterative Methods in Scientific Computation IV, D. R. Kincaid and A. C. Elster, eds., IMACS Series in Computational and Applied Mathematics, 5, IMACS, New Brunswick, NJ, 1999, pp. 167–178.
[7] M. Benzi, C. D. Meyer, and M. Tůma, *A sparse approximate inverse preconditioner for the conjugate gradient method*, SIAM J. Sci. Comput., 17 (1996), pp. 1135–1149.
[8] M. Benzi and M. Tůma, *A sparse approximate inverse preconditioner for nonsymmetric linear systems,* SIAM J. Sci. Comput., 19 (1998), pp. 968–994.
[9] M. Benzi and M. Tůma, *Numerical experiments with two approximate inverse preconditioners,* BIT, 38 (1998), pp. 234–241.
[10] M. Benzi and M. Tůma, *A comparative study of sparse approximate inverse preconditioners*, Appl. Numer. Math., 30 (1999), pp. 305–340.
[11] M. Benzi and M. Tůma, *Orderings for factorized sparse approximate inverse preconditioners*, SIAM J. Sci. Comput., 21 (2000), pp. 1851–1868.
[12] L. Bergamaschi, G. Pini, and F. Sartoretto, *Approximate inverse preconditioning in the parallel solution of sparse eigenproblems*, Numer. Linear Algebra Appl., 7 (2000), pp. 99–116.
[13] A. Berman and R. J. Plemmons, *Nonnegative Matrices in the Mathematical Sciences*, Academic Press, New York, 1979.
[14] R. Bridson, *Multi-Resolution Approximate Inverses*, M.Sc. thesis, Computer Science Department, Waterloo University, Waterloo, Ontario, Canada, 1999.
[15] R. Bridson and W.-P. Tang, *Ordering, anisotropy, and factored sparse approximate inverses*, SIAM J. Sci. Comput., 21 (1999), pp. 867–882.
[16] E. Chow, *A priori sparsity patterns for parallel sparse approximate inverse preconditioners*, SIAM J. Sci. Comput., 21 (2000), pp. 1804–1822.
[17] E. Chow and Y. Saad, *Experimental study of ILU preconditioners for indefinite matrices*, J. Comput. Appl. Math., 86 (1997), pp. 387–414.
[18] E. Chow and Y. Saad, *Approximate inverse preconditioners via sparse-sparse iterations*, SIAM J. Sci. Comput., 19 (1998), pp. 995–1023.

[19] T. Davis, *University of Florida Sparse Matrix Collection*, http://www.cise.ufl.edu/∼davis/sparse/ (1999).

[20] H. C. Elman, *A stability analysis of incomplete LU factorizations*, Math. Comp., 47 (1986), pp. 191–217.

[21] M. R. Field, *An Efficient Parallel Preconditioner for the Conjugate Gradient Algorithm*, Hitachi Dublin Laboratory Technical Report HDL-TR-97-175, Dublin, Ireland, 1997.

[22] M. R. Field, *Improving the Performance of Factorised Sparse Approximate Inverse Preconditioner*, Hitachi Dublin Laboratory Technical Report HDL-TR-98-199, Dublin, Ireland, 1998.

[23] M. J. Grote and T. Huckle, *Parallel preconditioning with sparse approximate inverses*, SIAM J. Sci. Comput., 18 (1997), pp. 838–853.

[24] N. I. M. Gould and J. A. Scott, *Sparse approximate-inverse preconditioners using norm-minimization techniques*, SIAM J. Sci. Comput., 19 (1998), pp. 605–625.

[25] I. Hladík, M. B. Reed, and G. Swoboda, *Robust preconditioners for linear elasticity FEM analyses*, Internat. J. Numer. Methods Engrg., 40 (1997), pp. 2109–2127.

[26] T. Huckle, *Approximate sparsity patterns for the inverse of a matrix and preconditioning*, Appl. Numer. Math., 30 (1999), pp. 291–303.

[27] I. E. Kaporin, *New convergence results and preconditioning strategies for the conjugate gradient method*, Numer. Linear Algebra Appl., 1 (1994), pp. 179–210.

[28] I. E. Kaporin, *High quality preconditioning of a general symmetric positive definite matrix based on its* $U^TU + U^TR + R^TU$ *decomposition*, Numer. Linear Algebra Appl., 5 (1998), pp. 483–509.

[29] S. A. Kharchenko, L. Yu. Kolotilina, A. A. Nikishin, and A. Yu. Yeremin, *A Reliable AINV-type Preconditioning Method for Constructing Sparse Approximate Inverse Preconditioners in Factored Form*, Moscow University, Moscow, Russia, 1999, preprint.

[30] L. Yu. Kolotilina, A. A. Nikishin, and A. Yu. Yeremin, *Factorized sparse approximate inverse preconditionings.* IV*: Simple approaches to rising efficiency*, Numer. Linear Algebra Appl., 6 (1999), pp. 515–531.

[31] L. Yu. Kolotilina and A. Yu. Yeremin, *Factorized sparse approximate inverse preconditioning.* I. *Theory*, SIAM J. Matrix Anal. Appl., 14 (1993), pp. 45–58.

[32] J. W. H. Liu, *Modification of the minimum degree algorithm by multiple elimination*, ACM Trans. Math. Software, 11 (1985), pp. 141–153.

[33] T. A. Manteuffel, *An incomplete factorization technique for positive definite linear systems*, Math. Comp., 34 (1980), pp. 473–497.

[34] J. A. Meijerink and H. A. van der Vorst, *An iterative solution method for linear systems of which the coefficient matrix is a symmetric* M*-matrix*, Math. Comp., 31 (1977), pp. 148–162.

[35] J. E. Morel, M. L. Hall, and M. J. Shashkov, *A Local Support-Operators Diffusion Discretization Scheme for Hexahedral Meshes*, Report LA-UR-99-4358, Los Alamos National Laboratory, Los Alamos, NM, 1999.

[36] National Institute of Standards, *Matrix Market*, http://math.nist.gov/MatrixMarket (1999).

[37] M. Suariana and K. H. Law, *A robust incomplete factorization based on value and space constraints*, Internat. J. Numer. Methods Engrg., 38 (1995), pp. 1703–1719.

[38] M. Tismenetsky, *A new preconditioning technique for solving large sparse linear systems*, Linear Algebra Appl., 154–156 (1991), pp. 331–353.

SIAM J. SCI. COMPUT.
Vol. 22, No. 4, pp. 1333–1353
© 2000 Society for Industrial and Applied Mathematics

PRECONDITIONING HIGHLY INDEFINITE AND NONSYMMETRIC MATRICES*

MICHELE BENZI†, JOHN C. HAWS‡, AND MIROSLAV TŮMA§

Abstract. Standard preconditioners, like incomplete factorizations, perform well when the coefficient matrix is diagonally dominant, but often fail on general sparse matrices. We experiment with nonsymmetric permutations and scalings aimed at placing large entries on the diagonal in the context of preconditioning for general sparse matrices. The permutations and scalings are those developed by Olschowka and Neumaier [*Linear Algebra Appl.*, 240 (1996), pp. 131–151] and by Duff and Koster [*SIAM J. Matrix Anal. Appl.*, 20 (1999), pp. 889–901; Tech. report Ral-Tr-99-030, Rutherford Appleton Laboratory, Chilton, UK, 1999]. We target highly indefinite, nonsymmetric problems that cause difficulties for preconditioned iterative solvers. Our numerical experiments indicate that the reliability and performance of preconditioned iterative solvers are greatly enhanced by such preprocessing.

Key words. linear systems of equations, sparse matrices, maximum transversal, nonsymmetric permutations, row and column scalings, preconditioned iterative methods, incomplete LU factorization, sparse approximate inverses

AMS subject classifications. Primary, 65F10, 65N22, 65F50; Secondary, 15A06

PII. S1064827599361308

1. Introduction.

1.1. Motivation and focus. We consider the solution of sparse linear systems $Ax = b$, where A is a general sparse $n \times n$ nonsingular matrix, by preconditioned Krylov subspace methods [25], [39]. For a *general* sparse matrix we mean a matrix that has no special properties, such as symmetry, positive definiteness, diagonal dominance, etc. In particular, we focus on matrices that are highly unstructured, nonsymmetric (structurally as well as numerically), and indefinite; i.e., the eigenvalues of A can lie anywhere in the complex plane. Such matrices arise frequently in the simulation of chemical engineering processes, in economic modeling, in management science, in the analysis of circuits and power system networks, and elsewhere. These problems are very different from the ones arising from the numerical solution of elliptic partial differential equations (PDEs) and can cause serious difficulties for standard iterative methods and preconditioners.

There have been a few attempts to use preconditioned Krylov subspace methods in these contexts, but in general the results have been far from satisfactory. For

*Received by the editors September 17, 1999; accepted for publication (in revised form) June 7, 2000; published electronically October 25, 2000. A preliminary version of this paper appeared as Technical Report LA-UR-99-4857, Los Alamos National Laboratory, Los Alamos, NM, June 2000. This work was performed by an employee of the U.S. Government or under U.S. Government contract. The U.S. Government retains a nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or allow others to do so, for U.S. Government purposes. Copyright is owned by SIAM to the extent not limited by these rights.

http://www.siam.org/journals/sisc/22-4/36130.html

†Department of Mathematics and Computer Science, Emory University, Atlanta, GA 30322 (benzi@mathcs.emory.edu). The research of this author was supported in part by Department of Energy grant W-7405-ENG-36 with Los Alamos National Laboratory.

‡Department of Mathematics, North Carolina State University, Raleigh, NC 27695-8205 (jchaws@math.ncsu.edu).

§Institute of Computer Science, Academy of Sciences of the Czech Republic, Pod vodárenskou věží 2, 182 07 Prague 8, Czech Republic (tuma@cs.cas.cz). The research of this author was supported by Grant Agency of the Czech Academy of Sciences grants 2030706 and 2030801.

example, in [11], various incomplete factorization (ILU) preconditioners and iterative solvers were tested on a set of standard problems from chemical engineering. The main conclusions of that study were that such linear systems are difficult to solve with iterative methods (as indicated by the large number of reported failures) and that realizing the potential of iterative solvers will require improvements in the reordering and/or preconditioning schemes. Similar conclusions were reached in [33] for the use of iterative methods in circuit simulations. The use of preconditioned Krylov subspace methods for the solution of sparse linear systems arising in economic modeling has been investigated in [37] and [24]. There the conclusion was that iterative solvers are often superior to direct ones. However, some of the problems could not be solved by iterative methods; see [37].

Preconditioned iterative methods work especially well when the coefficient matrix is, at least to some degree, diagonally dominant. Reliable methods also exist to handle matrices that are symmetric positive definite, or M-matrices. Matrices with these properties arise frequently from the discretization of second-order, elliptic PDEs. Standard preconditioners, such as those based on incomplete factorizations of the coefficient matrix, are usually reliable under these circumstances and typically deliver good rates of convergence. In contrast, such preconditioners are often unstable or may not even be defined when the coefficient matrix has zeros on the main diagonal and/or is highly nonsymmetric (see the discussion in section 2). Furthermore, the presence of many eigenvalues with arbitrary real part (positive, negative, and zero) causes serious difficulties for many Krylov subspace solvers. Matrices of this kind are loosely referred to as *highly indefinite*, regardless of whether or not they are symmetric. Coefficient matrices from chemical engineering, circuits, economics, etc., often exhibit a large number of zero diagonal entries and poor spectral distributions, and they represent a challenge for preconditioned Krylov subspace solvers.

1.2. Contributions of the paper. In [36], Olschowka and Neumaier introduce new permutations and scaling strategies for Gaussian elimination. The goal is to preprocess the coefficient matrix so as to obtain an equivalent system with a matrix that is more diagonally dominant. This preprocessing reduces the need for partial pivoting, thereby speeding up the solution process, and has a beneficial effect on the accuracy of the computed solution. Although the focus in [36] is on dense systems, the sparse case and the case of incomplete factorizations are also briefly discussed. These and other heuristics have been further developed and efficiently implemented by Duff and Koster; see [18] and [19]. Some evidence of the usefulness of these preprocessings in connection with sparse direct solvers and for ILU preconditioning has been provided in [18] and [19]; see also [30]. Our contribution is to carry out a systematic experimental study of the use of these permutation and scaling algorithms in the context of preconditioned iterative methods applied to challenging linear systems. We consider a number of different preconditioners (diagonal, ILU, sparse approximate inverses) and the combined use of nonsymmetric permutations aimed at improving numerical stability with symmetric ones aimed at reducing fill-in in the preconditioner. Our experiments indicate that this preprocessing, and particularly the use of maximum product transversals, enables the stable computation of the preconditioners, resulting in an overall solution strategy that is both reliable and cost effective.

While we do not claim that this approach to preconditioning general sparse matrices will always work, we do hope that the results in this paper will contribute to a reassessment of the role of iterative solvers in areas where these methods had been almost written off as unreliable, such as chemical engineering. We also hope that

one-sided permutations (and related scalings) will find widespread use in the arena of preconditioned iterative solvers for highly indefinite and nonsymmetric linear systems.

The paper is organized as follows. In section 2 we briefly discuss the preconditioners used in the paper. In section 3, which is based on [19], we recall the one-sided permutations and scalings used to preprocess the matrices. The test problems used for the numerical experiments are described in section 4, and the numerical experiments themselves in section 5. Finally, in section 6 we present our conclusions.

2. ILU and approximate inverse preconditioners. In this section we briefly discuss the preconditioners used in the numerical experiments. We focus our attention on ILU-type techniques and on a sparse approximate inverse preconditioner in factorized form, AINV. These are general-purpose, algebraic preconditioners that have been used successfully to solve a wide range of problems, particularly from PDEs. For a detailed treatment of ILU preconditioning we refer the reader to [39]. For a recent survey of sparse approximate inverse preconditioners, see [6].

ILU methods compute sparse approximations to the triangular factors L and U of A. The incomplete factors are obtained by dropping nonzero entries generated in the course of the factorization process (*fill-ins*) according to some rule. Different dropping rules give rise to different ILU preconditioners. When all fill-ins are discarded and only nonzeros in positions corresponding to the nonzero entries of A are retained, the ILU(0) preconditioner [34] is obtained. This preconditioner is easy to implement and inexpensive to compute, but it is often not good enough, particularly for the kind of challenging problems considered in this paper. More powerful preconditioners can be obtained by allowing more fill-in in the incomplete factors, or by dropping fill-ins based on their value rather than position. These techniques include level-of-fills ILU, denoted ILU(k), and dual threshold ILU, denoted ILUT(tol, p). Here $k \geq 1$ is the fill level, $tol \geq 0$ is a drop tolerance, and $p \geq 0$ denotes the number of off-diagonal nonzeros that are retained in each row of the incomplete factor (usually the p largest ones among those nonzeros that are greater than tol in absolute value).

Although fairly robust in practice, ILU preconditioners often fail on general sparse matrices because of instabilities (see below). In fact, the incomplete factors may not even exist. One way to improve their robustness is by incorporating partial (column) pivoting in the incomplete factorization. In the case of ILUT, this leads to a variant, called ILUTP [39], which is sometimes successful when ILUT fails. However, even this approach often fails when applied to general sparse matrices; see [10] and subsection 5.4 below.

There are two types of instability that ILU preconditioners may suffer from. These instabilities have been discussed in detail in [10]; here we give a brief discussion only. Let $\bar{L}$ and $\bar{U}$ denote the incomplete factors, and let

$$R := \bar{L}\bar{U} - A$$

denote the residual matrix. Also, let

$$E := I - A(\bar{L}\bar{U})^{-1}$$

denote the error matrix, assuming that preconditioning is being applied on the right. Notice that $E = R(\bar{L}\bar{U})^{-1}$. Let $\|\cdot\|_F$ denote the Frobenius matrix norm. In the symmetric positive definite case, there is a good correlation between the size of $\|R\|_F$ and the rate of convergence of the preconditioned conjugate gradient method [20]. On the other hand, in the case of general sparse matrices, $\|R\|_F$ alone is not a reliable

indicator of the quality of the preconditioner; instead, both $\|R\|_F$ and $\|E\|_F$ should be taken into account. Note that $\|R\|_F$ can be interpreted as a measure of the *accuracy* of the preconditioner, regarded as an approximation of A, while $\|E\|_F$ gauges the *stability* of the approximation. In general, a large value of $\|R\|_F$ means a poor approximation and hence a poor preconditioner. This can be caused, for instance, by very small pivots encountered in the course of the incomplete factorization, and accounts for a first kind of instability. However, even if $\|R\|_F$ is small, it can happen that $\bar{L}^{-1}$ and/or $\bar{U}^{-1}$ have very large entries. Therefore $\|E\|_F = \|R(\bar{L}\bar{U})^{-1}\|_F$ could be large, in which case the preconditioned matrix will be far from the identity, and the preconditioned iteration will either stagnate or diverge, leading to a second kind of instability. We stress that the instability here is not in the incomplete factorization process, but in the incomplete factors. Unstable (more accurately, ill-conditioned) ILU factors occur frequently for matrices that are far from symmetric and lack diagonal dominance; see [22], [10], [4].

For general sparse matrices, it is frequently the case that both kinds of instability occur simultaneously, with a crippling effect on the quality of the preconditioner. In general, the *complete* LU factorization of A may not even be defined without pivoting, or it may be unstable. This can happen, for example, when there are zero or small entries on the main diagonal. As long as A is nonsingular it has (in exact arithmetic) an LU factorization with pivoting, but this is not true for *incomplete* factorizations [38]. On the other hand, ILU factorizations that are both accurate and stable are possible for diagonally dominant matrices.

Because of these and other limitations of ILU-type preconditioners, alternative preconditioning techniques based on sparse approximate inverses have been intensively developed in the past few years. The AINV algorithm [3], [5], based on an incomplete biconjugation process, has been shown to be one of the most effective techniques in this class. Here the preconditioner is the product of two triangular matrices, which are sparse approximations to the inverses of the L and U factors of A. Sparsity is preserved by using a drop tolerance. The approximate inverse factors are computed directly from A; no knowledge of the factors L or U is needed. Factorized approximate inverse preconditioners share one drawback with ILU-type techniques: The construction phase is guaranteed to be breakdown-free only for special classes of matrices. Sufficient conditions are that A be symmetric positive definite or an H-matrix; see [29], [3], [28], [2]. For general sparse matrices, instabilities due to very small or zero pivots can occur during the construction of the preconditioner, with disastrous effects. This is perfectly analogous to the instability of the first kind for ILU. Notice that, in contrast to ILU, the instability of the second kind—unstable ILU solves—is not an issue here.

Like for ILU, the performance of approximate inverse preconditioners in factorized form is sensitive to the ordering of the matrix. For structurally symmetric (or nearly so) matrices having a stable AINV preconditioner, it was shown in [9] and [7] that symmetric reorderings that reduce fill-in in the inverse factors, like minimum degree or (generalized) nested dissection, can be used to improve the performance of the preconditioner. However, these symmetric reorderings alone are of little use for general sparse matrices, as in this case the AINV preconditioner may not even be defined. As we shall see, the stability and effectiveness of AINV can be dramatically improved by nonsymmetric permutations and scalings aimed at placing large entries on the main diagonal.

3. Overview of algorithms for finding maximum transversals. In this section we give a summary of algorithms for determining permutations of a matrix that place entries of large absolute value on the main diagonal. The codes used to

perform the permutations were taken from MC64, a set of Fortran routines that will be included in a forthcoming release of the Harwell Subroutine Library. Further details on the algorithms and implementations are provided in [19].

We examine three approaches for permuting large entries to the diagonal of matrices. First, we discuss methods for permuting a matrix so that the diagonal contains a maximum number of nonzeros; this method is referred to as finding a *maximum transversal* or *maximum matching*. Second, we discuss a method that permutes a matrix so that the product of the absolute value of the entries on the diagonal is maximized. Third, we discuss a variant of the second method, where the matrix is permuted so that the absolute value of the smallest entry on the diagonal is maximized.

Finally, we include a discussion on how to scale the entries of a matrix so that its diagonal entries are 1 in absolute value, and its off-diagonal entries are all less than or equal to 1 in absolute value.

In our experiments we found that, in general, the best results in terms of both preconditioner fill and convergence rates were obtained when matrices were permuted so as to maximize the product of the absolute value of the entries on the diagonal, and scaled so that the diagonal entries are 1 in absolute value, and off-diagonal entries are all less than or equal to 1 in absolute value. Also note that the timing behaviors mentioned in the discussions of the algorithms below are worst-case scenarios. The implementations in MC64 are efficient and the preprocessing phase is very fast, even for relatively dense matrices.

3.1. Finding a maximum transversal. Let $A = (a_{ij})$ be a general $n \times n$ matrix. Let $\mathcal{M}$ denote a set of at most n ordered index pairs $(i, j), 1 \le i, j \le n$, in which each row index i and each column index j appears at most once. $\mathcal{M}$ is called a *transversal* or *matching*. When $\mathcal{M}$ has maximum cardinality (n for structurally nonsingular matrices), $\mathcal{M}$ is called a *maximum transversal* or *maximum matching*.

In the case where $|\mathcal{M}| = n$, then $\mathcal{M}$ defines an $n \times n$ permutation matrix $Q = (q_{ij})$, where

$$\begin{cases} q_{ji} = 1 & \text{for } (i, j) \in \mathcal{M}, \\ q_{ji} = 0 & \text{otherwise}, \end{cases}$$

and thus AQ and QA are matrices with the transversal entries on the diagonal. Because our code is row oriented, we limit our discussion to row permutations, i.e., permutations of the form QA.

One of the options in MC64 uses the algorithm MC21 implemented by Duff [14], [15]. MC21 is a depth-first search algorithm with look-ahead; for a sparse matrix with τ nonzero entries, the algorithm has worst-case complexity of $\mathcal{O}(n\tau)$ but in practice exhibits $\mathcal{O}(n + \tau)$ behavior.

The use of such a reordering strategy is fundamental as the first step of permuting sparse reducible matrices to block triangular form. We mention that permuting to a zero-free diagonal has been examined before as a reordering strategy for preconditioning; see, for instance, [11] and [5]. Clearly, such a permutation will prove beneficial when, under the original ordering, there are zeros lying on the diagonal. However, in our experiments, we encountered few cases where finding a maximum transversal provided more benefit than finding a maximum product transversal.

3.2. Finding a maximum product transversal. In this subsection, we discuss permuting a matrix so that the product of the absolute value of the entries along

the diagonal is maximized. That is, we look for a permutation σ that maximizes

$$\prod_{i=1}^{n} |a_{\sigma(i)i}|. \tag{3.1}$$

This strategy, combined with the scalings mentioned below, was introduced in [36] for pivoting in dense Gaussian elimination.

First, this maximization problem is translated into a minimization problem. Let $a_i = \max_j |a_{ij}|$ denote the maximum value in row i of the matrix A. Define the matrix $C = (c_{ij})$ by

$$c_{ij} = \begin{cases} \log a_i - \log |a_{ij}| & \text{for } a_{ij} \neq 0, \\ \infty & \text{otherwise.} \end{cases}$$

Then maximizing (3.1) is equivalent to minimizing

$$\sum_{i=1}^{n} c_{\sigma(i)i}. \tag{3.2}$$

Minimizing the sum (3.2) is equivalent to finding a minimum weight perfect matching. In combinatorial optimization, this is known as the bipartite weighted matching problem, or linear assignment problem. MC64 uses a sparse variant of the bipartite weighted matching algorithm introduced in [36]. Fundamental to finding a minimum weight perfect matching is finding a shortest augmenting path, which in turn relies on a sparse variant of Dijkstra's algorithm [13]. For a full $n \times n$ matrix, the assignment problem can be solved in $\mathcal{O}(n^3)$ time; for a sparse matrix, the problem can be solved in $\mathcal{O}(n\tau \log n)$ time.

An important aspect of applying the bipartite weighted matching algorithm to the matrix C is the following result [21]: A perfect matching $\mathcal{M}$ has minimum weight if and only if there exist vectors $u = (u_i)$ and $v = (v_i)$, each of length n, such that

$$\begin{cases} u_i + v_j = c_{ij} & \text{for } (i,j) \in \mathcal{M}, \\ u_i + v_j \leq c_{ij} & \text{for } (i,j) \notin \mathcal{M}. \end{cases} \tag{3.3}$$

These vectors u and v are used in scaling the permuted matrix (see below).

3.3. Algorithm for finding a bottleneck transversal. In this subsection we consider a simple variation of the method discussed in the previous subsection. Here we are interested in finding a permutation of A such that the smallest absolute value on the diagonal is maximized. That is, we look for a permutation σ that maximizes

$$\min_{1 \leq i \leq n} |a_{\sigma(i)i}|, \tag{3.4}$$

and define the matrix $C = (c_{ij})$ by

$$c_{ij} = \begin{cases} a_i - |a_{ij}| & \text{for } a_{ij} \neq 0, \\ \infty & \text{otherwise.} \end{cases}$$

Then maximizing (3.4) is equivalent to minimizing

$$\max_{1 \leq i \leq n} c_{\sigma(i)i}. \tag{3.5}$$

Thus the maximum product transversal algorithm can be applied here, with only minor modifications, such as to replace the sum operation (3.2) by the max operation (3.5).

Note that this method regards only the smallest entry on the diagonal of the permuted matrix. For example, as mentioned in [19], consider a matrix having a row containing only one nonzero entry whose absolute value is the smallest in the matrix. Then the bottleneck transversal algorithm may return a transversal with small values on the diagonal. MC64 takes a somewhat different approach to avoid this problem; see [18], [19] for details.

Scaling the matrix prior to finding a bottleneck transversal also alleviates this problem. Ultimately, though, we note that we found few examples where bottleneck transversal permutations proved superior to maximum product transversal permutations.

3.4. Scaling. It is often beneficial to scale the matrices using the parameters u_i and v_j from (3.3) obtained in the weighted matching algorithm. To this end, define diagonal matrices D_1 and D_2 by

$$\begin{aligned} D_1 &= \operatorname{diag}(d_1^1, d_2^1, \ldots, d_n^1), \quad d_j^1 = \exp(v_j)/a_j, \\ D_2 &= \operatorname{diag}(d_1^2, d_2^2, \ldots, d_n^2), \quad d_i^2 = \exp(u_i). \end{aligned} \tag{3.6}$$

Then QD_1AD_2 is a matrix whose diagonal entries are 1 in absolute value, and whose off-diagonal entries are all less than or equal to 1, in absolute value. Such a matrix is referred to as an I-matrix in [36]. By simply changing the sign of rows (or columns) having a diagonal entry equal to -1 we obtain a matrix with unit diagonal. The eigenvalues of such a matrix lie in discs centered at 1, with radii equal to the sum of the absolute values of the off-diagonal entries in the corresponding row. Therefore, the less the matrix deviates from a diagonally dominant matrix, the more clustered the eigenvalues will be, around 1. In the ideal case, all the discs of such a matrix will have radii less than 1, and the matrix will be strictly rowwise diagonally dominant. In turn, this guarantees that ILU and AINV preconditioners will be well defined. It is also a favorable situation for Krylov subspace methods since, in particular, all the eigenvalues will have positive real part.

4. Description of test problems. In this section we describe the matrices that were used in the numerical experiments. Most of these matrices are available in the public domain [17], [12], [35]. They are representative of problems from a variety of applications, but they have in common the fact that they are difficult to solve with iterative methods. The matrices are listed in Table 1 below, together with some basic information. In Table 9 we report the estimate for the condition number returned by MATLAB (except for the last three problems, which are too large).

The first eight matrices are from chemical engineering and represent simulations of different chemical processes. While they are not large, these matrices are interesting because of the challenge they pose to iterative solvers. They are highly unstructured, very far from being structurally symmetric, with most of the diagonal entries equal to zero. In addition, they are highly indefinite and tend to be very ill conditioned. Matrices similar to these have been used to test ILU-type preconditioners in [11]. In that paper, MC21 was used to obtain a zero-free diagonal: The results were poor.

TABLE 1
Test problem information.

Application area: Chemical engineering			
Matrix	Description	Order	Nonzeros
WEST0655	Sixteen stage column section	655	2854
WEST0989	Seven stage column section	989	3537
WEST1505	Eleven stage column section	1505	5445
WEST2021	Fifteen stage column section	2021	7353
LHR01	Light hydrocarbon recovery	1477	18592
LHR02	Light hydrocarbon recovery	2954	37206
BAYER09	Chemical process simulation	3083	21216
BAYER10	Chemical process simulation	13436	94926
Application area: Economic models and linear programming			
Matrix	Description	Order	Nonzeros
MAHINDAS	Economic model of Victoria	1258	7682
ORANI678	Economic model of Australia	2529	90158
BP200	Simplex method basis matrix	822	3802
GEMAT11	Power flow in 2400 bus system —initial simplex method basis	4929	33185
GEMAT12	Power flow in 2400 bus system —basis after 100 iterations	4929	33111
Application area: Circuit modeling			
Matrix	Description	Order	Nonzeros
WATSON4a	Jacobian at step 4, 1 row added	468	2870
WATSON5a	Jacobian at step 4, 1 row added	1854	10848
CIRCUIT3	Jacobian from nonlinear DAE system	12127	48137
Application area: PDE problems			
Matrix	Description	Order	Nonzeros
SHERMAN2	Thermal simulation with steam injection	1080	23094
LNS3937	Linearized Navier–Stokes equations	3937	25407
UTM5940	Plasma physics, tokamak modeling	5940	83842
SLIDE	Solid deformation model (ALE3D)	20191	1192535
TWO-DOM	Solid deformation model (ALE3D)	22200	1188152
VENKAT25	2D unstructured Euler solver, time step=25	62424	1717792

The next five matrices come from the general area of mathematical economics and management. These matrices are also highly unstructured, nonsymmetric, indefinite with most diagonal entries equal to zero, and cause difficulties for preconditioned iterative methods.

The next three matrices arise in circuit design. The first two problems are described in [33] and are available from the Matrix Market [35]. The original matrices WATSON4 and WATSON5 are rectangular; as in [33], we appended one row at the bottom of these matrices to make them square (and we changed the names to WATSON4a and WATSON5a). The row vector used was $e_n^T = (0, \ldots, 0, 1)$. All diagonal entries are nonzero. The third circuit matrix was kindly provided by Wim Bomhof of Utrecht University; see [8]. This matrix has some zero diagonal entries. All three matrices are very sparse, and they exhibit a good deal of structural symmetry.

Finally, we included six matrices from the discretization of PDEs. Of these, SHERMAN2 does not present any difficulty for ILU preconditioning, but it has been reported by several researchers as being quite challenging for sparse approximate in-

verse preconditioners; see, e.g., [1], [6], [26], and [27]. Matrix LNS3937 has a zero diagonal block corresponding to the divergence constraint in the Navier–Stokes equations and is challenging for both ILU and approximate inverse techniques; see, respectively, [10] and [1]. Zero diagonal blocks induced by constraints also occur in the unstructured finite element matrices SLIDE and TWO-DOM, which were kindly provided by Ivan Otero (Lawrence Livermore National Laboratory). Matrix UTM5940 is fairly difficult to solve with ILU-type methods [10] and is even more so by sparse approximate inverse techniques. Matrix VENKAT25 was included because it is difficult for AINV. These last two matrices have no zero diagonal entries.

These matrices are just a selection from a larger set that was used for the tests; the chosen problems are representative of the results observed. As we will see, preprocessing makes all these problems solvable by iterative methods preconditioned with ILUT and AINV, and many of them even with ILU(0) or ILU(1) preconditioning. We found, however, several problems that could not be solved by the techniques employed in this paper. In these cases, the preconditioned iteration usually converged, but to a seriously inaccurate solution. Among these matrices we mention SHYY41 (also discussed in [10]), NNC666 and NNC1374, GRAHAM1, several of the FIDAP matrices, and some of the LHR0∗c matrices from chemical engineering, all available in [12] or [35]. We tried to solve these problems by a *direct* method, namely, Gaussian elimination with partial pivoting, but again the computed solution was affected by large errors. The same happened when we used the *complete* factors computed with the direct method as preconditioners for Krylov subspace methods—a form of iterative refinement. Not surprisingly, these matrices are severely ill conditioned. One cannot blame an algorithm for being unable to produce accurate solutions to extremely ill conditioned problems. In this paper, we only present results for problems that could be solved with some accuracy.

5. Numerical experiments. In this section we report on numerical experiments aimed at assessing the impact of the MC64 permutation and scaling routines on the robustness and performance of preconditioned Krylov subspace methods. All algorithms were implemented in Fortran77 using double precision arithmetic. The codes were compiled by `f77` with the `-O3` optimization option. Test runs were performed on a Sun Ultra 5 workstation for all test problems except the last three, for which one 250 MHz processor of an SGI Origin 2000 computer was used.

Three different Krylov subspace solvers were tried: BiCGSTAB [41], GMRES [40], and TFQMR [23]. While the three algorithms performed similarly on most problems, BiCGSTAB was found to be somewhat better than the others overall. Therefore, we will present results for BiCGSTAB only. The preconditioners used are diagonal (Jacobi) scaling, ILU(0), ILU(1), ILUT, and AINV. Besides these, we performed experiments also with ILUTP [39] and SPAI [26]. In all cases, right preconditioning was used. An artificial right-hand side b was used in the runs, so that the system $Ax = b$ had the known solution $x = (x_i)$ with $x_i = i$, $1 \le i \le n$. Runs were performed also with other choices of b, with similar results. The initial guess was always the zero vector, $x_0 = 0$. The iteration was stopped when the ℓ_2-norm of the initial residual was reduced by at least eight orders of magnitude, or when a maximum number of iterations maxit $= \min\{n, 2000\}$ was reached.

Some comments on the accuracy of the approximate solutions corresponding to this stopping criterion are in order. As reported in Table 9, many of the test matrices are very ill conditioned, and a small residual does not necessarily guarantee a small error. Nevertheless, the stopping criterion adopted was sufficient to produce

solutions with good relative accuracy except in a few cases (BAYER09, GEMAT12, CIRCUIT3), where some of the components of the solution were affected by large errors. For these cases, reducing the stopping tolerance from 10^{-8} to 10^{-12} resulted in accurate solutions, at the expense of an increase in the number of iterations of roughly 30%. However, all the results presented in the subsequent sections correspond to the stopping tolerance 10^{-8}.

5.1. Testing reliability. Here we present results aimed at assessing the impact of the preprocessing on the reliability of preconditioned BiCGSTAB. Only iteration counts are reported. In Table 2 we report the results of runs using diagonal (Jacobi) preconditioning for the original matrix (under "orig") and for different preprocessings: the basic maximum transversal (under "mc21"), the bottleneck transversal (under "bt"), the maximum product transversal without scalings (under "mpd"), and with scalings (under "mps").

TABLE 2
Iteration count, Jacobi preconditioning.

Matrix	orig.	mc21	bt	mpd	mps
WEST0655	‡	†	†	†	†
WEST0989	‡	†	†	239	171
WEST1505	‡	†	†	536	570
WEST2021	‡	†	†	342	455
LHR01	‡	†	†	421	238
LHR02	‡	†	†	1195	1109
BAYER09	‡	†	†	†	244
BAYER10	‡	†	†	†	†
MAHINDAS	‡	†	†	348	137
ORANI678	‡	†	1133	217	196
BP200	‡	†	†	†	†
GEMAT11	‡	†	†	†	†
GEMAT12	‡	†	*bd*	†	†
WATSON4a	467	467	467	222	183
WATSON5a	†	†	†	†	†
CIRCUIT3	‡	†	†	†	†
SHERMAN2	†	†	†	466	†
LNS3937	‡	†	†	†	†
UTM5940	†	†	†	†	†
SLIDE	‡	†	†	†	†
TWO-DOM	‡	†	†	†	†
VENKAT25	†	†	†	†	†

A "‡" means that the preconditioner was not defined, due to zeros on the main diagonal. A "†" means failure to converge within the maximum number of allowed iterations. A "*bd*" denotes a breakdown in the BiCGSTAB acceleration. Notice that mc21 leaves matrices WATSON4a, WATSON5a, SHERMAN2, UTM5940, and VENKAT25 unchanged, since these matrices have no zero diagonal entries.

The only problem that can be solved without any preprocessing is the small circuit matrix WATSON4a, which requires a number of iterations almost equal to the order of the matrix. While the maximum and bottleneck transversals lead to virtually no improvements, the maximum product transversals result in convergence in nine cases. In particular, six out of the eight chemical engineering problems and both economics problems can be solved using Jacobi preconditioning in connection with

the maximum product transversal and associated scalings. Notice that with mps, Jacobi preconditioning simply has the effect of changing the sign of those matrix columns for which the corresponding diagonal entry is -1.

The next three tables report the results obtained for various ILU-type preconditioners. As is well known, these preconditioners are sensitive to the ordering of the equations and unknowns. Therefore, after applying the various preprocessings, we also reorder the matrix with a symmetric permutation. Consider for instance mps preprocessing. Indicated with D_1 and D_2, respectively, the row and column scalings associated with the maximum product transversal and with Q the corresponding row permutation that permutes the scaled matrix to a zero-free diagonal form, the symmetric permutations are based on the adjacency graph of the symmetric matrix $\hat{A} + \hat{A}^T$, where $\hat{A} = QD_1AD_2$. Once the permutation matrix P has been determined, one solves the following linear system:

$$(P^T Q D_1 A D_2 P)y = P^T Q D_1 b.$$

The solution of the original linear system is then $x = D_2 P y$. The preconditioner calculation is carried out on the scaled and reordered matrix

$$\tilde{A} = P^T Q D_1 A D_2 P.$$

While the purpose of the scalings D_1, D_2 and of the one-sided permutation Q is to improve stability, the main effect of the symmetric permutation P is to increase the effectiveness of the preconditioner by making it more accurate. For structurally symmetric matrices that are numerically unsymmetric and far from diagonally dominant, it was found in [4] that the performance and robustness of ILU-type preconditioners was generally improved by the reverse Cuthill–McKee ordering [16], denoted "rcm" in the tables (see column "SO"). Thus, we used rcm as the default ordering for ILU preconditioners. Although it is not always the best ordering, rcm gave good results in a majority of cases, in good agreement with the conclusions in [4]. Whenever rcm performed poorly we switched to another symmetric ordering, like multiple minimum degree [32] (denoted "mmd") or generalized nested dissection [31] (denoted "gnd"). Occasionally, we found that no symmetric reordering was the best option (denoted "no" in the tables). In these tables, a "‡" indicates a failure due to pivot breakdown; i.e., a zero or exceedingly small pivot was encountered in the course of the incomplete factorization.

The results for ILU(0) preconditioning are reported in Table 3. Again, using just mc21 is ineffective, with the only exception of matrix ORANI678. Results for the bottleneck transversal are not much better. However, the robustness of ILU(0) is greatly improved when the maximum product transversal is used. No pivot breakdown occurs, and all but five problems can be solved. The five failures are due to very slow convergence except for CIRCUIT3, for which the ILU(0) factors are unstable. Using different symmetric reorderings for these problems did not help.

We notice that there are two matrices, SHERMAN2 and VENKAT25, which can be easily solved with ILU(0) preconditioning without any preprocessing. Indeed, using mps results in a slight deterioration in the rate of convergence. Clearly, if a given combination of preconditioner and Krylov subspace solver gives good results, the use of preprocessing is not necessary and should not be used. We further observe that in several cases, mpd (without scalings) gives better results than mps. The same phenomenon occurs also for Jacobi and ILU(1) preconditioning (see Tables 2 and 4).

TABLE 3
Iteration count, ILU(0) preconditioning.

Matrix	SO	orig.	mc21	bt	mpd	mps
WEST0655	gnd	‡	‡	‡	157	144
WEST0989	gnd	‡	‡	‡	51	60
WEST1505	mmd	‡	‡	‡	1498	903
WEST2021	mmd	‡	‡	‡	136	162
LHR01	rcm	‡	†	†	52	49
LHR02	rcm	‡	‡	‡	210	62
BAYER09	rcm	‡	‡	‡	32	42
BAYER10	rcm	‡	‡	‡	†	†
MAHINDAS	rcm	‡	‡	167	34	32
ORANI678	rcm	‡	83	50	28	21
BP200	mmd	‡	‡	126	510	603
GEMAT11	rcm	‡	†	†	153	140
GEMAT12	rcm	‡	†	†	702	838
WATSON4a	rcm	131	131	127	60	89
WATSON5a	rcm	†	†	†	†	†
CIRCUIT3	rcm	‡	‡	†	†	†
SHERMAN2	no	8	8	†	12	11
LNS3937	rcm	‡	†	†	†	†
UTM5940	no	†	†	†	†	†
SLIDE	rcm	‡	†	628	335	335
TWO-DOM	rcm	‡	†	†	419	513
VENKAT25	rcm	90	90	†	88	117

TABLE 4
Iteration count, ILU(1) preconditioning.

Matrix	SO	orig.	mc21	bt	mpd	mps
WEST0655	gnd	‡	‡	‡	‡	†
WEST0989	rcm	‡	‡	‡	32	38
WEST1505	rcm	‡	‡	‡	37	45
WEST2021	rcm	‡	‡	‡	68	74
LHR01	rcm	‡	‡	†	64	58
LHR02	rcm	‡	‡	‡	143	104
BAYER09	rcm	‡	‡	‡	10	14
BAYER10	rcm	‡	‡	‡	1176	1707
MAHINDAS	rcm	‡	‡	108	9	16
ORANI678	rcm	‡	‡	69	14	13
BP200	mmd	‡	58	36	94	†
GEMAT11	rcm	‡	‡	107	43	46
GEMAT12	rcm	‡	‡	†	183	55
WATSON4a	rcm	126	126	92	27	26
WATSON5a	no	†	†	†	†	932
CIRCUIT3	rcm	‡	‡	†	†	†
SHERMAN2	rcm	8	8	†	4	4
LNS3937	gnd	379	*bd*	†	264	355
UTM5940	no	151	151	†	168	119
SLIDE	mmd	‡	‡	†	226	259
TWO-DOM	mmd	‡	‡	†	151	155
VENKAT25	rcm	56	56	†	57	81

In Table 4 we report the results obtained with ILU(1) preconditioning. Again, mc21 and bt offer little benefit, whereas the use of mpd or mps allows all but three problems to be solved. With mpd we had one failure due to pivot breakdown (WEST0655), one due to slow convergence (WATSON5a), and one due to unstable ILU factors (CIRCUIT3). For mps, unstable ILU solves were the cause of all three failures. Notice that in a few cases, ILU(1) performs worse than ILU(0). However, all PDE problems can be solved with ILU(1) when either mpd or mps is used, with mpd giving somewhat better results on average.

TABLE 5
Iteration count, ILUT preconditioning.

Matrix	SO	$tol,\ p$	orig.	mc21	bt	mpd	mps
WEST0655	gnd	$10^{-1}, 5$	‡	‡	‡	65	37
WEST0989	rcm	$10^{-1}, 5$	‡	‡	470	21	9
WEST1505	mmd	$10^{-1}, 5$	‡	‡	†	513	53
WEST2021	rcm	$10^{-1}, 5$	‡	‡	‡	110	34
LHR01	rcm	$10^{-1}, 5$	‡	‡	‡	252	41
LHR02	rcm	$10^{-1}, 5$	‡	‡	‡	‡	78
BAYER09	rcm	$10^{-1}, 5$	‡	‡	†	†	14
BAYER10	mmd	$10^{-4}, 20$	‡	‡	‡	‡	65
MAHINDAS	rcm	$10^{-1}, 5$	‡	†	959	8	9
ORANI678	rcm	$10^{-1}, 5$	‡	†	62	12	14
BP200	mmd	$10^{-1}, 5$	‡	†	28	17	11
GEMAT11	rcm	$10^{-1}, 5$	‡	‡	†	1471	303
GEMAT12	rcm	$10^{-1}, 5$	‡	‡	†	354	306
WATSON4a	rcm	$10^{-1}, 5$	457	457	457	208	31
WATSON5a	rcm	$10^{-5}, 25$	6	6	6	6	6
CIRCUIT3	rcm	$10^{-4}, 20$	80	‡	516	36	90
SHERMAN2	rcm	$10^{-1}, 5$	37	37	†	8	10
LNS3937	rcm	$10^{-3}, 10$	‡	‡	†	776	24
UTM5940	no	$10^{-4}, 20$	164	164	†	302	141
SLIDE	rcm	$10^{-1}, 5$	‡	†	†	†	491
TWO-DOM	rcm	$10^{-1}, 5$	‡	†	†	†	494
VENKAT25	rcm	$10^{-2}, 5$	†	†	‡	†	98

It is already clear from these results that the reliability of preconditioned iterative solvers is greatly enhanced by the use of one-sided permutations aimed at placing large entries on the main diagonal, even when simple preconditioners like ILU(0) and ILU(1) are used. There are, however, a few hard problems for which this approach does not work. Additional robustness can be achieved by using a drop tolerance.

In Table 5 we present results for the popular dual-threshold ILUT(tol, p) preconditioner. Our strategy for the experiments was the following. We used the default parameters $tol = 10^{-1}$, $p = 5$ with rcm as the basic symmetric reordering. Whenever we found that this combination produced poor results, we switched to a different symmetric reordering until a good one was found (in the following order: mmd, gnd, no reordering). When this didn't work we changed the ILUT parameters by decreasing tol and, if necessary, increasing p. We do not claim that this leads to an optimal, or even good preconditioning strategy: Rather, the purpose of these experiments is to demonstrate that iterative solvers can be made reliable without much need for fine-tuning. We emphasize that the values $tol = 10^{-1}$ and $p = 5$ are not typical for ILUT. Usually, a much smaller drop tolerance and a much larger value of p are used,

particularly for hard problems: see, e.g., [10]. In other words, we chose to use a very sparse preconditioner, in many cases even sparser than the original matrix. We made this choice deliberately, with the intent to show that most problems become very easy to solve once the preprocessing phase is applied. In general, better results can be obtained with a different choice of the parameters.

The results in Table 5 show that with mps, all problems can be solved with ILUT preconditioning. In a majority of cases the basic combination of rcm reordering with $tol = 10^{-1}$ and $p = 5$ was sufficient to solve the problem. In some cases, an alternative symmetric reordering and/or additional fill-in had to be used in order to achieve convergence. The hardest problems for ILUT appear to be BAYER10, CIRCUIT3, and UTM5940. Notice that complementing the maximum product transversal with scalings has the effect of improving the reliability and performance of ILUT-preconditioned BiCGSTAB in nearly all cases. The only serious exception to this rule occurs for CIRCUIT3, for which the number of iterations increases from 36 to 90.

TABLE 6
Iteration count, AINV preconditioning.

Matrix	SO	*tol*	orig.	mc21	bt	mpd	mps
WEST0655	no	10^{-1}	‡	‡	‡	†	176
WEST0989	mmd	10^{-1}	‡	‡	‡	141	32
WEST1505	mmd	10^{-2}	‡	‡	†	1693	37
WEST2021	mmd	10^{-2}	‡	‡	†	148	8
LHR01	mmd	10^{-1}	‡	†	‡	251	74
LHR02	mmd	10^{-1}	‡	‡	‡	385	127
BAYER09	mmd	10^{-1}	‡	‡	†	15	8
BAYER10	no	3.5×10^{-2}	‡	‡	‡	781	43
MAHINDAS	mmd	10^{-1}	‡	‡	29	15	7
ORANI678	no	10^{-1}	‡	†	251	10	9
BP200	mmd	5×10^{-2}	‡	‡	20	‡	5
GEMAT11	mmd	10^{-1}	‡	‡	†	802	230
GEMAT12	mmd	10^{-1}	‡	‡	†	†	389
WATSON4a	mmd	10^{-1}	21	21	21	14	20
WATSON5a	mmd	10^{-3}	51	51	51	51	114
CIRCUIT3	no	10^{-2}	†	‡	†	268	43
SHERMAN2	mmd	10^{-1}	†	†	†	50	14
LNS3937	gnd	10^{-1}	‡	†	†	†	112
UTM5940	gnd	10^{-1}	1268	1268	†	1388	406
SLIDE	mmd	10^{-1}	†	†	†	603	306
TWO-DOM	mmd	10^{-1}	†	†	965	491	224
VENKAT25	mmd	10^{-1}	†	†	‡	385	411

In Table 6 we present results for the drop-tolerance-based sparse approximate inverse AINV preconditioner. Like ILU-type preconditioners, AINV is sensitive to the ordering. In [9] and [7] it was shown that mmd is generally a good ordering for AINV, with gnd a close second. On the other hand, rcm cannot be recommended in general. Thus, the default parameters used were mmd reordering and $tol = 10^{-1}$. For a few hard problems, it was necessary to switch to gnd reordering (or no reordering) and to reduce the drop tolerance. However, no effort was made to tune the parameters for optimal performance. As for the ILU preconditioner, failures due to pivot breakdowns are denoted by "‡" and failures due to slow convergence by "†."

TABLE 7
Test results for ILUT preconditioning.

Matrix	NSO-time	SO-time	P-time	ρ	Its	It-time	Tot-time
WEST0655	0.009	0.006	0.005	1.22	37	0.110	0.130
WEST0989	0.011	0.004	0.004	0.967	9	0.056	0.075
WEST1505	0.018	0.017	0.007	0.916	53	0.323	0.365
WEST2021	0.024	0.009	0.011	0.997	34	0.333	0.378
LHR01	0.091	0.019	0.013	0.349	41	0.369	0.494
LHR02	0.214	0.042	0.026	0.346	78	1.54	1.82
BAYER09	0.113	0.025	0.017	0.401	14	0.280	0.440
BAYER10	0.799	0.556	0.189	1.22	65	6.63	8.19
MAHINDAS	0.019	0.009	0.009	0.613	9	0.080	0.117
ORANI678	0.185	0.132	0.077	0.087	14	0.452	0.850
BP200	0.009	0.016	0.004	0.856	11	0.049	0.078
GEMAT11	0.079	0.038	0.038	0.632	303	8.05	8.21
GEMAT12	0.097	0.038	0.039	0.662	306	8.26	8.43
WATSON4a	0.006	0.004	0.004	0.799	31	0.063	0.078
WATSON5a	0.015	0.031	0.031	1.85	6	0.128	0.211
CIRCUIT3	0.127	0.106	0.610	2.02	90	7.08	7.88
SHERMAN2	0.053	0.021	0.020	0.228	10	0.111	0.206
LNS3937	0.131	0.030	0.129	2.59	24	0.973	1.27
UTM5940	0.223	—	1.50	3.13	141	14.5	16.2
SLIDE	0.977	0.938	0.588	0.090	491	67.1	69.6
TWO-DOM	0.976	0.956	0.595	0.100	494	69.5	72.0
VENKAT25	1.43	1.28	2.59	1.05	98	48.5	53.8

As for ILUT, we see that mps preprocessing enables BiCGSTAB preconditioned with AINV to solve all our test problems. For instance, problem SHERMAN2, which is notoriously difficult for approximate inverse methods, becomes very easy to solve. We also see that in most cases scalings improve the performance of the maximum product transversal. More importantly, scalings improve reliability, as shown by the fact that there are four problems that cannot be solved with mpd alone.

5.2. Timing results. From the experiments performed so far it appears that mps preprocessing dramatically increases the reliability and performance of Krylov subspace methods preconditioned with drop-tolerance-based preconditioners, like ILUT and AINV. The question remains to be addressed of how expensive the preprocessing phase is in practice. This is an important question because this preprocessing phase, dependent as it is on the numerical values of the matrix coefficients, is not easy to amortize, except for the situation where a sequence of linear systems with the same coefficient matrix and different right-hand sides must be solved. If the coefficient matrix changes, the preprocessing has to be applied anew.

Timing results relative to ILUT and AINV are presented in Tables 7 and 8, respectively. We report the time to perform the mps preprocessing (under NSO-time), the time for the symmetric reordering (under SO-time), the time for computing the preconditioner (under P-time), the time to perform the iterative solve phase (It-time), and, in the last column, the total solution time (under Tot-time). Timings are in seconds, and were measured with the `dtime` function. We also report the density ρ of the preconditioner, defined as the ratio between the number of nonzeros in the preconditioner over the number of nonzeros in the original coefficient matrix A. The number of iterations to converge (Its) is also included. The parameters and symmetric

TABLE 8
Test results for AINV preconditioning.

Matrix	NSO-time	SO-time	P-time	ρ	Its	It-time	Tot-time
WEST0655	0.009	—	0.122	10.2	176	1.54	1.67
WEST0989	0.011	0.010	0.018	2.36	32	0.175	0.214
WEST1505	0.018	0.017	0.094	5.92	37	0.499	0.628
WEST2021	0.024	0.023	0.203	7.87	8	0.228	0.478
LHR01	0.091	0.108	0.499	2.67	74	1.56	2.26
LHR02	0.214	0.231	1.10	3.00	127	5.94	7.49
BAYER09	0.113	0.128	0.228	1.73	8	0.270	0.739
BAYER10	0.799	—	4.67	5.96	43	11.2	16.7
MAHINDAS	0.019	0.139	0.061	1.27	7	0.082	0.301
ORANI678	0.185	—	0.776	0.146	9	0.332	1.29
BP200	0.009	0.016	0.026	2.46	5	0.046	0.097
GEMAT11	0.079	0.057	0.172	1.62	230	8.57	8.88
GEMAT12	0.097	0.058	0.224	2.04	389	15.8	16.2
WATSON4a	0.006	0.014	0.011	1.01	20	0.049	0.080
WATSON5a	0.015	0.072	0.167	1.77	114	1.34	1.59
CIRCUIT3	0.127	—	3.82	1.89	43	3.43	7.38
SHERMAN2	0.053	0.071	0.094	0.352	14	0.148	0.366
LNS3937	0.131	0.059	0.971	5.89	112	7.05	8.22
UTM5940	0.223	0.158	2.80	2.77	406	45.0	48.2
SLIDE	0.977	1.39	12.0	0.273	306	50.4	64.8
TWO-DOM	0.976	1.68	11.8	0.254	224	36.1	50.6
VENKAT25	1.43	1.40	20.1	0.838	411	184.	207.

reorderings are the same as those used for the runs in Tables 5 and 6.

Concerning the cost of the preprocessing, we see that it is usually small compared with the overall solution costs. For the larger problems the cost of preprocessing, including the time to apply the symmetric reordering, is negligible compared with the total solve time. We also see that in most cases the time for the nonsymmetric permutation and scaling is comparable to the time required for the symmetric permutations, which are based on the (unweighted) graph of the matrix only.

The performance of ILUT preconditioning is impressive: In most cases, rapid convergence is obtained with a very sparse preconditioner. For the few cases where a high number of iterations is required to achieve convergence (GEMAT11, GEMAT12, SLIDE, and TWO-DOM) we found that much faster convergence and smaller timings result if more fill-in is allowed. For instance, using ILUT$(10^{-3}, 10)$ on GEMAT11 results in convergence in 40 iterations with preconditioner density $\rho = 1.46$, and the total solution time becomes 1.81 seconds.

It is our opinion that when used with mps preprocessing, ILUT preconditioning has the potential to become a useful tool for solving linear equations from chemical engineering applications. Because rapid convergence can be achieved with very sparse incomplete factorization preconditioners, this approach may be competitive with sparse direct methods. This approach also has considerable potential for matrices arising from economic modeling and management science, although we have less experience with such problems.

On the other hand, it is unclear whether this approach is really useful for matrices arising in circuit simulations. Sparse direct solvers suffer very little fill-in on such problems when a good ordering is adopted. Hence, for iterative solvers to be

competitive with direct methods they must converge extremely fast with very sparse preconditioners. From our test runs, this seems to be difficult to achieve unless the incomplete factorization closely approaches a complete one. We note that in [8] a combined direct/iterative method is described in which the preconditioned iteration is applied to a relatively small Schur complement matrix. This algorithm appears to be competitive with sparse direct solvers, so there is a role for iterative methods in this area.

In terms of timings and storage requirements, AINV (Table 8) is generally more expensive than ILUT, but it should be kept in mind that an important advantage of AINV, its parallelizability, is not captured by these one-processor experiments. There are a few cases where the density of the preconditioner is rather high. Sparser preconditioners can be obtained by using a larger value of *tol*, but this may slow down convergence considerably. This is especially true for the chemical engineering problems. On the other hand, we found that for the PDE problems faster convergence and smaller overall timings can be obtained by allowing more fill in the preconditioner. The same applies to the GEMAT$*$ problems. Notice the good performance of AINV on the matrices from economics, MAHINDAS and ORANI678. As for the circuit matrices, similar remarks as for ILUT apply. The main point here is that mps preprocessing dramatically increases the reliability of both ILUT and AINV preconditioning, therefore considerably widening the range of applicability of such techniques.

5.3. Further analysis of the results. In order to have a better understanding of the effect of the MC64 permutations and scalings used for preprocessing, we collected the statistics presented in Table 9. Under "condest" we report the condition number (estimated with the MATLAB function `condest`) for the original matrix and for the matrix scaled by the row and column scalings associated with mps preprocessing. The condition number could not be estimated for the three largest matrices.

Under "d.d. rows" we report the number of (weakly) diagonally dominant rows in the original matrix, in the matrix permuted with the maximum product transversal (mpd), and in the matrix scaled and permuted with mps. Under "d.d. cols" similar statistics are reported relative to the number of (weakly) diagonally dominant columns.

The statistics show that the chemical engineering matrices greatly benefit from the permutations and scalings: The number of diagonally dominant rows and columns is greatly increased, and ill conditioning is drastically reduced. Analogous remarks apply to problems MAHINDAS, ORANI678, BP200, and GEMAT11. Hence, it is not surprising that preconditioned iterative methods perform well on these problems when mps preprocessing is used.

For problems GEMAT12 and CIRCUIT3, the scalings have the effect of increasing the condition number. However, for GEMAT12 the number of weakly diagonally dominant rows and columns is greatly increased by the mpd preprocessing. The increase is somewhat smaller when scalings are used, but it is still a significant improvement. This makes the preconditioner computation stable, and therefore the net effect is positive. Matrices WATSON4a and WATSON5a are better conditioned after scaling and have a greater fraction of diagonally dominant rows after mpd. However, the scalings have the effect of reducing the number of diagonally dominant rows as compared with using mpd alone. Perhaps this explains why using mps often gives worse results than using mpd alone for these matrices (see Tables 3, 5, and 6).

TABLE 9
Conditioning and diagonal dominance statistics.

	condest		d.d. rows			d.d. cols		
Matrix	orig.	scaled	orig.	mpd	mps	orig.	mpd	mps
WEST0655	1.6E+12	2.0E+04	4	321	355	3	260	340
WEST0989	5.7E+12	1.9E+04	2	638	666	0	412	641
WEST1505	9.0E+12	2.5E+04	2	957	1004	0	619	969
WEST2021	7.5E+12	2.4E+04	2	1256	1340	0	808	1309
LHR01	5.4E+06	4.4E+04	20	323	438	0	766	710
LHR02	8.2E+06	5.5E+04	40	626	856	0	1532	1409
BAYER09	2.3E+21	1.1E+04	1	1610	1733	1	1737	2168
BAYER10	3.8E+15	1.5E+05	2	4653	7048	0	7081	7501
MAHINDAS	1.0E+13	5.0E+03	2	910	896	0	635	762
ORANI678	1.0E+07	1.1E+06	72	1866	1826	0	1653	1692
BP200	8.9E+08	3.8E+03	0	317	317	1	368	480
GEMAT11	3.7E+08	6.8E+06	2	1508	1536	2	1409	1238
GEMAT12	3.7E+08	1.2E+13	1	1465	1213	1	1398	1298
WATSON4a	8.8E+10	8.9E+07	161	374	270	377	161	261
WATSON5a	5.5E+07	3.2E+06	187	1264	1099	1268	187	332
CIRCUIT3	3.6E+07	3.0E+08	7865	7503	8245	7354	7650	8227
SHERMAN2	1.4E+12	3.3E+03	634	204	292	74	560	460
LNS3937	1.0E+17	1.9E+04	509	1283	689	307	892	701
UTM5940	1.9E+09	7.6E+08	762	915	882	925	766	926
SLIDE	n/a	n/a	1330	1203	1885	1201	1277	1274
TWO-DOM	n/a	n/a	2917	4627	4727	2917	4627	4700
VENKAT25	n/a	n/a	0	0	0	0	0	0

For the matrices arising from PDE problems, the permutations and scalings are generally beneficial. For VENKAT25, it is not clear from the results reported in Table 9 whether the preprocessing does any good. However, we know that ILUT and AINV benefit from the preprocessing. Although no row or column became diagonally dominant after the permutations and scalings, we found that the Gerschgorin bounds on the real and imaginary parts of the eigenvalues were one order of magnitude smaller after mps preprocessing, suggesting that the scaled and permuted matrix is not as far from diagonally dominant and has a more clustered spectrum than the original matrix.

5.4. Experiments with ILUTP and SPAI. We also experimented with the ILUTP preconditioner [39], which is generally more reliable than ILUT. However, we found that ILUTP preconditioning applied to the original matrices, or even to those permuted with mc21, is hardly better than ILUT, unless very large amounts of fill are allowed. Moreover, we found several problems that could not be solved by ILUTP even with very large amounts of fill ($tol = 0$, $p = 30$). These are WEST0655, LHR01, LHR02, BAYER09, BAYER10, BP200, and LNS3937. The failure of ILUTP was due to pivot breakdowns in all cases, except for LNS3937, where it appeared to be due to unstable ILU factors. It is mentioned in [10] that ILUTP with row pivoting (rather than column pivoting) is able to solve LHR01, but it took 134 iterations of GMRES(50) and a rather dense preconditioner. Using ILUT with mps, we can solve LHR01 in 41 iterations with a very sparse preconditioner (see Table 7). On the other hand, none of the techniques used in [10] succeeded in solving problem LNS3937, which is easily solved by ILUT with mps preprocessing. Hence, based on our experiments,

using standard ILUT with mps preprocessing is a more robust approach than using ILUTP alone on the original problem.

When ILUTP is used in combination with mps it gives good results, sometimes better than those obtained with ILUT for the same choice of the parameters. However, the use of column pivoting makes ILUTP slightly more expensive than ILUT, and it is unclear whether ILUTP is worth using with mps preprocessing.

Additional experiments were performed with the sparse approximate inverse preconditioner SPAI [26]. This technique is based on adaptive Frobenius norm minimization. Because the SPAI preconditioner is not factorized, it is insensitive to the ordering of the equations and unknowns. However, it is sensitive to scalings. We tested SPAI on the original matrices and with mps preprocessing. We found that overall, the use of scalings improves the reliability and performance of SPAI. This was especially true for the chemical engineering matrices; none of these matrices could be solved with SPAI, even with generous amounts of fill, but all of them could be solved after applying mps preprocessing. The same happened with MAHINDAS, BP200, SHERMAN2, LNS3937, and UTM5940.

In [1, p. 112], the authors give results for LNS3937 using a parallel implementation of BiCGSTAB preconditioned with SPAI. This took 1942 iterations with an approximate inverse containing 1588045 nonzeros, corresponding to $\rho = 61.3$! The total solution time was 567.3 seconds using 16 processors of a Cray T3E. Using SPAI with mps preprocessing, we can solve LNS3937 in 660 iterations with a preconditioner containing 150270 nonzeros, corresponding to $\rho = 5.9$, for a total solution time of 88.5 seconds on a Sun Ultra 5 workstation. Nevertheless, SPAI is outperformed by ILU(1), ILUT, and AINV used in combination with mps. Furthermore, SPAI failed on GEMAT12 and CIRCUIT3 (with or without mps). Finally, we found that mps had a detrimental effect on the convergence rate obtained with SPAI applied to matrices SLIDE, TWO-DOM, and VENKAT25.

6. Conclusions. The experiments presented in this paper indicate that the reliability and performance of Krylov subspace methods preconditioned with standard incomplete factorizations can be dramatically enhanced by means of nonsymmetric permutations and scalings aimed at placing large entries on the main diagonal. A similar effect is observed for other preconditioning techniques, such as factorized sparse approximate inverses. This preprocessing phase is inexpensive, both in absolute terms and relative to the total solution costs.

Of the heuristics considered in this paper, the maximum product transversal algorithm gave the best results. With this preprocessing, many of the diagonal entries are large relative to the off-diagonal ones, and the matrix is closer to being diagonally dominant. This not only has a stabilizing effect on the computation of the preconditioner, but also results in preconditioners of high quality in terms of convergence rates in many cases. When used in combination with scalings, which in most cases improve the conditioning of the problem, preconditioners based on drop tolerances (like ILUT and AINV) become quite reliable.

Much work remains to be done before preconditioned iterative methods can become a viable alternative to direct methods in areas such as chemical engineering, economics and management, circuit design, etc. Nevertheless, it is now at least conceivable to use iterative methods in such areas, and fair comparisons with direct solvers can be established. In addition to this, our experiments suggest that nonsymmetric permutations and scalings can be useful to improve the performance of iterative solvers even in areas where these are already widely used, such as PDE problems.

Acknowledgments. We would like to thank Iain Duff and Jacko Koster for providing us with the MC64 subroutines. We are indebted to Jane Cullum, Iain Duff, Daniel Szyld, and two anonymous referees for their constructive criticism of an early draft of the paper. This work was performed while the second author was a Graduate Research Assistant at Los Alamos National Laboratory (LANL): The hospitality and support of LANL are greatly appreciated.

REFERENCES

[1] S. T. Barnard, L. M. Bernardo, and H. D. Simon, *An MPI implementation of the SPAI preconditioner on the T3E*, Int. J. High. Perf. Comput. Appl., 13 (1999), pp. 107–123.
[2] M. Benzi, J. K. Cullum, and M. Tůma, *Robust approximate inverse preconditioning for the conjugate gradient method*, SIAM J. Sci. Comput., 22 (2000), pp. 1318–1332.
[3] M. Benzi, C. D. Meyer, and M. Tůma, *A sparse approximate inverse preconditioner for the conjugate gradient method*, SIAM J. Sci. Comput., 17 (1996), pp. 1135–1149.
[4] M. Benzi, D. B. Szyld, and A. van Duin, *Orderings for incomplete factorization preconditioning of nonsymmetric problems*, SIAM J. Sci. Comput., 20 (1999), pp. 1652–1670.
[5] M. Benzi and M. Tůma, *A sparse approximate inverse preconditioner for nonsymmetric linear systems,* SIAM J. Sci. Comput., 19 (1998), pp. 968–994.
[6] M. Benzi and M. Tůma, *A comparative study of sparse approximate inverse preconditioners*, Appl. Numer. Math., 30 (1999), pp. 305–340.
[7] M. Benzi and M. Tůma, *Orderings for factorized sparse approximate inverse preconditioners*, SIAM J. Sci. Comput., 21 (2000), pp. 1851–1868.
[8] W. Bomhof and H. A. van der Vorst, *A parallel linear system solver for circuit simulation problems*, Numer. Linear Algebra Appl., submitted.
[9] R. Bridson and W.-P. Tang, *Ordering, anisotropy and factored sparse approximate inverses*, SIAM J. Sci. Comput., 21 (1999), pp. 867–882.
[10] E. Chow and Y. Saad, *Experimental study of ILU preconditioners for indefinite matrices*, J. Comput. Appl. Math., 86 (1997), pp. 387–414.
[11] H. N. Cofer and M. A. Stadtherr, *Reliability of iterative linear equations solvers in chemical process simulation*, Computers Chem. Engrg., 20 (1996), pp. 1123–1132.
[12] T. Davis, *University of Florida Sparse Matrix Collection*, University of Florida, Gainesville, FL, available online from http://www.cise.ufl.edu/∼davis/sparse/.
[13] E. W. Dijkstra, *A note on two problems in connection with graphs*, Numer. Math., 1 (1959), pp. 269–271.
[14] I. S. Duff, *On algorithms for obtaining a maximum transversal*, ACM Trans. Math. Software, 7 (1981), pp. 315–330.
[15] I. S. Duff, *Algorithm* 575: *Permutations for a zero-free diagonal*, ACM Trans. Math. Software, 7 (1981), pp. 387–390.
[16] I. S. Duff, A. M. Erisman, and J. K. Reid, *Direct Methods for Sparse Matrices*, Clarendon Press, Oxford, UK, 1986.
[17] I. S. Duff, R. G. Grimes, and J. G. Lewis, *The Rutherford–Boeing Sparse Matrix Collection*, Technical Report RAL-TR-97-031, Rutherford Appleton Laboratory, Chilton, UK, 1997.
[18] I. S. Duff and J. Koster, *The design and use of algorithms for permuting large entries to the diagonal of sparse matrices*, SIAM J. Matrix Anal. Appl., 20 (1999), pp. 889–901.
[19] I. S. Duff and J. Koster, *On Algorithms for Permuting Large Entries to the Diagonal of a Sparse Matrix*, Technical Report RAL-TR-99-030, Rutherford Appleton Laboratory, Chilton, UK, 1999.
[20] I. S. Duff and G. Meurant, *The effect of ordering on preconditioned conjugate gradients*, BIT, 29 (1989), pp. 635–657.
[21] J. Edmonds, *Maximum matching and a polyhedron with* 0,1 *vertices*, J. Res. Nat. Bur. Standards Sect. B, 69B (1965), pp. 125–130.
[22] H. C. Elman, *A stability analysis of incomplete LU factorizations*, Math. Comp., 47 (1986), pp. 191–217.
[23] R. W. Freund, *A transpose-free quasi-minimal residual algorithm for non-Hermitian linear systems*, SIAM J. Sci. Comput., 14 (1993), pp. 470–482.
[24] M. Gilli and G. Pauletto, *Krylov methods for solving models with forward-looking variables*, J. Econom. Dynam. Control, 22 (1998), pp. 1275–1289.
[25] A. Greenbaum, *Iterative Methods for Solving Linear Systems*, SIAM, Philadelphia, 1997.
[26] M. J. Grote and T. Huckle, *Parallel preconditioning with sparse approximate inverses*,

SIAM J. Sci. Comput., 18 (1997), pp. 838–853.
[27] N. I. M. Gould and J. A. Scott, *Sparse approximate-inverse preconditioners using norm-minimization techniques*, SIAM J. Sci. Comput., 19 (1998), pp. 605–625.
[28] S. A. Kharchenko, L. Yu. Kolotilina, A. A. Nikishin, and A. Yu. Yeremin, *A Robust AINV-Type Preconditioning Method for Constructing Sparse Approximate Inverse Preconditioners in Factored Form*, preprint, Russian Academy of Sciences, Moscow, 1999.
[29] L. Yu. Kolotilina and A. Yu. Yeremin, *Factorized sparse approximate inverse preconditioning* I. *Theory*, SIAM J. Matrix Anal. Appl., 14 (1993), pp. 45–58.
[30] X. S. Li and J. W. Demmel, *Making sparse Gaussian elimination scalable by static pivoting*, in Proceedings of the SuperComputing'98 Conference, CD-ROM, ACM, New York, 1998.
[31] R. J. Lipton, D. J. Rose, and R. E. Tarjan, *Generalized nested dissection*, SIAM J. Numer. Anal., 16 (1979), pp. 346–358.
[32] J. W. H. Liu, *Modification of the minimum degree algorithm by multiple elimination*, ACM Trans. Math. Software, 11 (1985), pp. 141–153.
[33] W. D. McQuain, C. J. Ribbens, L. T. Watson, and R. C. Melville, *Preconditioned iterative methods for sparse linear algebra problems arising in circuit simulation*, Comput. Math. Appl., 27 (1994), pp. 25–45.
[34] J. A. Meijerink and H. A. van der Vorst, *An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix*, Math. Comp., 31 (1977), pp. 148–162.
[35] *Matrix Market*, National Institute of Standards and Technology, Gaithersburg, MD, available online from http://math.nist.gov/MatrixMarket.
[36] M. Olschowka and A. Neumaier, *A new pivoting strategy for Gaussian elimination*, Linear Algebra Appl., 240 (1996), pp. 131–151.
[37] G. Pauletto, *Solution and Simulation of Macroeconometric Models*, Ph.D. thesis, Department of Econometrics, University of Geneva, Switzerland, 1995.
[38] Y. Saad, *Preconditioning techniques for indefinite and nonsymmetric linear systems*, J. Comput. Appl. Math., 24 (1988), pp. 89–105.
[39] Y. Saad, *Iterative Methods for Sparse Linear Systems*, PWS Publishing, Boston, 1996.
[40] Y. Saad and M. H. Schultz, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 856–869.
[41] H. A. van der Vorst, *BI-CGSTAB: A fast and smoothly converging variant of BI-CG for the solution of nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 13 (1992), pp. 631–644.

SIAM J. SCI. COMPUT.
Vol. 22, No. 4, pp. 1354–1385
© 2000 Society for Industrial and Applied Mathematics

MODE JUMPING IN THE VON KÁRMÁN EQUATIONS*

C.-S. CHIEN†, S.-Y. GONG†, AND Z. MEI‡

Dedicated to Professor Klaus Böhmer on the occasion of his sixtieth birthday.

Abstract. We study numerical approximations of bifurcating solution curves of the von Kármán equations with simply supported and clamped boundary conditions, respectively. Of special interest here is the splitting of a double bifurcation point into two simple bifurcation points, and the tracing of the associated secondary solution branches, which corresponds to the phenomenon of mode jumping in the buckling of a rectangular plate. A continuation-unsymmetric Lanczos algorithm is utilized for curve-tracking. Our numerical results verify the theoretical prediction of Schaeffer and Golubitsky, namely, mode jumping occurs when the boundary is partially clamped, but not if it is merely simply supported.

Key words. von Kármán equations, mode jumping, bifurcation, continuation methods, unsymmetric Lanczos method

AMS subject classifications. 73H05, 35B32, 65N06, 65F10

PII. S1064827596307324

1. Introduction.

We consider a long rectangular plate P in compression. The Airy stress function $f(x, y)$ produced in P and the deformation $w(x, y)$ of P from its flat state are described by the von Kármán equations [4, 5, 6, 7, 14, 15, 24]

$$\begin{array}{rcl} \Delta^2 f \ + \ \frac{1}{2}[w,w] & = & 0 \\ \Delta^2 w \ - \ [w,f] \ + \ \lambda w_{xx} & = & 0 \end{array} \qquad \text{in} \quad \Omega = [0,l] \times [0,1]. \tag{1.1}$$

Here λ is the external load, Δ^2 is the biharmonic operator in the plane, Ω is the shape of the plate, and the bracket operator $[\cdot, \cdot]$ is defined by

$$[u, v] = u_{xx}v_{yy} - 2u_{xy}v_{xy} + u_{yy}v_{xx}.$$

As is well known, a rectangular plate can support a number of different buckled configurations, which may be distinguished by the number of buckles or wave numbers, respectively. A plate has zero deflection along its nodal lines. Bauer, Keller, and Reiss [3] observed that a double bifurcation can be split into two simple bifurcations by perturbation. Let λ_1 and λ_2 with $\lambda_1 < \lambda_2$ be the load corresponding to these two simple bifurcations. Suppose that the load λ is gradually increased from zero. For $\lambda < \lambda_1$ there is no deflection of the plate and we proceed along the trivial solution. If $\lambda_1 < \lambda < \lambda_2$, the plate is deformed and we follow the first bifurcating solution branch. When the load λ is increased far beyond λ_2, the plate jumps suddenly and violently to a new configuration, the number of buckles (or wave numbers) increasing by one. This phenomenon is called "mode jumping." For example, if there are k buckles in the solution branch bifurcating from $(0, \lambda_1)$ and $(k + 1)$ buckles in the next solution

*Received by the editors July 24, 1996; accepted for publication (in revised form) August 27, 1997; published electronically October 25, 2000.

http://www.siam.org/journals/sisc/22-4/30732.html

†Department of Applied Mathematics, National Chung-Hsing University, Taichung, Taiwan 402, Republic of China (cschien@flower.amath.nchu.edu.tw). These authors were supported by the N.S.C of R.O.C. through project NSC 85-2121-M-005-002 C.

‡Department of Mathematics, University of Marburg, 35032 Marburg, F. R. Germany (meizhen@mathematik.uni-marburg.de). This author was supported by the N.S.C of R.O.C. through project NSC 84-2816-M-005-008 L.

branch, and these two branches are connected by another solution curve, then we say that "mode jumping" occurs in this system.

Mode jumping is perhaps one of the most noteworthy features of experimental studies of the postbuckling behavior of plates. Bauer, Keller, and Reiss indicated in [3] that perturbation of a multiple eigenvalue often causes secondary bifurcation, which in turn leads to mode jumping if it connects two adjacent bifurcating solution branches. It was Schaeffer and Golubitsky [24], and then Holder and Schaeffer [15], who showed that mode jumping depends on the boundary conditions of the plate. More precisely, Holder and Schaeffer considered two sets of boundary conditions for w,

(1) simply supported on all four edges;

(2) simply supported on unloaded sides, clamped on loaded ends,

and two sets of boundary conditions for f,

(a) homogeneous Dirichlet conditions $f = \Delta f = 0$;

(b) homogeneous Neumann conditions $f_n = (\Delta f)_n = 0$.

Here and in what follows $f_n = \frac{\partial f}{\partial n}$ represents the partial derivatives of f in the outer normal directions on the boundary $\partial\Omega$. We refer to the four cases as (1a), (1b), etc. Golubitsky, Holder, and Schaeffer found that mode jumping can occur in the cases (2a) and (2b) as a result of unfolding or imperfection of double bifurcations, but not in (1a) or (1b). To pursue a more detailed study of the influence of boundary conditions in the bifurcation scenario of the von Kármán equations, one may consider the following Robin boundary conditions:

$$\begin{array}{rcl} f &=& (1-\alpha)\Delta f + \alpha f_n = 0 \\ w &=& (1-\beta)\Delta w + \beta w_n = 0 \end{array} \qquad \text{on } x = 0 \text{ and } x = l,\ \alpha,\ \beta \in [0,1], \tag{1.2}$$

and

$$f = \Delta f = 0, \quad w = \Delta w = 0 \qquad \text{on } y = 0 \text{ and } y = 1.$$

For $\alpha = \beta = 0$ we obtain simply supported boundary conditions for f and w, while for $\alpha = \beta = 1$ we have the partially clamped boundary conditions for both f and w. We remark here that the Robin-type boundary conditions with the homotopy parameters $\alpha,\ \beta$ is a natural extension of classical boundary conditions, aiming for more accurate approximations of the real physical systems; see, e.g., [26, 21], and the further references cited therein. While in the context of mixed finite element schemes, the totally clamped boundary conditions

$$f = f_n = 0, \qquad w = w_n = 0 \quad \text{on } \partial\Omega$$

are imposed normally; see, e.g., [5, 6, 7].

Our aim here is to study mode jumping of a plate through numerical approximations, in particular, continuation methods [1, 2, 18]. According to the experiments of Stein [25] and the results of Schaeffer and Golubitsky [24], and Holder and Schaeffer [15], varying the length l of the plate P and the load λ can make the first bifurcation double. On the other hand, numerical discretizations and perturbations of the domain split the double bifurcation into two simple bifurcations. Moreover, perturbation of the domain is unavoidable if the length l of the domain is an irrational number. Hence, we will follow the solution curves branching from the two simple bifurcations, investigate secondary bifurcation points along the two solution curves, and calculate the secondary solution branch. If the secondary bifurcation points on

each of the two primary solution curves is connected by a secondary solution branch, we say that mode jumping occurs in this system. Otherwise there is no mode jumping at this double bifurcation. Our numerical results are consistent with the prediction of Holder and Schaeffer [15] and Schaeffer and Golubitsky [24].

This paper is organized as follows. In section 2 we follow the discussion in [24] and show how to derive the eigenfunctions of the linearized von Kármán equation. Section 3 describes some properties of the central difference discretization of the von Kármán equations. A continuation-unsymmetric Lanczos algorithm for bifurcation problems is proposed in section 4. Here the unsymmetric Lanczos method is used to solve the linear systems associated to the discrete von Kármán equations, to detect bifurcation points along the solution curves, and to compute the Ritz vector at the bifurcation point for branch-switching. Numerical results are presented in section 5.

2. Some basic results for the linearized problems. In this section we will show how double bifurcations of the von Kármán equations are generated for a rectangular domain $\Omega = [0, l] \times [0, 1]$, where the length l is considered as the second bifurcation parameter in the problem. We will restrict the discussion to the simply supported and partially clamped boundary conditions, respectively, as in [15, 24]. The more general Robin boundary conditions will be discussed elsewhere.

2.1. Simply supported boundary conditions. Let us consider the linearized von Kármán equation with simply supported boundary conditions

$$\begin{aligned} \Delta^2 w \ + \ \lambda w_{xx} = 0 \qquad & \text{in} \quad \Omega = [0, l] \times [0.1], \\ w \ = \ \Delta w = 0 \qquad & \text{on} \quad \partial\Omega. \end{aligned} \tag{2.1}$$

It has nontrivial solutions

$$w = w_{m,n}(x, y) = \sin \frac{m\pi x}{l} \cdot \sin n\pi y, \quad m, n = 1, 2, \ldots, \tag{2.2}$$

if and only if

$$\lambda = \lambda_{m,n} = \left(\frac{\pi}{l}\right)^2 \left(m + \frac{n^2 l^2}{m}\right)^2, \quad m, \ n = 1, 2, \ldots. \tag{2.3}$$

Here m, n are called the wave numbers of $w_{m,n}(x, y)$. For fixed length l elementary calculations show that the smallest eigenvalue

$$\min_{m,n} \lambda_{m,n} = \min_{m} \lambda_{m,1} = \pi^2 \min_{m} \left(\frac{m}{l} + \frac{l}{m}\right)^2$$

is achieved at one of the integers $m = [l], \ [l] + 1$. If we consider λ as a function of the length l in (2.3), then any two simple eigenvalues of (2.1) can be coupled to generate a double eigenvalue by varying m or n. For example, the equality $\lambda_{m,n} = \lambda_{m+1,n}$ holds if and only if

$$l = \frac{\sqrt{m(m+1)}}{n}. \tag{2.4}$$

Generally, if we require $\lambda_{m,n} = \lambda_{m+k,n}$ for some $k \in \mathbf{N}$, then the length l should be modified as

$$l = \frac{\sqrt{m(m+k)}}{n}. \tag{2.5}$$

For $m = n = 1$ and $k = 1,\ 3$, we obtain $l = \sqrt{2}$ and $l = 2$, respectively. Figure 1 shows how the eigenvalues of (2.1) vary with respect to the length l, where the $\lambda_{1,1}$-eigenpath intersects the $\lambda_{2,1}$-, $\lambda_{3,1}$-, and $\lambda_{4,1}$-eigenpath at $l = \sqrt{2}$, $\sqrt{3}$, and 2, respectively. For large m we have

$$l = m + \frac{k}{2} + \mathbf{O}(m^{-1}), \quad m \gg k.$$

Taking this into account and substituting (2.5) into (2.3), we obtain

$$\lambda_{m,n} = n^2\pi^2 \left(\sqrt{\frac{m}{m+k}} + \sqrt{\frac{m+k}{m}} \right)^2.$$

Therefore, the smallest eigenvalue is double and

$$\begin{aligned} \min_{m,n,k} \lambda_{m,n} &= \min_{m,k} \lambda_{m,1} \\ &= \pi^2 \min_{m} \left(\sqrt{\frac{m}{m+1}} + \sqrt{\frac{m+1}{m}} \right)^2 \\ &= 4\pi^2 + \mathbf{O}\left(\frac{1}{m}\right). \end{aligned}$$

The corresponding eigenfunctions are

$$\begin{aligned} w_{m,1}(x,y) &= \sin \frac{m\pi x}{l} \cdot \sin \pi y, \\ w_{m+1,1}(x,y) &= \sin \frac{(m+1)\pi x}{l} \cdot \sin \pi y. \end{aligned}$$

In particular, for $l = \sqrt{2}$ the smallest eigenvalue of (2.1) is double and occurs at

$$\lambda_{1,1} = \lambda_{2,1} = \frac{9}{2}\pi^2.$$

2.2. Partially clamped boundary conditions. The linearized von Kármán equation with partially clamped boundary conditions is

$$\begin{aligned} \Delta^2 w \ + \ \lambda w_{xx} = 0 \qquad & \text{in} \quad \Omega = [0,l] \times [0,1], \\ w \ = \ w_n = 0 \qquad & \text{on} \quad x = 0 \text{ and } x = l, \\ w \ = \ \Delta w = 0 \qquad & \text{on} \quad y = 0 \text{ and } y = 1. \end{aligned} \tag{2.6}$$

Following the discussion in [24] we consider separation of variables, i.e., let

$$w(x,y) = u(x)\ v(y) \neq 0.$$

From (2.6) we derive

$$\frac{u^{(4)} + \lambda u''}{u} + 2\frac{u''v''}{uv} + \frac{v^{(4)}}{v} = 0 \tag{2.7}$$

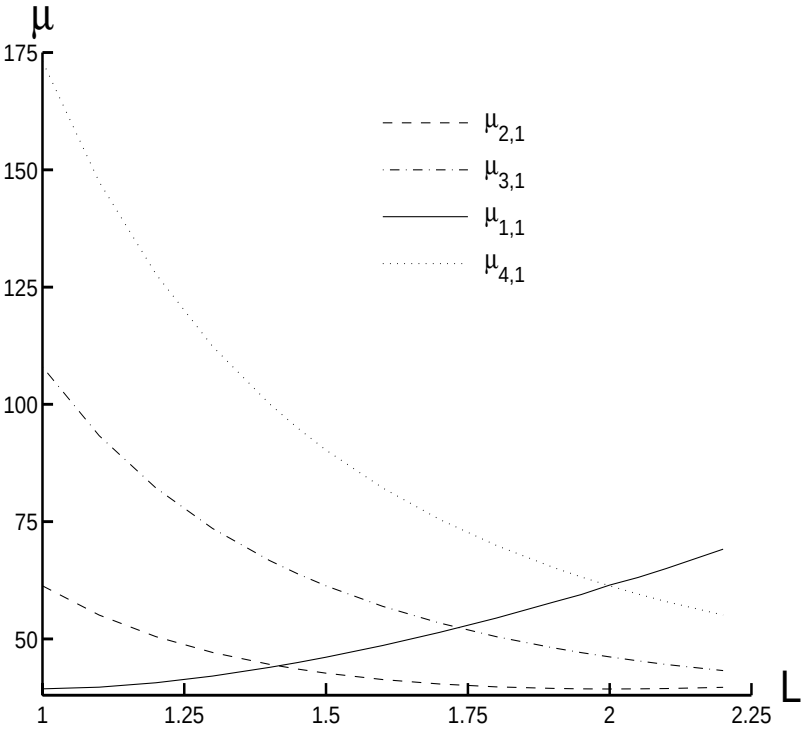


FIG. 1. *Eigenvalues of the von Kármán equations as functions of the parameter l in the domain* $\Omega := [0, l] \times [0, 1]$.

with the boundary conditions

$$u(0) = u(l) = u'(0) = u'(l) = 0, \tag{2.8a}$$

$$v(0) = v(1) = v''(0) = v''(1) = 0. \tag{2.8b}$$

The conditions in (2.8b) are satisfied by functions of the form

$$v(y) = \sin n\pi y, \quad n = 1, 2, \ldots.$$

Note that for any $n \in \mathbf{N}$ the subspace $\{u(x) \sin n\pi y;\ u(x) \in C^{4,s}[0, 1]\}$ is invariant under the operator $\Delta^2 + \lambda \frac{\partial^2}{\partial x^2}$. Thus in this subspace (2.7) reduces to

$$u^{(4)} + (\lambda - 2\ n^2\pi^2)u'' + n^4\pi^4 u = 0. \tag{2.9}$$

Its characteristic equation is

$$a^4 + [\lambda - 2\ (n\pi)^2]a^2 + (n\pi)^4 = 0$$

and has four solutions

$$a_1 := \left\{ (n\pi)^2 - \frac{\lambda}{2} + \left[\lambda \left(\frac{\lambda}{4} - n^2\pi^2 \right) \right]^{\frac{1}{2}} \right\}^{\frac{1}{2}}, \quad a_2 := -a_1,$$

$$a_3 := \left\{ (n\pi)^2 - \frac{\lambda}{2} - \left[\lambda \left(\frac{\lambda}{4} - n^2\pi^2 \right) \right]^{\frac{1}{2}} \right\}^{\frac{1}{2}}, \quad a_4 := -a_3. \tag{2.10}$$

The general solution of (2.9) is of the form

$$u(x) = \sum_{i=1}^{4} c_i e^{a_i x}, \tag{2.11}$$

where the constants $c_i \in \mathbf{C}$ will be determined by the boundary conditions (2.8a). We are interested in real solutions of (2.9). To this end, we consider the following three cases in (2.10): (a) $\frac{\lambda}{4} - (n\pi)^2 < 0$; (b) $\frac{\lambda}{4} - (n\pi)^2 = 0$; (c) $\frac{\lambda}{4} - (n\pi)^2 > 0$. In the case (a) there is no real solution for λ, while (b) implies that $u \equiv 0$, the trivial solution. For $\frac{\lambda}{4} > (n\pi)^2$ the solutions of (2.9) have the form

$$u(x) = c_1 \cos \omega_1 x + c_2 \sin \omega_1 x + c_3 \cos \omega_2 x + c_4 \sin \omega_2 x,$$

where

$$\begin{aligned} \omega_1 &= \sqrt{\alpha_1 - \alpha_2} > 0, \qquad \omega_2 = \sqrt{\alpha_1 + \alpha_2} > 0, \\ \alpha_1 &:= \frac{\lambda}{2} - (n\pi)^2, \qquad \alpha_2 := \left[\lambda \left(\frac{\lambda}{4} - n^2\pi^2 \right) \right]^{\frac{1}{2}}. \end{aligned}$$

By elementary calculations we obtain

$$\omega_1^2 + \omega_2^2 = \lambda - 2(n\pi)^2, \qquad \omega_2^2 - \omega_1^2 = \sqrt{\lambda^2 - 4\lambda\,(n\pi)^2}. \tag{2.12}$$

Elimination of λ in these equations yields

$$\omega_1 \omega_2 = (n\pi)^2.$$

The boundary conditions $u(0) = 0$ and $u'(0) = 0$ imply that $c_3 = -c_1$ and $c_4 = -c_2\omega_1/\omega_2$ in (2.11). Thus,

$$\begin{aligned} u(x) &= c_1(\cos \omega_1 x - \cos \omega_2 x) + c_2(\omega_2 \sin \omega_1 x - \omega_1 \sin \omega_2 x) \\ &:= c_1 \phi(x) + c_2 \psi(x), \end{aligned}$$

where

$$\phi(x) = \cos \omega_1 x - \cos \omega_2 x, \quad \psi(x) = \omega_2 \sin \omega_1 x - \omega_1 \sin \omega_2 x. \tag{2.13}$$

Since we are interested in double eigenvalues, both $\phi(x)$ and $\psi(x)$ can be chosen as eigenfunctions of (2.9). Note that $\psi'(x) = \omega_1\omega_2\phi(x)$. Therefore, the boundary conditions (2.8a) are equivalent to

$$\phi(l) = \phi'(l) = \psi(l) = 0, \tag{2.14}$$

which implies that

$$\omega_1 = \frac{k}{l}\pi, \quad \omega_2 = \frac{k+2m}{l}\pi$$

for some positive integers k and m. Together with (2.12) we derive

$$l = \frac{[k(k+2m)\,]^{\frac{1}{2}}}{n}, \qquad \lambda = 4n^2\pi^2 \frac{(k+m)^2}{k(k+2m)}. \tag{2.15}$$

The first double bifurcation corresponds to $m = 1$ in (2.15) and

$$l = \sqrt{k(k+2)}. \tag{2.16}$$

The eigenfunctions for (2.6) are

$$w_1(x,y) = \left[\cos\left(\frac{k}{l}\pi x\right) - \cos\left(\frac{k+2}{l}\pi x\right) \right] \sin n\pi y,$$

$$w_2(x,y) = \pi\left[\frac{k+2}{l}\sin\left(\frac{k}{l}\pi x\right) - \frac{k}{l}\sin\left(\frac{k+2}{l}\pi x\right) \right] \sin n\pi y. \tag{2.17}$$

The structure of w_1, w_2 indicates that the concept of wave numbers for (2.2) is actually no more suitable for (2.17), though it is used frequently in the literature.

In our numerical experiments we assume that $k = 1$ and $n = 1$ in (2.16) and (2.17), respectively. Then $l = \sqrt{3}$ and a mode jumping is observed; see section 5.

3. Discretization for the von Kármán equations.

3.1. Linearized problem with simply supported boundary conditions. We discretize (2.1) by the standard thirteen-point centered difference approximations with uniform meshsize $h = 1/(N+1)$ on the x- and y-axis, respectively. We have to assume that l is divisible by h, say, $l/h = L+1$ for some positive integer. The domain Ω is covered by a net of discrete points S, where

$$S = \{(x_i, y_j) : x_i = ih,\ y_j = jh,\ i = 0, 1, \ldots, L, L+1,\ j = 0, 1, \ldots, N, N+1\}.$$

At the discrete points the function $w(x,y)$ is to be approximated by a net function $W(x_i, y_j) := W_{i,j}$. If l is an irrational number, the domain must be perturbed in the numerical computations.

Let D, A, $B \in \mathbf{R}^{LN\times LN}$ be the discretization matrices corresponding to the differential operators $\frac{\partial^2}{\partial x^2}$, Δ, and Δ^2 defined in $[0,l]\times[0,1]$, respectively. Then

$$D = \text{diag}\,(A_L, \ldots, A_L) \tag{3.1}$$

with

$$A_L = \frac{1}{h^2}\begin{pmatrix} -2 & 1 & & & 0 \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & 1 \\ 0 & & & 1 & -2 \end{pmatrix} \in \mathbf{R}^{L\times L} \quad \text{and}$$

$$A = \frac{1}{h^2}\begin{pmatrix} \bar{A}_L & I_L & & & 0 \\ I_L & \bar{A}_L & \ddots & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & I_L \\ 0 & & & I_L & \bar{A}_L \end{pmatrix} \tag{3.2}$$

with

$$\bar{A}_L = A_L - 2I_L$$

and

$$B = \frac{1}{h^4}\begin{pmatrix} B_{L_1} & C_L & I_L & & & & 0 \\ C_L & B_{L_2} & C_L & I_L & & & \\ I_L & C_L & B_{L_3} & T_L & I_L & & \\ & \ddots & \ddots & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & \ddots & \ddots & I_L \\ & & & \ddots & \ddots & B_{L_{N-1}} & C_L \\ 0 & & & & I_L & C_L & B_{L_N} \end{pmatrix} \in \mathbf{R}^{LN\times LN} \tag{3.3}$$

with

$$B_{L_1} = B_{L_N} = \begin{pmatrix} 18 & -8 & 1 & & & & 0 \\ -8 & 19 & -8 & 1 & & & \\ 1 & -8 & 19 & -8 & 1 & & \\ & \ddots & \ddots & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & \ddots & \ddots & 1 \\ & & & \ddots & \ddots & 19 & -8 \\ 0 & & & & 1 & -8 & 18 \end{pmatrix} \in \mathbf{R}^{L\times L},$$

$$B_{L_j} = \begin{pmatrix} 19 & -8 & 1 & & & & 0 \\ -8 & 20 & -8 & 1 & & & \\ 1 & -8 & 20 & -8 & 1 & & \\ & \ddots & \ddots & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & \ddots & \ddots & 1 \\ & & & \ddots & \ddots & 20 & -8 \\ 0 & & & & 1 & -8 & 19 \end{pmatrix} \in \mathbf{R}^{L\times L},\ j = 2, 3, \ldots, N-1,$$

and

$$C_L = \begin{pmatrix} -8 & 2 & & & 0 \\ 2 & -8 & 2 & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & 2 \\ 0 & & & 2 & -8 \end{pmatrix} \in \mathbf{R}^{L\times L}.$$

It follows from [10] that $A^2 = B$. The centered difference analogue of (2.1) is

$$H(W, \lambda) = BW + \lambda DW = 0, \tag{3.4}$$

where B and D are defined as above. Let $(\mu^h_{p,q}, U_{p,q})$ be an eigenpair of A. We have (see [17])

$$\begin{aligned} \mu^h_{p,q} &= -4\left[\frac{(L+1)^2}{l^2}\sin^2\left(\frac{\pi}{2}\frac{p}{L+1}\right) + (N+1)^2\sin^2\left(\frac{\pi}{2}\frac{q}{N+1}\right)\right] \\ &:= \mu^h_p + \nu^h_q, \quad 1 \le p \le L, \quad 1 \le q \le N; \end{aligned} \tag{3.5}$$

(3.6)

$$U_{p,q}(x_i, y_j) = U_p(x_i) \cdot U_q(y_j) = \left(\sin\left(\frac{ip\pi}{L+1} \right) \cdot \sin\left(\frac{jq\pi}{N+1} \right) \right), \quad 1 \le i,\ p \le L, \quad 1 \le j,\ q \le N.$$

Here (μ_p^h, U_p) and (ν_q^h, U_q) are the eigenpairs of the discretization matrices A_L and A_N corresponding to $-\frac{d^2}{dx^2}$ defined on $[0, l]$ and $[0, 1]$, respectively, with homogeneous Dirichlet boundary conditions. Actually, we have $U_{p,q} = U_q \otimes U_p$, where $\otimes$ denotes the matrix tensor product; see subsection 3.3.

The result in [10] can be generalized as follows.

THEOREM 3.1. *The eigenpairs of* (3.4) *are* $(\lambda_{p,q}^h, U_{p,q})$ *with*

$$\lambda_{p,q}^h = -\frac{(\mu_{p,q}^h)^2}{\mu_p^h}, \quad p = 1, 2, \ldots, L,\ q = 1, 2, \ldots, N; \tag{3.7}$$

$$U_{p,q}(x_i, y_j) = U_p(x_i) \cdot U_q(y_j), \quad 1 \le i \le L,\ 1 \le j \le N. \tag{3.8}$$

Proof. Let $(\lambda_{p,q}^h, U_{p,q})$ be any eigenpairs of (3.4). Since $B = A^2$, we have

$$\begin{aligned} A^2 U_{p,q} &= -\lambda_{p,q}^h D U_{p,q} \\ &= -\lambda_{p,q}^h D(U_q \otimes U_p) \ \ \text{(see [10])} \\ &= -\lambda_{p,q}^h \mu_p^h (U_q \otimes U_p). \end{aligned}$$

On the other hand,

$$A^2 U_{p,q} = (\mu_{p,q}^h)^2 U_{p,q} = (\mu_{p,q}^h)^2 (U_q \otimes U_p).$$

Hence,

$$\lambda_{p,q}^h = -\frac{(\mu_{p,q}^h)^2}{\mu_p^h}. \qquad \square$$

In general, a double bifurcation of (2.1) will be split into two simple bifurcations by using central difference discretizations. Detailed discussions for second order differential equations can be found in [20]. We are interested in how to maintain the bifurcation scenario of the von Kármán equations at a double bifurcation point in the discrete problems. To this end, the first step is to preserve the double bifurcation of the continuous problem. Let

$$\bar{\mu}_p^h = -4(L+1)^2 \sin^2\left(\frac{\pi}{2} \frac{p}{L+1} \right), \tag{3.9}$$

then $\mu_p^h = l^{-2} \bar{\mu}_p^h$ and (3.7) can be expressed as

$$\begin{aligned} \lambda_{p,q}^h &= -\frac{(l^{-2}\bar{\mu}_p^h + \nu_q^h)^2}{l^{-2} \cdot \bar{\mu}_p^h} \\ &= -\left[\ l^{-2} \cdot \bar{\mu}_p^h + 2\nu_q^h + l^2 \frac{(\nu_q^h)^2}{\bar{\mu}_p^h} \ \right]. \end{aligned}$$

In particular, the equality

$$\lambda_{p,1}^h = \lambda_{p+1,1}^h$$

holds if and only if

$$l = l^h = \left[\frac{(\bar{\mu}_p^h \cdot \bar{\mu}_{p+1}^h)}{(\nu_1^h)^2} \right]^{\frac{1}{4}}. \tag{3.10}$$

For example, if we choose $l = \sqrt{2} \approx 1.4142136$, then the double bifurcation is preserved in the discrete problem by setting $l^h = 1.4126064$ for $h = 0.1$.

In order to study mode jumping of the von Kármán equations, we take the length l as the second bifurcation parameter and make the coordinate transformation $(x, y) \leftrightarrow (l\tilde{x}, y)$. The equation (2.1) can be rewritten as

$$\begin{aligned} \left(\frac{1}{l^2}\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right)^2 w \ + \ \lambda \frac{1}{l^2}\frac{\partial^2 w}{\partial x^2} = 0 \qquad & \text{in} \quad \tilde{\Omega} := [0,1]^2, \\ w \ = \ \left(\frac{1}{l^2}\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right) w = 0 \qquad & \text{on} \quad \partial\tilde{\Omega}. \end{aligned} \tag{3.11}$$

Note that (3.11) has the same eigenpairs as (2.1). We discretize (3.11) with the thirteen-point centered difference approximations as before. Let

$$\tilde{S} := \{(\tilde{x}_i, y_j) \ : \ \tilde{x}_i = ih, \ y_j = jh, \ i, j = 1, 2, \ldots, N\}$$

be a net of discrete points which covers the domain $\tilde{\Omega} = [0,1]^2$, $\tilde{D} = \mathrm{diag}(A_N, \ldots, A_N)$ and $\tilde{E} \in \mathbf{R}^{N^2 \times N^2}$ be the discretization matrices corresponding $\frac{\partial^2}{\partial x^2}$ and $\frac{\partial^2}{\partial y^2}$ defined in $\tilde{\Omega} = [0,1]^2$ with homogeneous Dirichlet boundary conditions. Here $A_N \in \mathbf{R}^{N \times N}$ has the same form as A_L defined in (3.1), and $\tilde{E}$ is given by (see [10])

$$\tilde{E} = \begin{pmatrix} -2I_N & I_N & & & 0 \\ I_N & -2I_N & \ddots & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & I_N \\ 0 & & & I_N & -2I_N \end{pmatrix}. \tag{3.12}$$

We have $\tilde{D} = I_N \otimes A_N$ and $\tilde{E} = A_N \otimes I_N$. The centered difference analogue of (3.11) corresponds to the fact that we choose the same number of discrete points in $\Omega = [0, \ell] \times [0, 1]$. If $\ell \neq 1$, the meshsizes in the x- and y-axis are different.

LEMMA 3.2. *The matrices $\tilde{D}$, $\tilde{E} \in \mathbf{R}^{N^2 \times N^2}$, defined as above, are similar to each other.*

Proof. It is easy to verify that the eigenpairs of $\tilde{D}$ are $(\mu_p^h, U_p \otimes e_i)$, while the eigenpairs of $\tilde{E}$ are $(\mu_p^h, e_i \otimes U_p)$, $p = 1, 2, \ldots, N$. Here e_i is the ith standard unit vector in $\mathbf{R}^N$, the multiplicity of μ_p^h is N, and μ_p^h and U_p are defined as in (3.5) and (3.6), respectively. This completes the proof. □

The centered difference analogue of (3.11) is

$$\left(\frac{1}{l^2}\tilde{D} + \tilde{E}\right)^2 W + \lambda \frac{1}{l^2}\tilde{D}W = 0. \tag{3.13}$$

Let $(\tilde{\mu}^h_{p,q}, U_{p,q})$ be any eigenpair of (3.13), $1 \le p,\ q \le N$. We have

$$\left(\frac{1}{l^2}\tilde{D} + \tilde{E}\right)^2 U_{p,q} = -\tilde{\mu}^h_{p,q}\frac{1}{l^2}\tilde{D}U_{p,q},$$

or

$$\left(\frac{1}{l^2}\mu^h_p + \nu^h_q\right)^2 U_{p,q} = -\tilde{\mu}^h_{p,q}\frac{1}{l^2}\mu^h_p U_{p,q},$$

where μ^h_p, ν^h_q, and $U_{p,q}$ are defined as in (3.5) and (3.6), respectively, $1 \le p,\ q \le N$. This implies

$$\tilde{\mu}^h_{p,q} = -\frac{(l^{-2}\mu^h_p + \nu^h_q)^2}{l^{-2}\mu^h_p}. \tag{3.14}$$

Concluding the discussions above, we have the following.

THEOREM 3.3. *The eigenpairs of* (3.13) *are* $(\tilde{\mu}^h_{p,q}, U_{p,q})$, $p,\ q = 1, \ldots, N$.

In comparison to (3.10), the first eigenvalue of (3.13) is double if and only if

$$l = l^h = \left[\ \frac{(\mu^h_p \cdot \mu^h_{p+1})}{(\nu^h_1)^2}\ \right]^{\frac{1}{4}}. \tag{3.15}$$

3.2. Linearized problem with partially clamped boundary conditions. For convenience we rewrite (2.6) as

$$\begin{aligned}
\left(\frac{1}{l^2}\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right)^2 w \ + \ \lambda\frac{1}{l^2}\frac{\partial^2 w}{\partial x^2} = 0 \qquad & \text{in} \quad \tilde{\Omega} = [0,1]^2,\\
w \ = \ -\frac{1}{l}\frac{\partial}{\partial x}w = 0 \qquad & \text{on} \quad x = 0,\\
w \ = \ \frac{1}{l}\frac{\partial}{\partial x}w = 0 \qquad & \text{on} \quad x = 1,\\
w \ = \ \Delta w = 0 \qquad & \text{on} \quad y = 0 \text{ and } y = 1.
\end{aligned} \tag{3.16}$$

Let $\bar{D} = \text{diag}\,(\bar{A}_N, \ldots, \bar{A}_N) \in \mathbf{R}^{N^2 \times N^2}$ be the discretization matrix corresponding to $\frac{\partial^2}{\partial x^2}$ defined in (3.16), where

$$\bar{A}_N = \frac{1}{h^2}\begin{pmatrix} -2 & 2 & & & 0 \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \\ 0 & & & 2 & -2 \end{pmatrix} \in \mathbf{R}^{N\times N}, \quad h = \frac{1}{(N+1)}. \tag{3.17}$$

It is well known that the eigenpairs of $\bar{A}_N$ are

$$\begin{aligned}
&\nu_p = -2(N+1)^2\left(1 - \cos\frac{p\pi}{N-1}\right), \qquad p = 0, 1, \ldots, N-1,\\
&\bar{U}_p(x_i) = \cos\frac{ip\pi}{N-1}, \qquad 1 \le i \le N, \quad p = 0, 1, \ldots, N-1.
\end{aligned} \tag{3.18}$$

Thus, the centered difference analogue of (3.16) is

$$\left(\frac{1}{l^2}\bar{D}+\tilde{E}\right)^2 W+\lambda\frac{1}{l^2}\bar{D}W=0, \tag{3.19}$$

where $\tilde{E}$ is defined as in (3.12).

It seems that an explicit formula for the eigenpairs of (3.16) is no more available.

3.3. Discretization for the von Kármán equations. We recall the following definition given in [10].

DEFINITION 3.4. *For any $x=(x_1,x_2,\ldots,x_N)^T$, $y=(y_1,y_2,\ldots,y_N)^T\in\mathbf{R}^N$, the vector product $z=x*y\in\mathbf{R}^N$ of x and y is defined by $z_i=x_iy_i$, $i=1,\ldots,N$. Similarly, for any $A\in\mathbf{R}^{N\times N}$ with a_i^T as the ith row, we define the convolution of A and x as*

$$A*x=\begin{pmatrix} x_1a_1^T \\ \vdots \\ x_Na_N^T \end{pmatrix}\in\mathbf{R}^{N\times N}.$$

For $\Omega=[0,l]\times[0,1]$ we will discretize the von Kármán equations (1.1) with simply supported boundary conditions. Let $D\in\mathbf{R}^{LN\times LN}$ be defined as in (3.1), which can be expressed as $D=I_N\otimes A_L$. Similarly, let $E\in\mathbf{R}^{LN\times LN}$ be the discretization matrix corresponding to $\frac{\partial^2}{\partial y^2}$. It has the same form as $\tilde{E}$ defined in (3.12), which can be written as $E=A_L\otimes I_N$. The discretization matrix associated to $\frac{\partial^2}{\partial x\partial y}$ is

$$V=\begin{pmatrix} 0 & V_L & & & 0 \\ -V_L & 0 & V_L & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & V_L \\ 0 & & & -V_L & 0 \end{pmatrix}\in\mathbf{R}^{LN\times LN}, \tag{3.20}$$

where

$$V_L=\frac{1}{h^2}\begin{pmatrix} 0 & 1 & & & 0 \\ -1 & 0 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & 1 \\ 0 & & & -1 & 0 \end{pmatrix}\in\mathbf{R}^{L\times L}.$$

We have $V=V_N\otimes V_L$.

With $Z=(F,W)^T\in\mathbf{R}^{LN}$, the discrete nonlinear system of equations associated to (1.1) with simply supported boundary conditions is

$$H(Z,\lambda)=(H_1(Z,\lambda),H_2(Z,\lambda))^T=0 \tag{3.21}$$

with

$$H_1(Z,\lambda):=BF+(DW)*(EW)-\frac{1}{16}(VW)*(VW),$$

$$H_2(Z,\lambda):=BW-(DW)*(EF)-(EW)*(DF)+\frac{1}{8}(VW)*(VF)+\lambda DW,$$

where B is defined as in (3.3), and D, E, V are defined as above. The Jacobian matrix corresponding to (3.21) is

$$DH = (D_Z H,\ D_\lambda H) = \begin{pmatrix} B & M & 0 \\ & & \\ -M & \tilde{B} & DW \end{pmatrix} \in \mathbf{R}^{2LN\times(2LN+1)}, \tag{3.22}$$

where

$$M := D*(EW) + E*(DW) - \frac{1}{8}V*(VW),$$
$$\tilde{B} := B - D*(EF) - E*(DF) + \frac{1}{8}V*(VF) + \lambda D.$$

LEMMA 3.5. *The discrete nonlinear system* (3.21) *has the same bifurcation points as the discrete linearized von Kármán equation* (3.4) *along the trivial solution* $F = W = 0,\ \lambda \in \mathbf{R}$.

Proof. On the trivial solution of (3.21) we have $F = W = 0$. Thus, the Jacobian matrix DH in (3.22) reduces to

$$DH = \begin{bmatrix} B & 0 & 0 \\ & & \\ 0 & B+\lambda D & 0 \end{bmatrix}. \tag{3.23}$$

Since B is symmetric and positive definite, the singularity of $B + \lambda D$, which is just the coefficient matrix in (3.4), reveals rank-deficiency of DH, and in turn, bifurcation point of (3.21). □

In [5, 6] Brezzi, Rappaz, and Raviart have discussed error estimates for the mixed finite element approximations of the von Kármán equations. In the context of centered difference approximations one may adapt the techniques in [17] to give error analysis for the discrete solution branches of (3.21); see [10] for details.

4. Linear solvers and branch-switching techniques. Recently various conjugate gradient methods have been exploited as linear and nonlinear solvers for continuation problems; see, e.g., [9, 11, 12, 16]. Specifically, in [11] the symmetric Lanczos method was used to solve linear systems and to detect simple bifurcation points by solving the associated linear eigenvalue problems, while the band Lanczos method with reorthoganalization [22] was utilized to handle multiple bifurcations. If the discretization matrices associated to the continuation problem are nonsymmetric, the continuation-Arnoldi algorithm proposed in [12] can handle bifurcation problems as well.

It is well known that the unsymmetric Lanczos method [19] is unstable for solving unsymmetric eigenvalue problems [13]. Furthermore, a serious breakdown of this method could happen. In spite of this, the unsymmetric Lanczos algorithm can effectively handle many unsymmetric bifurcation problems for the following reasons. First, it is an efficient solver for the unsymmetric linear systems in the context of continuation methods. Next, it can reveal bifurcation points along the solution curve. Finally, it can be exploited to compute the tangent vector at a bifurcation point for branch-switching.

Let us consider the unsymmetric linear system $Ax = b$, where $A \in \mathbf{R}^{N\times N}$ is nonsingular and $b \in \mathbf{R}^N$. For $j = 1, 2, \ldots, N$ the unsymmetric Lanczos method

generates two biorthogonal systems $V_j = [v_1, v_2, \ldots, v_j]$ and $W_j = [w_1, w_2, \ldots, w_j]$ such that $W_j A V_j = T_j$ is tridiagonal. Let x_0 be the initial guess and $r_0 = b - Ax_0$ the initial residual. We seek an approximate solution x_j by requiring

$$(b - Ax_j, w_i) = 0, \quad i = 1, 2, \ldots, j, \tag{4.1}$$

and

$$x_j = x_0 + z_j, \tag{4.2}$$

where $z_j = V_j y_j \in \text{span}\,\{v_1, v_2, \ldots, v_j\} =: K_j$ for some $y_j \in \mathbf{R}^j$ and $(\cdot, \cdot)$ denotes the inner product in $\mathbf{R}^N$. It follows from (4.1) that

$$(r_0 - Az_j, w_i) = 0, \quad i = 1, 2, \ldots, j. \tag{4.3}$$

Assume that $\det(T_j) \neq 0$. Then (4.3) implies that

$$W_j^T (r_0 - Az_j) = 0. \tag{4.4}$$

Replacing z_j by $V_j y_j$ in (4.4) yields an equation for y_j,

$$T_j y_j = W_j^T r_0, \tag{4.5}$$

where

$$T_j = W_j^T A V_j = \begin{pmatrix} \alpha_1 & \beta_2 & & & 0 \\ \delta_2 & \alpha_2 & \ddots & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & \beta_j \\ 0 & & & \delta_j & \alpha_j \end{pmatrix}.$$

The entries α_j, β_j, and δ_j will be determined later. If we set $w_1 = v_1 = r_0/\|r_0\|_2$ and $\beta = \|r_0\|_2$, then it follows from the biorthogonality that the equalities

$$W_j^T r_0 = \beta W_j^T w_1 = \beta e_1$$

hold, where $e_1 = [1, 0, \ldots, 0]^T \in \mathbf{R}^j$ is the first standard unit vector. Thereafter, via (4.5) the approximate solution x_j in (4.2) is calculated as

$$x_j = x_0 + V_j y_j = x_0 + \beta V_j T_j^{-1} e_1. \tag{4.6}$$

The unsymmetric Lanczos algorithm for solving the linear system $Ax = b$ is summarized as follows.

ALGORITHM 4.1. *Lanczos algorithm for solving unsymmetric linear system* $Ax = b$.

1. *Input*

$$x_0 = v_0 = w_0 = 0; \quad r_0 = b; \quad v_1 = w_1 = r_0/\beta \ \text{ with } \ \beta = \|r_0\|_2; \ \ \beta_1 = \delta_1 = 0.$$

2. *For* $i = 1, 2, \ldots, j$ *do*

$$\begin{aligned} \alpha_i &= (Av_i, w_i); \\ \hat{v}_{i+1} &= Av_i - \alpha_i v_i - \beta_i v_{i-1}; \\ \hat{w}_{i+1} &= A^T w_i - \alpha_i w_i - \delta_i w_{i-1}; \end{aligned}$$

furthermore, choose $\delta_{i+1} = |(\hat{v}_{i+1}, \hat{w}_{i+1})|^{\frac{1}{2}}$; $\beta_{i+1} = \delta_{i+1}$ sign $(\hat{v}_{i+1}, \hat{w}_{i+1})$; *and define*

$$v_{i+1} = \frac{\hat{v}_{i+1}}{\delta_{i+1}}; \quad w_{i+1} = \frac{\hat{w}_{i+1}}{\beta_{i+1}}.$$

3. *Form the approximate solution via*

$$x_j = x_0 + \beta V_j T_j^{-1} e_1 \tag{4.7}$$

and check the accuracy by

$$\rho_j := \|b - Ax_j\| = \|\hat{v}_{j+1}\| \cdot |e_j^T y_j|. \tag{4.8}$$

If $\rho_j < \varepsilon$, *the given error tolerance, then stop; otherwise set* $j = j + 1$ *and go back to* 2.

The approximate solution in (4.7) is easily obtained by solving the tridiagonal linear system

$$T_j y = \beta e_1 \tag{4.9}$$

via Crout's algorithm [13].

In continuation problems such as the von Kármán equations we are concerned with the bifurcation scenario at the first few bifurcation points. The discretization matrix A associated to the continuation problem with arclength as the curve parameter becomes singular at the bifurcation points. In particular, 0 is an extremal eigenvalue of A when it is evaluated at the first bifurcation point.

Suppose that at the kth continuation step we have obtained the approximating solution y_k. Let T_j be the tridiagonal matrix generated by Algorithm 4.1 and $\theta_1, \ldots, \theta_j$ be the eigenvalues of T_j such that $|\theta_1| < |\theta_2| < \cdots < |\theta_j|$. If $|\theta_1| < \varepsilon^*$ for some $\varepsilon^* > 0$ which is small enough, then a bifurcation point y^* is signaled on the solution curve, and we may use y_k as a rough approximation of y^*. Next, we compute the eigenvector v corresponding to the minimum eigenvalue θ_1, and choose $u = V_j v$ as an approximation to the tangent vector at the bifurcation point. As a summary we obtain the following continuation-unsymmetric Lanczos algorithm.

ALGORITHM 4.2. *A continuation-unsymmetric Lanczos algorithm.*

Input :

$y \in \mathbf{R}^{N+1}$ *such that* $H(y) = 0$; {*an approximate point on* $H^{-1}(0)$}
$\delta > 0$; {*initial stepsize*}
$u > 0$; {*initial tangent vector*}
$\varepsilon^* > 0$; {*tolerance of the zero eigenvalue*}

Step 1. $v := y + \delta u$ {*predictor step*}

Step 2. (i) *Use Algorithm* 4.1 *to solve the linear systems and then calculate y;* {*Newton corrector*}
(ii) *Adapt stepsize;* {*stepsize control*}

Step 3. *Test for bifurcation point and compute tangent vector:*
(i) *Compute the minimum eigenvalue* θ_1 *of* T_j.
(ii) *If* $|\theta_1| < \varepsilon^*$, *then*
compute the Ritz vector u_1 *for branch-switching;*
else
compute tangent vector by the predictor-corrector continuation methods;
goto Step 1.

until traversing is stopped.

Remarks. 1. It is possible to accelerate the rate of convergence of the unsymmetric Lanczos method by using the preconditioning techniques. Here we describe only a robust unsymmetric Lanczos algorithm.

2. From the computational point of view it seems to us that Algorithm 4.2 is superior to the continuation-Arnoldi algorithm if the unsymmetric Lanczos method does not break down. Since in Algorithm 4.2 one deals with tridiagonal matrices, while the matrices generated by the Arnoldi method are of the upper Hessenberg form.

3. In Step 3 the minimum eigenvalue of T_j can be replaced by the minimum singular value or the condition number of T_j for detecting bifurcation points along the solution branch. Our numerical experience shows that the minimum singular value of T_j and the associated singular vector gives in many cases even more accurate approximations of the bifurcation point and null vectors of the linearized operator; see section 5 and [11] for details.

At last we discuss how local perturbation may be incorporated in the context of continuation methods for branch-switching. Suppose that $y^* = (z^*, \lambda^*)$ is a detected bifurcation point on the curve $c \subset H^{-1}(0)$. Let $V \subset \mathbf{R}^{N+1}$ be a small neighborhood of y^* and $f : \mathbf{R}^{N+1} \to \mathbf{R}$ be a smooth mapping such that $f(y) = 0$ for $y = (z, \lambda) \notin V$ and $f(y) > 0$ for $y \in V$. Instead of solving $H(y) = 0$, we solve

$$H_d(y) = H(y) + f(y)d$$

in the neighborhood V of y^*; see [1, 10] for details and for other branch-switching techniques.

5. Numerical results. We use the numerical methods described in the previous sections to investigate mode jumping of the von Kármán equations. All computations were performed on a VAX 9210 machine with double precision arithmetic at National Chung-Hsing University.

Example 1. We consider the von Kármán equations defined in $\Omega = [0, \sqrt{2}] \times [0, 1]$ with simply supported boundary conditions. The first bifurcation point is double and located at $(0, \lambda_{1,1}) = (0, \lambda_{2,1}) = (0, \frac{9}{2}\pi^2)$.

To study whether mode jumping occurs in this example, we perform local perturbation and domain perturbation simultaneously with $l = 1.3,\ 1.4$, and 1.5, respectively. Figure 2 depicts the solution curves at the first two simple bifurcation points for $l = 1.3$, while Figures 3–4 show the first three solution curves for $l = 1.4,\ 1.5$, respectively. Specifically, one sees in Figures 3–4 that the double bifurcation of the continuous problem is split into two simple bifurcations, and the solution curves bifurcating from $(0, \mu_{1,1})$ and $(0, \mu_{3,1})$ connects each other. That is, the former may be viewed as the secondary solution branch of the latter, and vice versa. Note that there is no other secondary bifurcation point on the solution curve branching from $(0, \mu_{1,1})$. Figures 5–8 show how the contours of the solution curve branching from $(0, \mu_{1,1})$ vary with respect to different μ, and the wave number changes gradually from one to three. Figure 9 depicts the contour of the solution curve branching from $(0, \mu_{2,1})$ at $\mu = 55.4358$.

Example 2. We repeat the numerical experiments in [10], i.e., the von Kármán equations defined in $\Omega = [0, 2] \times [0, 1]$ and examine the double bifurcation at $(0, \lambda_{1,1}) = (0, \lambda_{4,1}) = (0, \frac{25}{4}\pi^2)$.

First, we follow the solution curve branching from $(0, \mu_{4,1})$ by choosing d with $\|d\|_\infty = 5 \cdot 10^{-5}$. Figure 10 shows that this solution curve has a secondary bifurcation at $(0, 88.3397)$. Figures 11–14 show the contours of this solution curve at different

values of μ. We observe variation of nodal lines of this solution curve with respect to μ. In particular, Figure 14 shows that the contour at $\lambda \approx 72.769$ already has the same nodal line as the solution curve bifurcating at $(0, \mu_{2,1})$. On the other hand, if we follow the solution curve bifurcating at $(0, \mu_{2,1})$, where $\| d \|_\infty = 5 \cdot 10^{-5}$, we find that this solution curve coincides with the secondary solution branch of the solution curve branching from $(0, \mu_{4,1})$. Thus the solution can be regarded as the secondary solution branch of the solution curve branching from $(0, \mu_{2,1})$, and vice versa.

Next, we follow the solution curve bifurcating at $(0, \mu_{1,1})$ by choosing d with $\|d\|_\infty = 5 \cdot 10^{-6}$. We observe that the the solution curves branching from $(0, \mu_{1,1})$ and $(0, \mu_{5,1})$ approach each other and meet at the secondary bifurcation point $(0, 96.5475)$; see Figure 10. The contours of this solution curve at different values of μ are shown in Figures 15–18. Figure 19 shows the solution curves bifurcating at $(0, \mu_{1,1})$ and $(0, \mu_{4,1})$, respectively, where the domain Ω is perturbed to $\Omega_1 = [0, 1.9] \times [0, 1]$. We remark here that part of the solution curves given in Figures 10 and 19 have the same shapes as the one given in [10].

Example 3. We consider the von Kármán equations with partially clamped boundary conditions,

$$
\begin{aligned}
\Delta^2 f \ + \ \frac{1}{2}[w, w] &= 0, \\
\Delta^2 w \ - \ [w, f] \ + \ \lambda w_{xx} &= 0 \qquad \text{in} \quad \Omega = [0, \sqrt{3}] \times [0, 1], \\
w \ = \ w_n &= 0 \qquad \text{on} \quad x = 0 \text{ and } x = \sqrt{3}, \\
w \ = \ \Delta w &= 0 \qquad \text{on} \quad y = 0 \text{ and } y = 1, \\
f \ = \ \Delta f &= 0 \qquad \text{on} \quad \partial\Omega;
\end{aligned} \tag{5.1}
$$

see section 2. To investigate mode jumping of (5.1), we perform local perturbation and domain perturbation with $l = 1.6,\ 1.7$, and 1.8, respectively. The first bifurcation is double and is split into two simple bifurcations via perturbations.

Figures 20–22 show the first two solution curves of (5.1) with $l = 1.6,\ 1.7$, and 1.8, respectively. Here W_1 and W_2 are the discrete solutions of (5.1). In Figure 20 we observe that a secondary bifurcation point is detected at $\mu \approx 59.3393$ and $\mu \approx 57.4876$ on the W_2- and W_1-solution branches, respectively. These two secondary bifurcation points are connected by a secondary solution branch, which corresponds to the phenomenon of mode jumping in this system. Figure 23 shows the contour of the W_1-solution branch at $\mu \approx 57.4876$, while Figure 24 depicts the contour of the secondary solution branch at $\lambda \approx 58.0849$. Similarly, Figures 25–27 show the contours of the W_2-solution branches at $\mu \approx 59.3373,\ 53.2187$, and 82.3688, respectively. We remark here that the contour of the W_2-solution branch varies with respect to μ, while the contour of the W_1-solution branch is invariant with respect to μ.

For $l = 1.7$ and $l = 1.8$ the contours of the W_1- and W_2-solution branches are similar to those for $l = 1.6$.

Example 4. Choosing $\Omega := [0, 1] \times [0, 1]$, we discretize the von Kármán equations with simply supported boundary conditions by the central difference approximation. We use the uniform meshsize $h = 0.0625$ on the x- and y-axis, respectively. First, we take Bi-CGSTAB and the preconditioned Bi-CGSTAB in [27] to trace the solution curve bifurcating from $(0, \mu_{1,1})$, where the diagonal of the discretization matrix is used as the preconditioner. Next, Algorithm 4.2 is implemented to trace the same

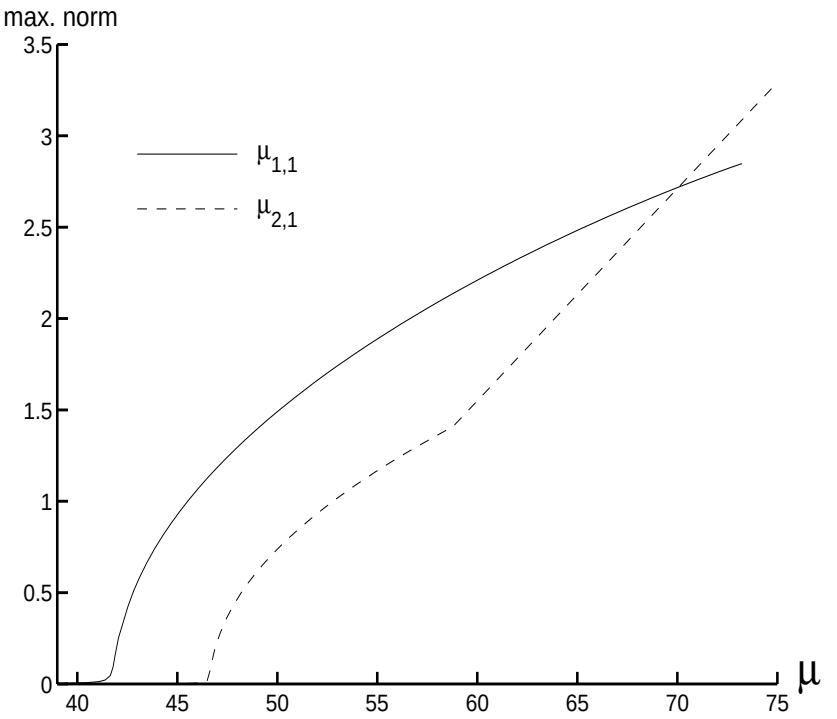


FIG. 2. *The first two solution curves of the von Kármán equations, Example* 1, $l = 1.3$.

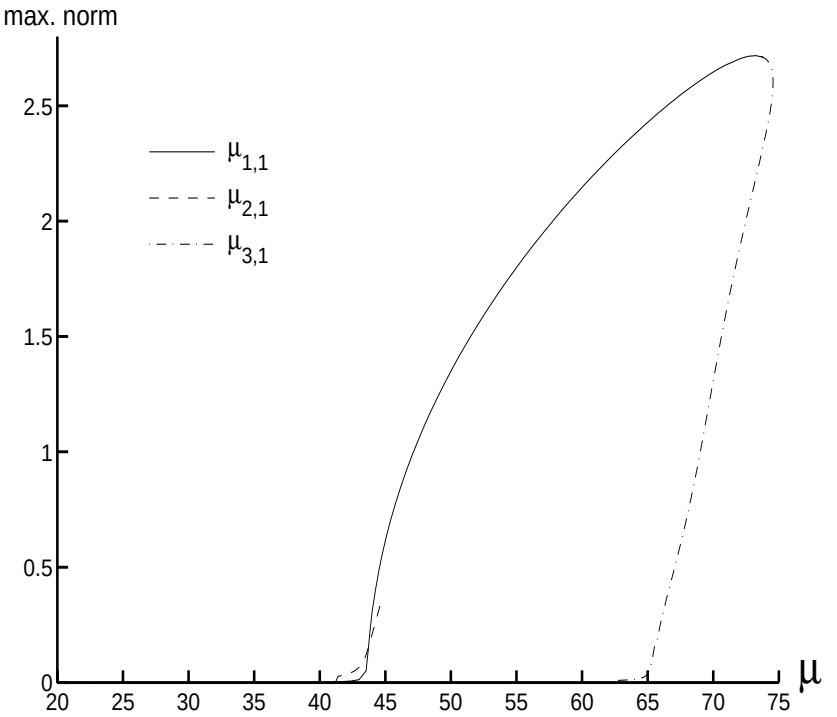


FIG. 3. *The first three solution curves of the von Kármán equations, Example* 1, $l = 1.4$.

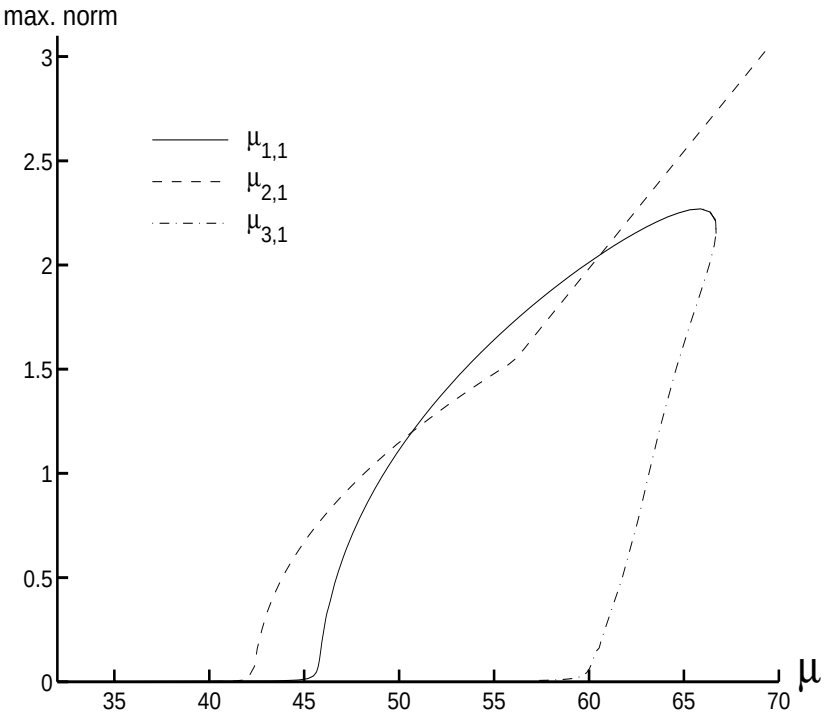


FIG. 4. *The first three solution curves of the von Kármán equations, Example* 1, $l = 1.5$.

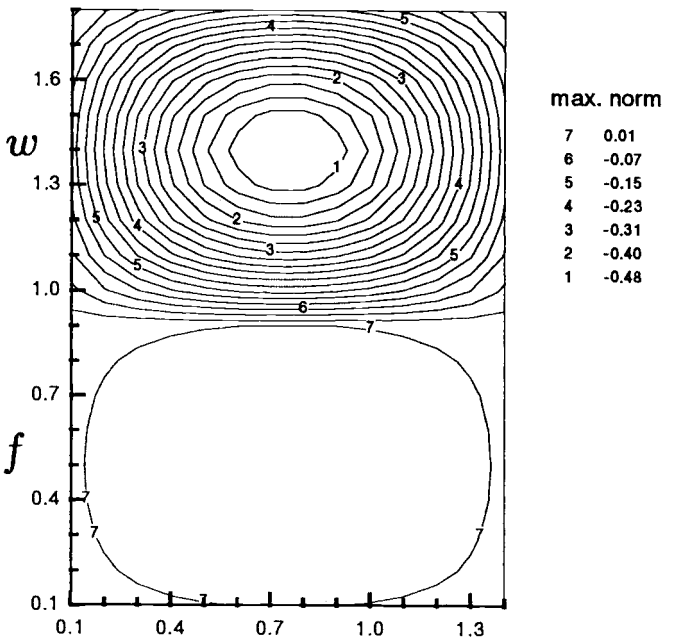


FIG. 5. *Contour of the solution curve bifurcating at* $(0, \mu_{1,1})$, $\lambda = 46.7431$, *Example* 1, $l = 1.5$.

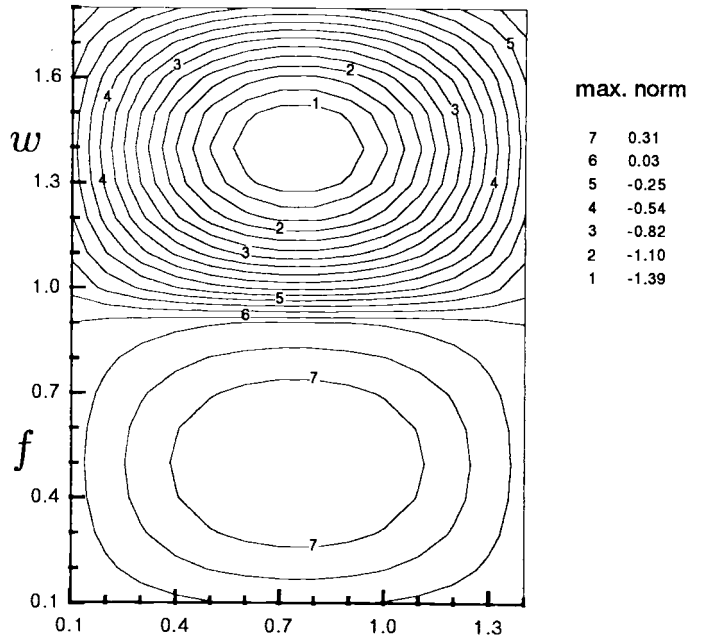


FIG. 6. *Contour of the solution curve bifurcating at* $(0, \mu_{1,1})$, $\lambda = 53.5272$, *Example* 1, $l = 1.5$.

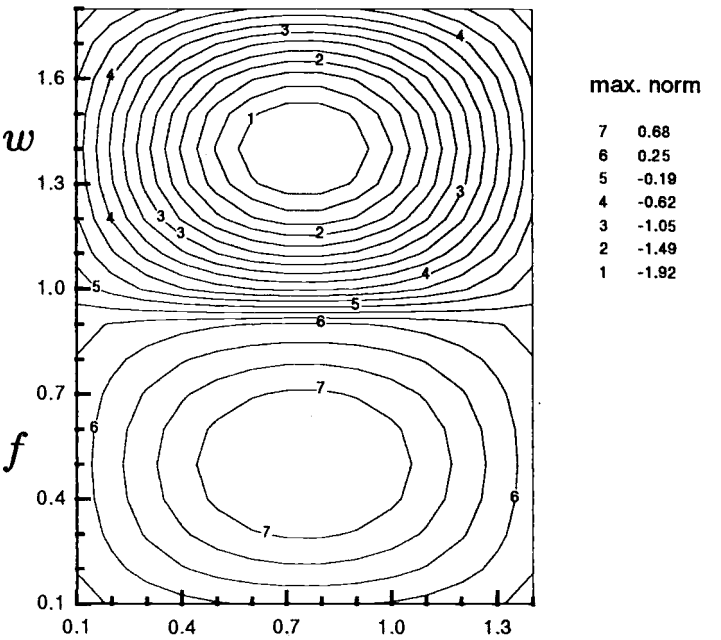


FIG. 7. *Contour of the solution curve bifurcating at* $(0, \mu_{1,1})$, $\lambda = 61.5244$, *Example* 1, $l = 1.5$.

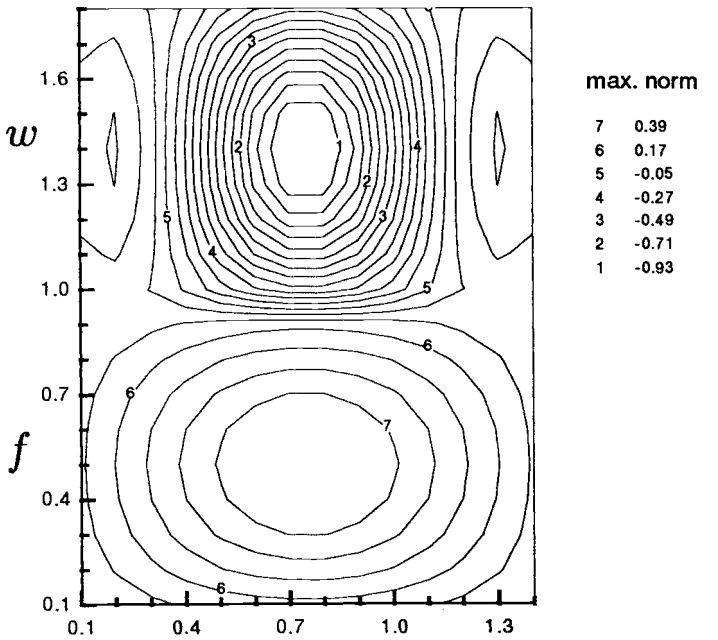


FIG. 8. *Contour of the solution curve bifurcating at* $(0, \mu_{3,1})$, $\lambda = 63.2113$, *Example* 1, $l = 1.5$.

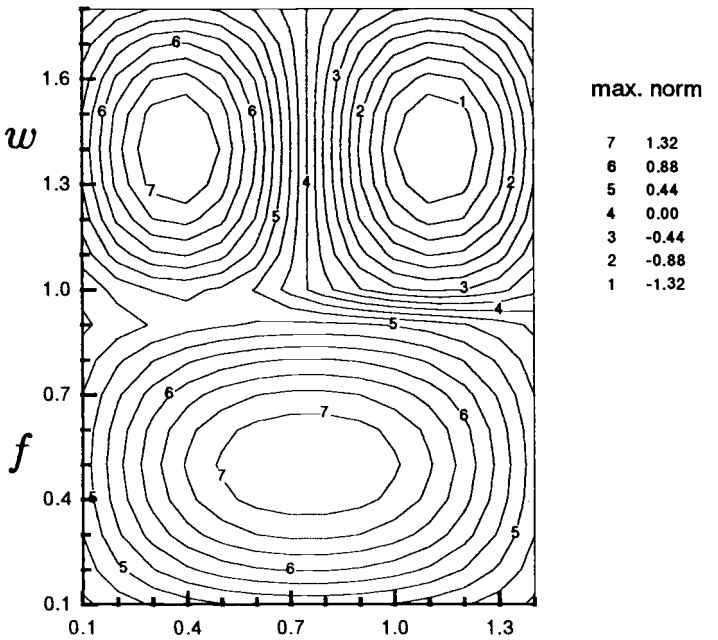


FIG. 9. *Contour of the solution curve bifurcating at* $(0, \mu_{2,1})$, $\lambda = 55.4358$, *Example* 1, $l = 1.5$.

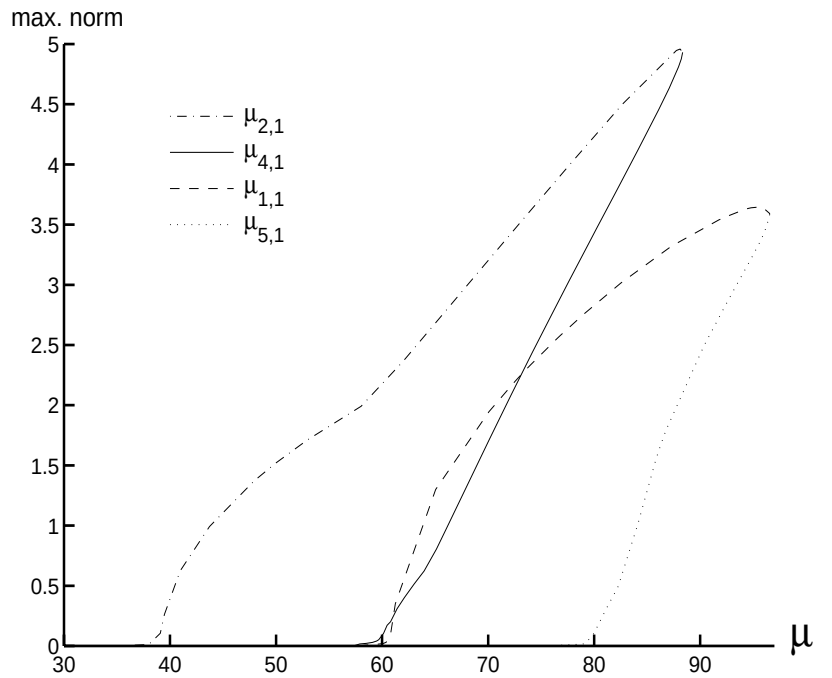


FIG. 10. *The first four solution branches of the von Kármán equations, Example* 2.

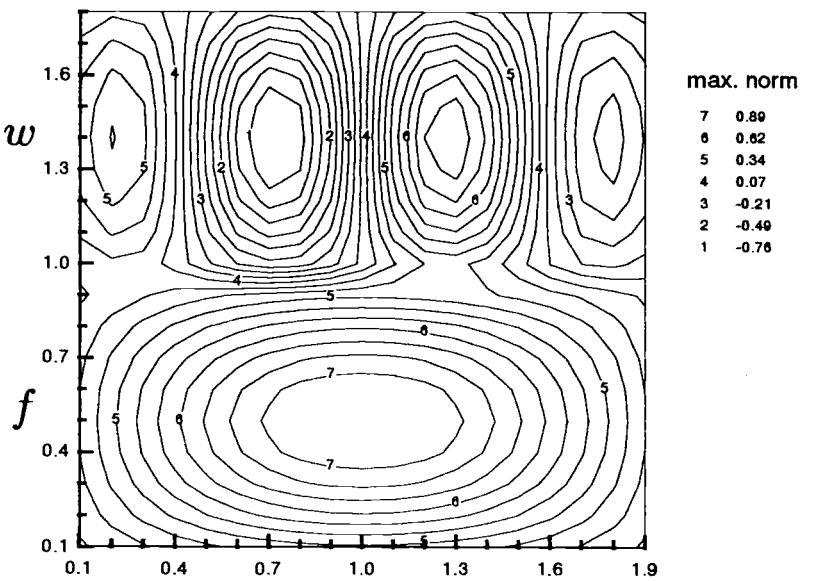


FIG. 11. *Contour of the solution curve bifurcating at* $(0, \mu_{4,1})$, $\lambda = 66.2729$, *Example* 2.

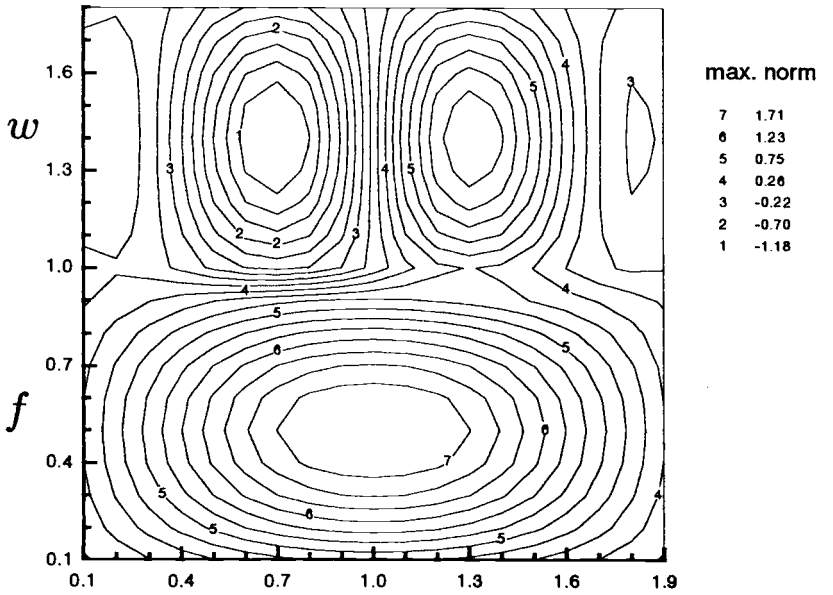


FIG. 12. *Contour of the solution curve bifurcating at* $(0, \mu_{4,1})$, $\lambda = 71.2353$, *Example* 2.

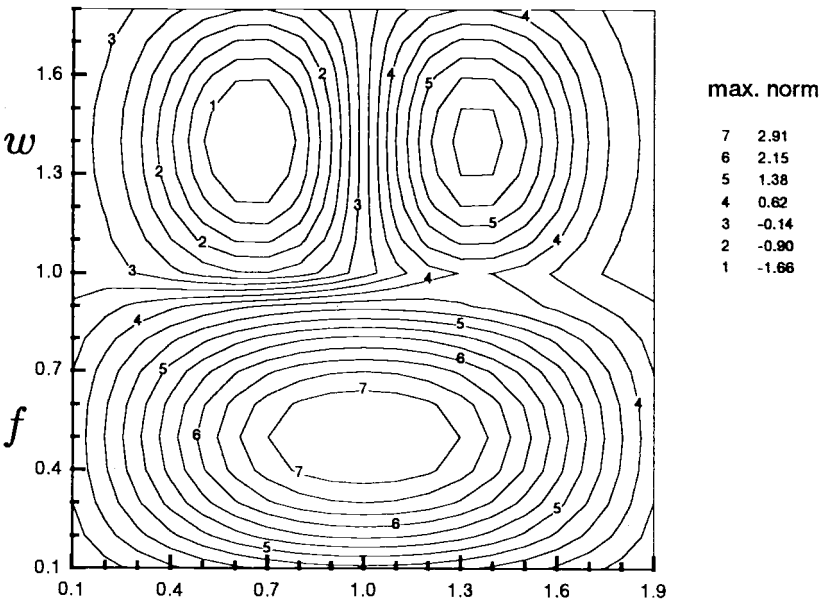


FIG. 13. *Contour of the solution curve bifurcating at* $(0, \mu_{4,1})$, $\lambda = 78.8257$, *Example* 2.

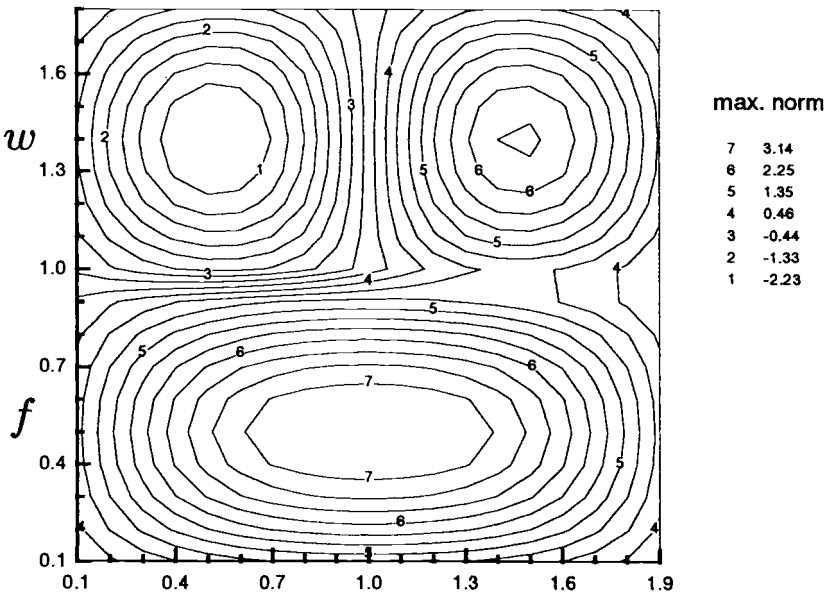


FIG. 14. *Contour of the solution curve bifurcating at* $(0, \mu_{2,1})$, $\lambda = 72.7690$, *Example* 2.

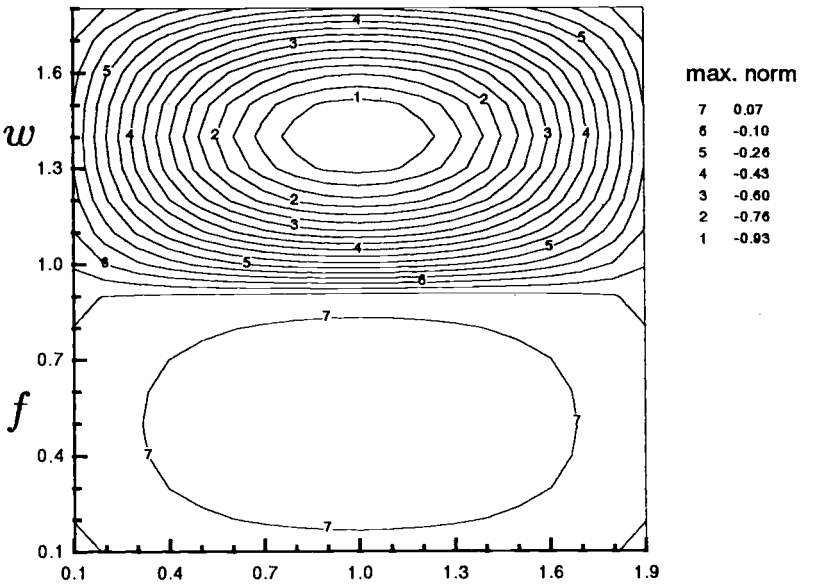


FIG. 15. *Contour of the solution curve bifurcating at* $(0, \mu_{1,1})$, $\lambda = 63.3935$, *Example* 2.

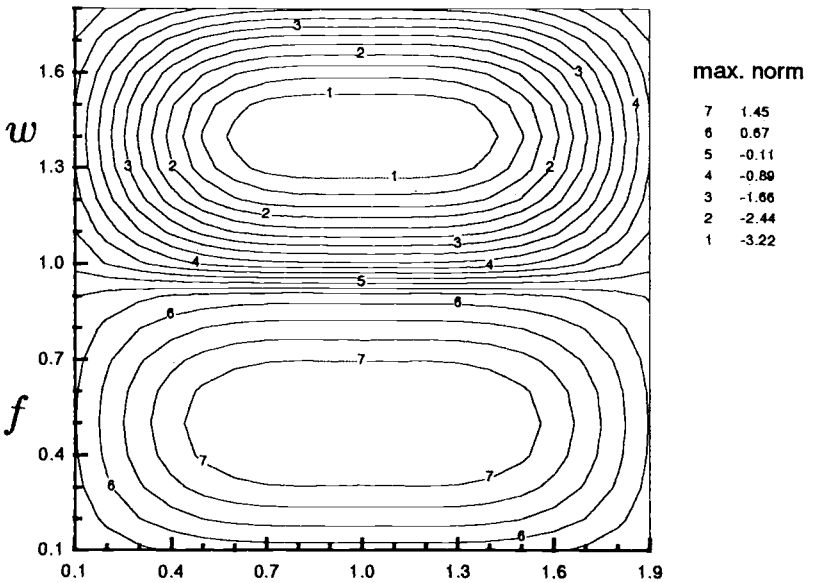


FIG. 16. *Contour of the solution curve bifurcating at* $(0, \mu_{1,1})$, $\lambda = 92.0174$, *Example* 2.

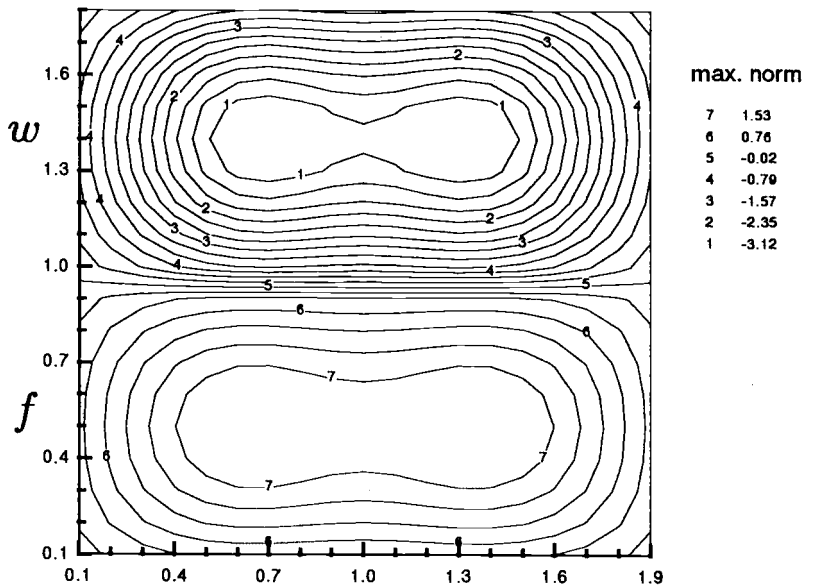


FIG. 17. *Contour of the solution curve bifurcating at* $(0, \mu_{5,1})$, $\lambda = 96.1390$, *Example* 2.

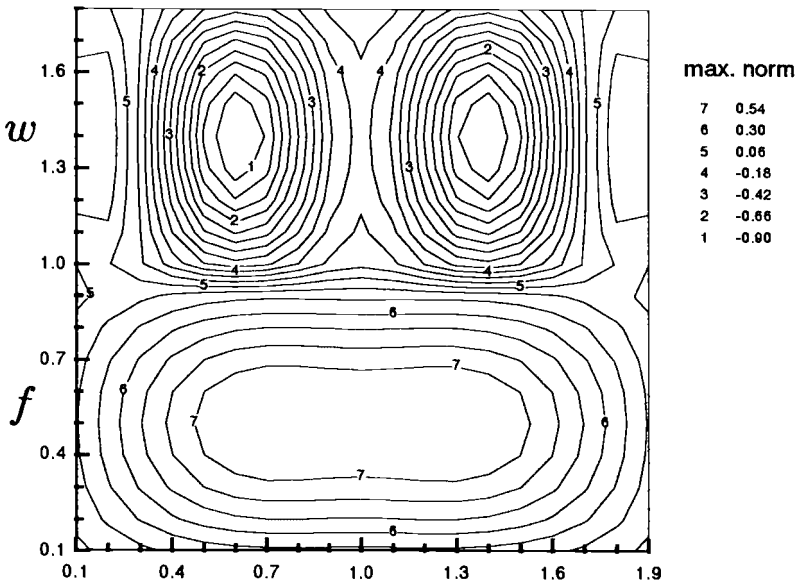


FIG. 18. *Contour of the solution curve bifurcating at* $(0, \mu_{5,1})$, $\lambda = 84.0867$, *Example* 2.

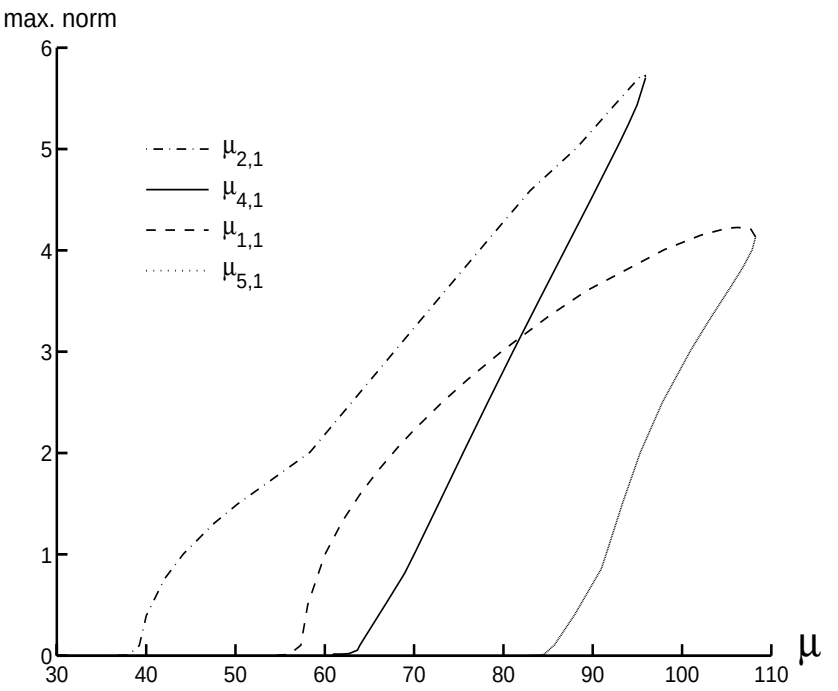


FIG. 19. *Solution curves bifurcating at* $(0, \mu_{1,1})$ *and* $(0, \mu_{4,1})$, $\Omega = [0, 1.9] \times [0, 1]$, *Example* 2.

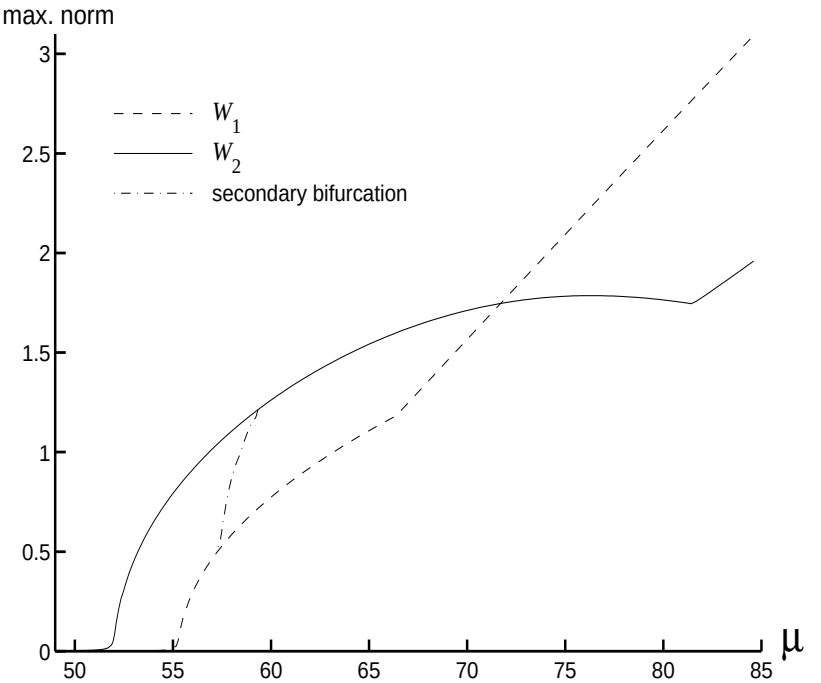


FIG. 20. *The first two solution curves of the von Kármán equations, Example* 3, $l = 1.6$.

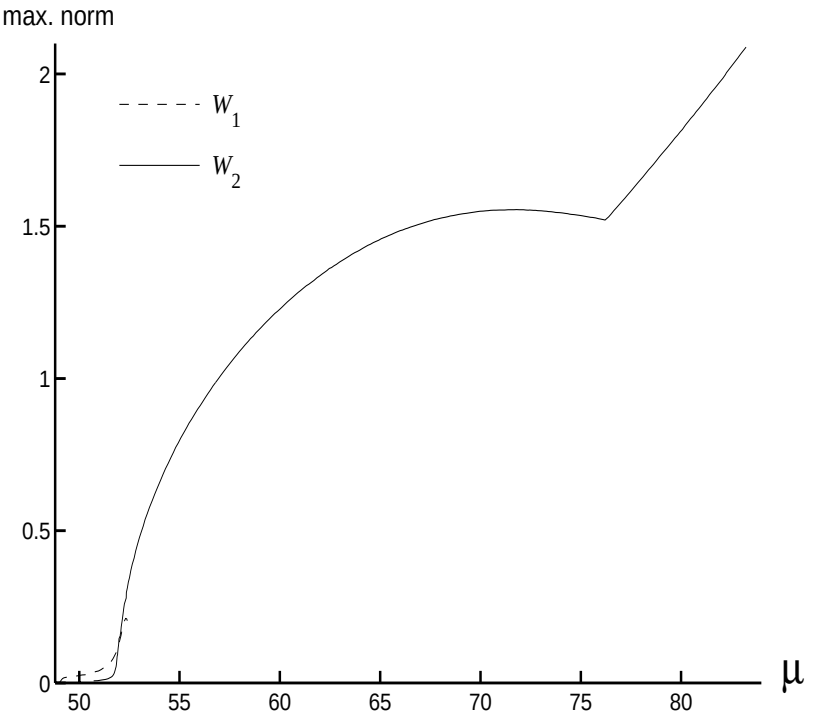


FIG. 21. *The first two solution curves of the von Kármán equations, Example* 3, $l = 1.7$.

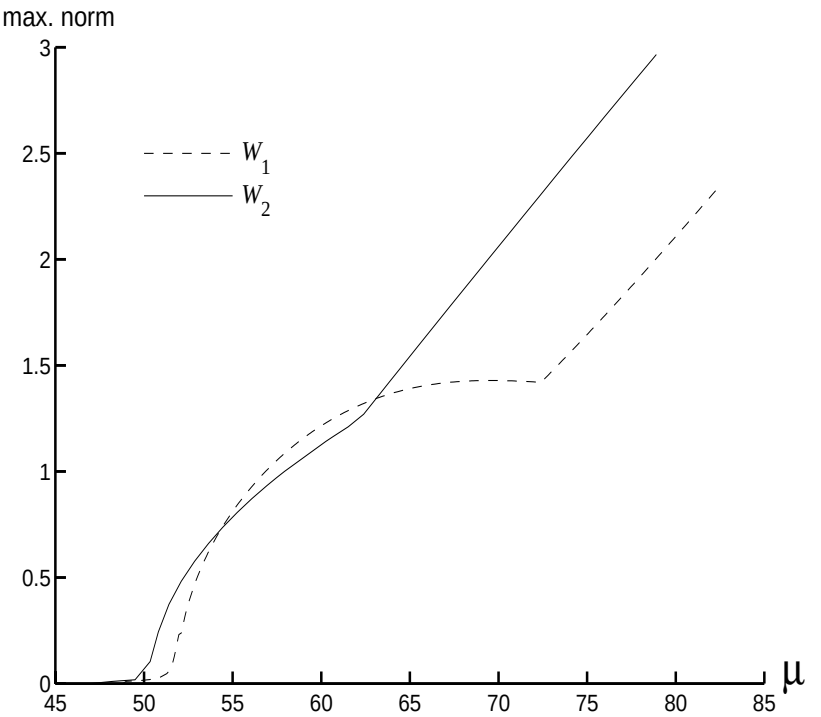


FIG. 22. *The first two solution curves of the von Kármán equations, Example* 3, $l = 1.8$.

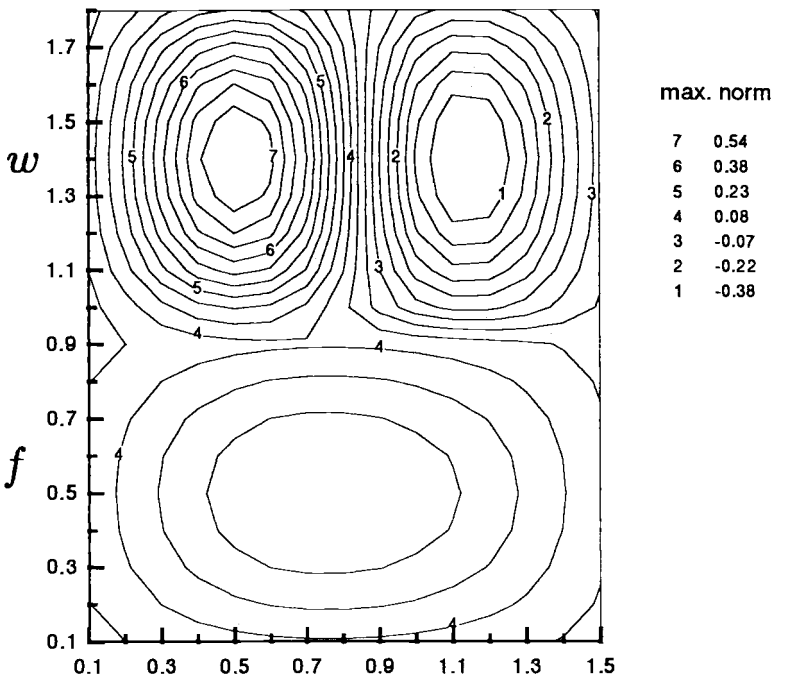


FIG. 23. *Contour of the* W_1 *solution branch at* $\mu \approx 57.4876$, *Example* 3.

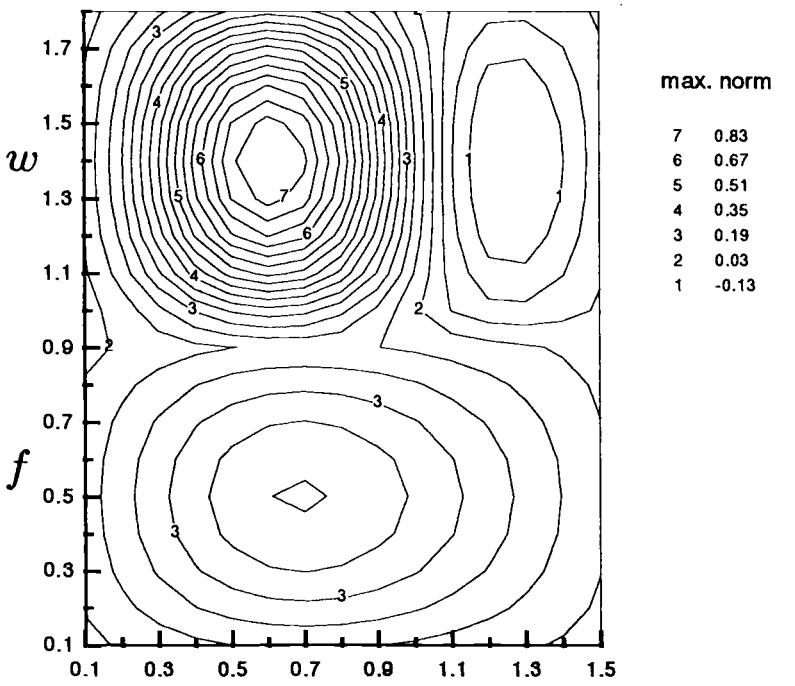


FIG. 24. *Contour of the secondary solution branch at* $\mu \approx 58.0849$, *Example* 3.

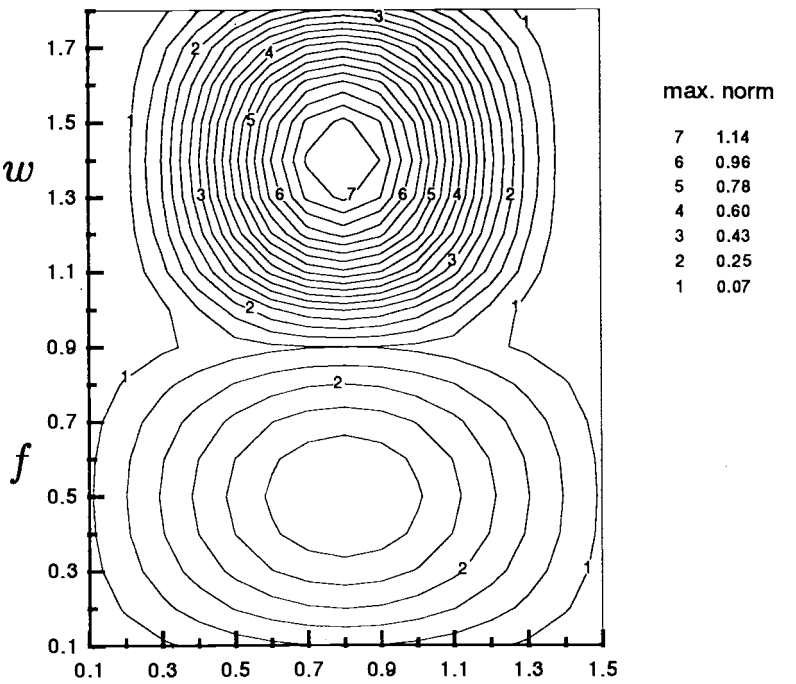


FIG. 25. *Contour of the* W_2 *solution branch at* $\mu \approx 59.3373$, *Example* 3.

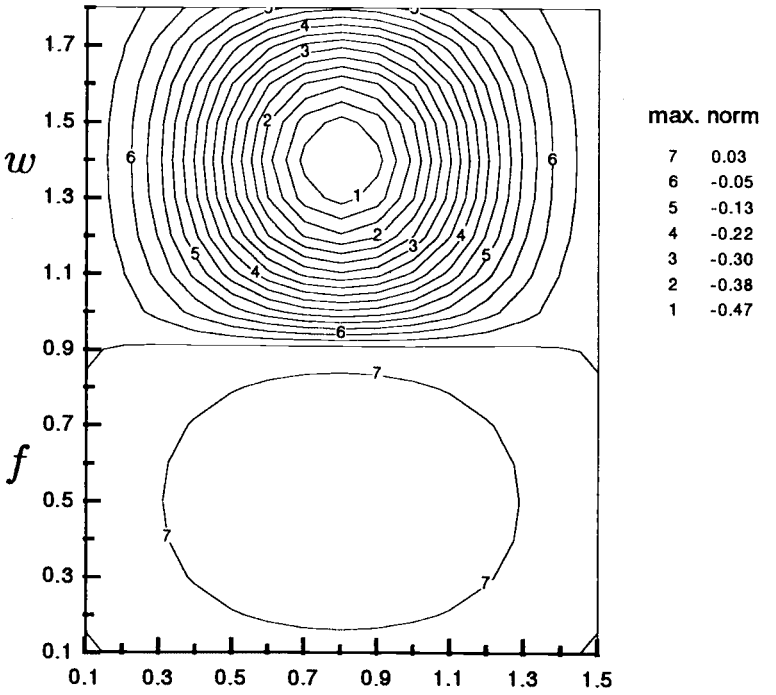


FIG. 26. *Contour of the W_2 solution branch at $\mu \approx 53.2187$, Example* 3.

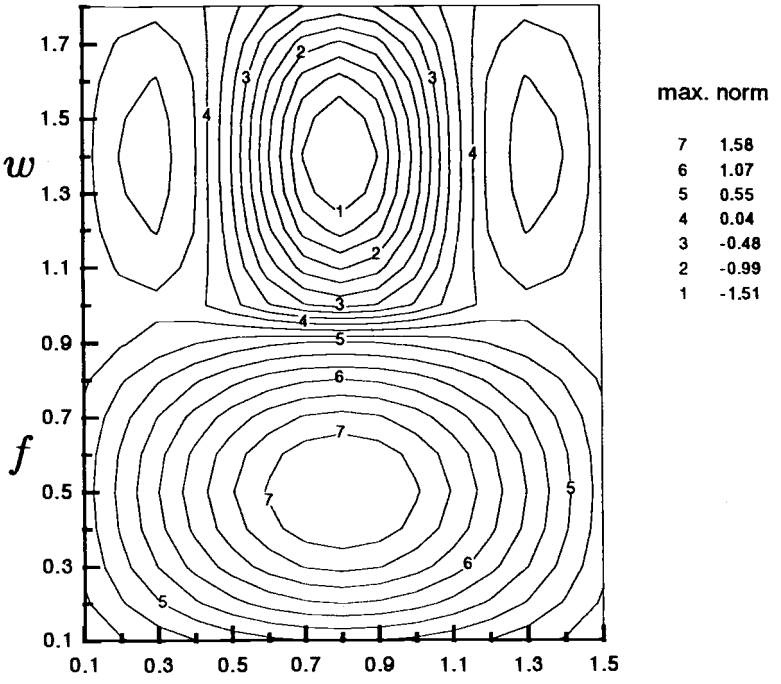


FIG. 27. *Contour of the W_1 solution branch at $\mu \approx 82.3688$, Example* 3.

solution curve. A comparison of these methods and some information of the solution curve are described in Tables 1–2 with the following notation:

MAXNORM	:	maximal norm of W along the solution curve.
x_0	:	initial guess for the Bi-CGSTAB.
$\tilde{r}_0$	:	an arbitrary vector such that $(\tilde{r}_0, r_0) \neq 0$, e.g., $\tilde{r}_0 = r_0 = b - Ax_0$.
NCS	:	order of the continuation steps.
ε	:	tolerance in Newton corrector.
θ_1	:	minimum eigenvalue of T_j.
σ_1	:	minimum singular value of T_j.
κ_2	:	the two-norm condition number of T_j.
tol	:	stopping criterion for the unsymmetric Lanczos method.
η	:	number of negative eigenvalues of T_j.

TABLE 1

Sample result for Example 4, $h = 0.0625, \varepsilon = 5 \cdot 10^{-4}, \|d\|_\infty = 5 \cdot 10^{-6}, tol = 5 \cdot 10^{-9}, \mu_{1,1} =$ 39.351783.

Method	$x_0^{(i)}$	$\tilde{r}_0^{(i)}$	NCS	Parameter interval	Total iterations	Average iterations
Bi-CGSTAB	1.0E-3	1.0E-2	52	(0,89.678)	26419	145.159
Preconditioned Bi-CGSTAB	1.0E-3	1.0E-2	52	(0,89.678)	24778	136.143
unsymmetric Lanczos	1.0E-7	*	51	(0,89.681)	29006	162.045

TABLE 2

Sample result for Example 4, $h = 0.0625, \varepsilon = 5 \cdot 10^{-4}, \|d\|_\infty = 5 \cdot 10^{-6}, tol = 5 \cdot 10^{-9}, \mu_{1,1} =$ 39.351783, *using unsymmetric Lanczos method*, $x_0^{(i)} = 10^{-7}$.

μ	MAXNORM	NCS	η	$\dim(T_j)$	θ_1	σ_1	κ_2
19.0000	1.51E-7	2	0	36	1.322E-1	1.322E-1	4.743E+2
38.4717	5.96E-2	7	10	159	1.679E-4	1.673E-4	8.324E+6
38.5584	6.58E-2	8	2	127	1.658E-4	1.652E-4	8.324E+6
38.6403	7.30E-2	9	0	130	1.686E-4	1.680E-4	3.496E+6
39.1444	1.92E-1	20	12	179	9.054E-4	1.441E-5	2.372E+9
39.1683	2.05E-1	21	16	152	1.032E-3	9.993E-4	1.813E+6
39.2919	2.78E-1	27	16	165	1.798E-3	9.792E-4	3.698E+6
39.3108	2.90E-1	28	16	173	1.914E-3	1.196E-3	1.958E+6
39.4596	3.88E-1	36	22	164	2.652E-3	5.436E-4	5.317E+6
39.8913	5.91E-1	42	24	138	4.87E-3+5.85E-4i	3.044E-4	2.926E+6

REFERENCES

[1] E. L. ALLGOWER AND C.-S. CHIEN, *Continuation and local perturbation for multiple bifurcations*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 1265–1281.

[2] E. L. ALLGOWER AND K. GEORG, *Numerical Continuation: An Introduction*, Springer-Verlag, Berlin, 1990.

[3] L. BAUER, H. B. KELLER, AND E. L. REISS, *Multiple eigenvalues lead to secondary bifurcation*, SIAM Rev., 17 (1975), pp. 101–122.

[4] M. S. BERGER, *On von Kármán's equations and the buckling of a thin elastic plate,* I. *The clamped plate*, Comm. Pure Appl. Math., 20 (1967), pp. 687–719.

[5] F. BREZZI, J. RAPPAZ, AND P. A. RAVIART, *Finite dimensional approximation of nonlinear problems. Part* 1: *Branches of nonsingular solutions*, Numer. Math., 36 (1980), pp. 1–25.

[6] F. Brezzi, J. Rappaz, and P. A. Raviart, *Finite dimensional approximation of nonlinear problems. Part* 3: *Simple bifurcation points*, Numer. Math., 38 (1981), pp. 1–30.
[7] P. G. Ciarlet, *Plates and Junctions in Elastic Multi-Structures: An Asymptotic Analysis*, Springer-Verlag, Berlin, 1990.
[8] C.-S. Chien, *Secondary bifurcations in the buckling problem*, J. Comput. Appl. Math., 25 (1989), pp. 277–287.
[9] C.-S. Chien, N.-H. Lu, and Z.-L. Weng, *Conjugate gradient methods for continuation problems* II, J. Comput. Appl. Math., 62 (1995), pp. 197–216.
[10] C.-S. Chien and M.-S. Chen, *Multiple bifurcation in the von Kármán equations*, SIAM J. Sci. Comput., 18 (1997), pp. 1737–1766.
[11] C.-S. Chien, J.-L. Weng, and C.-L. Shen, *Lanczos-type methods for continuation problems*, Numer. Linear Algebra Appl., 4 (1997), pp. 23–41.
[12] C.-S. Chien and C.-L. Shen, *A Continuation-Arnoldi Algorithm for Bifurcation Problems*, Technical report NSC86-2115-M005-004, National Science Council of ROC (Taiwan).
[13] G. H. Golub and C. F. Van Loan, *Matrix Computations*, Johns Hopkins University Press, Baltimore, MD, 1989.
[14] M. Golubitsky and D. G. Schaeffer, *Singularities and Groups in Bifurcation Theory*, Vol. I, Springer-Verlag, New York, 1984.
[15] E. J. Holder and D. Schaeffer, *Boundary conditions and mode jumping in the von Kármán's equations*, SIAM J. Math. Anal., 15 (1984), pp. 446–458.
[16] J. Huitfeldt and A. Ruhe, *A new algorithm for numerical path following applied to an example from hydrodynamical flow*, SIAM J. Sci. Comput., 11 (1990), pp. 1181–1192.
[17] E. Isaacson and H. B. Keller, *Analysis of Numerical Methods*, John Wiley, New York, 1965.
[18] H. B. Keller, *Lectures on Numerical Methods in Bifurcation Problems*, Springer-Verlag, Berlin, New York, 1987.
[19] C. Lanczos, *Solution of systems of linear equations by minimized iteration*, J. Research Nat. Bur. Standards, 49 (1952), pp. 33–53.
[20] Z. Mei, *Bifurcations of a simplified buckling problem and the effect of discretizations*, Manuscripta Math., 71 (1991), pp. 225–252.
[21] Z. Mei and F. Theil, *Variation of bifurcations along a homotopy from Neumann to Dirichlet problems*, Nonlinear Anal., 27 (1996), pp. 1381–1395.
[22] B. N. Parlett, *The Symmetric Eigenvalue Problems*, Prentice-Hall, Englewood Cliffs, NJ, 1980.
[23] Y. Saad, *The Lanczos biorthogonalization algorithm and other oblique projection methods for solving large unsymmetric systems*, SIAM J. Numer. Anal., 19 (1982), pp. 485–506.
[24] D. G. Schaeffer and M. Golubitsky, *Boundary conditions and mode jumping in the buckling of a rectangular plate*, Commun. Math. Phys., 69 (1979), pp. 209–236.
[25] M. Stein, *Loads and Deformations of Buckled Rectangular Plates*, NASA Technical report R-40, 1959.
[26] R. Szilard, *Theory and Analysis of Plates—Classical and Numerical Methods*, Civil Engrg. & Engrg. Mech. Series, Prentice Hall, Englewood Cliffs, NJ, 1974.
[27] H. A. van der Vorst, *Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 13 (1992), pp. 631–644.

SIAM J. SCI. COMPUT.
Vol. 22, No. 4, pp. 1386–1410
© 2000 Society for Industrial and Applied Mathematics

HIGH-ORDER DIRECT STOKES SOLVERS WITH OR WITHOUT TEMPORAL SPLITTING: NUMERICAL INVESTIGATIONS OF THEIR COMPARATIVE PROPERTIES*

E. LERICHE† AND G. LABROSSE ‡

Abstract. A recently proposed direct Stokes solver which decouples the velocity and pressure operators without calling for a temporal scheme is numerically analyzed, in comparison, first with the splitting scheme proposed by G. Karniadakis, M. Israeli, and S. Orszag in [*J. Comput. Phys.*, 97 (1991), pp. 414–443], and with the unique grid $(\mathbb{P}_N, \mathbb{P}_{N-2})$ Uzawa approach for the space accuracy and computational costs. The Chebyshev collocation approximation is used to analyze the spectra of the continuous temporal evolution operators, and their discrete time versions, from the first to the fourth order. An explicit boundary condition is also involved in the proposed Stokes solver, and it is numerically shown that the trace of $\mathbf{\Delta u}$ on the boundary must be evaluated through its $-\nabla \times \nabla \times$ contribution only; otherwise the ellipticity is lost before proceeding to the time discretization. The explicit evaluation of the rotational boundary term does not prevent the first and second order in time schemes from being unconditionally stable, while the schemes built at the next two higher orders are limited by the usual explicit $(\mathcal{O}(N^{-4}))$ stability criterion on the time step. The proposed $(\mathbf{u}, p)$ decoupling gives this limitation a less restrictive coefficient and supplies the expected temporal orders on the whole explored range of time step sizes. The effective accuracy obtained for the Navier–Stokes 2D solutions has been measured, for the $Re = 1000$ lid-driven cavity problem, and has been found equivalent to what is supplied by the much more expensive Uzawa decoupling method.

Key words. Stokes equations, decoupling methods, temporal splitting methods, Stokes eigenvalues and eigenmodes, spectral Chebyshev approximation

AMS subject classifications. 35Q30, 65N35

PII. S1064827598349641

1. Introduction.

Since the first numerical experiments were performed with the incompressible Navier–Stokes equations, solving the linear Stokes system has been of major concern with respect to its stability, accuracy, and computational efficiency once the nonlinear terms are explicitly treated as extra source. The question remains how to decouple the pressure and velocity fields in order to get an easily tractable numerical system with given stability and space-time accuracy properties. Many strategies have been proposed. They belong essentially to three main categories. The first historically proposed decoupling, the so-called Uzawa method, has been designed in a finite element framework [1] and applied within the spectral and spectral element context in many different implementations [31]. This method has the main advantage that the discrete problem is equivalent to the continuous case, and the numerical solutions converge to the exact one [25, 6]. When the diffusion terms are implicitly evaluated in time, this is obtained at the cost of an expensive, iterative inversion of the resulting full matrix, the Uzawa operator, including a vectorial Helmholtz solver [26]. An alternative comes from the fractional step methods, initially proposed by

*Received by the editors December 22, 1998; accepted for publication (in revised form) July 18, 2000; published electronically October 25, 2000.

http://www.siam.org/journals/sisc/22-4/34964.html

†Fluid Mechanics Laboratory, Department of Mechanical Engineering, Swiss Federal Institute of Technology (Lausanne), LMF-DGM-EPFL, CH-1015 Ecublens, Switzerland (leriche@azur.epfl.ch). The work of this author was supported by the Fonds National Suisse de la Recherche Scientifique under grant FN 20-43470.95.

‡Université Paris-Sud, LIMSI-CNRS, BP 133, 91403 Orsay Cedex, France (labrosse@limsi.fr). The work of this author was supported by the ERCOFTAC visitor program sponsored by the L. Euler Pilot Center (Switzerland) for several long stays at EPFL.

[8] and [34, 35], and significantly improved by an extension to high temporal order discretizations [17]. In these methods, the decoupling of the velocity and pressure operators is obtained by a particular splitting of the discretized Stokes system, based on the time integration scheme. The corresponding linear system, as well as the induced properties (stability and temporal accuracy) of such a splitting are analyzed in [28]. The third category refers to the influence matrix or Green's function method [18]. Its implementation for three-dimensional (3D) flows, with N^3 nodes and no periodic direction, leads to the storage of the inverse of a large and full matrix of order $6N^2$, the capacitance matrix, and its application at each time step. As a consequence, this method is limited to two-dimensional (2D) flows simulation.

A projection-diffusion method (hereafter referred to as the PRDI method) has been introduced in [4] for the Chebyshev collocation method. No time scheme is required at the decoupling stage of the velocity and pressure operators. Thus, although not very far from the splitting formulation of [17] (hereafter referred to as the KIO method), the PRDI method is not of the fractional step type and does not suffer from the inconsistency that the authors in [17] point out for the splitting.

In contrast with the Uzawa approach which has received much theoretical attention (see [25] and the many references quoted therein), the splitting (or fractional step) methods have not yet been fully analyzed to the authors' knowledge. Theoretical works have been published for their low (1,2) time-order versions which require homogeneous boundary conditions on the pressure [33, 14]. However, according to Shen, "we can only expect a projection (or splitting) scheme to deliver higher than second-order accuracy if a more accurate boundary condition for the pressure is employed. A rigorous analysis for more general cases is not yet available" (page 1042 in [33]). Both schemes under consideration in this paper belong to this category where accurate pressure conditions are imposed on the boundaries. They involve a $-\nabla\times\nabla\times$ boundary contribution (see [27, 17] and below) which makes the theoretical work very difficult and unavailable so far. In absence of theoretical support, numerical results can be considered as a welcome help to better understand the Stokes schemes.

Thus, in this paper, a first extensive comparison of both spectral Stokes solvers, PRDI and KIO, is performed for (a) their consistency with the initial continuous problem, (b) their ability to respect the expected ellipticity, (c) their genuine stability properties, and (d) their respective temporal accuracy. The time splitting is clearly identified as responsible for inconsistencies of the KIO scheme, with an analysis which can be extrapolated to any fractional step scheme. Finally, it is shown that using the trace of $\mathbf{\Delta u}$ (instead of $-\nabla\times\nabla\times$) in a normal boundary condition breaks the ellipticity of both Stokes solvers, explaining the origin of the previously observed numerical instabilities [27, 17].

The analysis reported hereafter has been carried out in a simple test case, the bounded square Cartesian geometry, as a first step to reach before considering complex domains to be treated with more sophisticated numerical methods. Both solvers are thus based on a monodomain Chebyshev collocation method [13, 7]. As an aside, a first insight is offered into the eigenmodes of this 2D Stokes problem, since, to the authors' knowledge, they are not yet analytically known in a bounded domain.

The paper is organized as follows. After section 2, where the Stokes problem is recalled, in section 3 both methods and their main features are presented. In section 4, the continuous temporal evolution operator spectra are given for both solvers. Then, section 6 deals with the fully discretized solvers of temporal order one to four. A stability analysis is performed by eigenvalue computation. The first- and second- order

in time schemes are found to be unconditionally stable, whereas the third- and fourth-order present an $\mathcal{O}(N^{-4})$ stability restriction on the time step. This restriction is less severe for the PRDI than for the KIO solver. Then, the effective temporal accuracy of both solvers is measured by 3D numerical experiments performed with the first and second order in time collocation schemes. The PRDI method supplies exactly the expected temporal order on the whole explored time step range. Nevertheless, the best effective accuracy comes always from the KIO solver in the tests we have implemented.

The last section is dedicated to some practical comparisons between the PRDI and the unique grid $(\mathbb{P}_N, \mathbb{P}_{N-2})$ Uzawa approach applied to a classical Navier–Stokes test-case, namely the 2D regularized lid-driven cavity problem. The solutions supplied by the PRDI solver are as accurate as the Uzawa ones but obtained at a lower cost.

2. The governing equations. Let us consider the dimensionless unsteady (time t) 2D or 3D Stokes equations, written in the Cartesian open domain $\Omega = (]-1,+1[)^d$ with coordinates $\mathbf{x} = (x_i, i = 1, d)$, $\equiv (x, y)$ or $\equiv (x, y, z)$ according $d = 2, 3$, respectively, and T a real positive number,

$$\frac{\partial \mathbf{u}}{\partial t} = \boldsymbol{\Delta}\mathbf{u} - \nabla p + \mathbf{f} \qquad \text{for } (\mathbf{x}, t) \in \Omega\times]0, T[\ , \tag{2.1}$$

$$\nabla \cdot \mathbf{u} = 0 \qquad \text{for } (\mathbf{x}, t) \in \Omega\times]0, T[\ , \tag{2.2}$$

where $\mathbf{u} = (u, v, w)$ is the velocity field, p the pressure (normalized by the fluid's density), and $\mathbf{f}$ the source term. We denote the closure of Ω by $\overline{\Omega}$ and the boundary by $\partial\Omega$. For the sake of simplicity, we consider Dirichlet boundary conditions

$$\mathbf{u} = \mathbf{U} \qquad \text{for } (\mathbf{x}, t) \in \partial\Omega\times]0, T[, \tag{2.3}$$

and compatible initial conditions are given,

$$\mathbf{u}(t = 0) = \mathbf{U}^0 \quad \text{for } \mathbf{x} \in \Omega, \tag{2.4}$$

$\mathbf{U}^0$ being divergence-free. The viscosity has been inserted in the time, pressure, and source scales.

The present study is focused on the Stokes problem (2.1)–(2.2), but the Navier–Stokes equations are behind it if the source term $\mathbf{f}$ includes the nonlinear advective terms beside prescribed body forces. It is assumed from now on that the continuous spaces for the velocity and pressure are, respectively, $(H_0^1(\Omega))^d$ and $L_0^2(\Omega)$ for almost every time $t \in]0, T[$. $L_0^2(\Omega)$ is the space of functions which are square integrable in Ω with a zero mean value on Ω, $H^1(\Omega)$ is the space of functions of which the first derivative is square integrable in Ω, and $H_0^1(\Omega)$ is the closure in $H^1(\Omega)$ of the space of all functions continuously differentiable any number of times with compact support in Ω. The existence and uniqueness of a solution, lying in those spaces, of the unsteady Navier–Stokes equations are established and discussed in [29], under some assumptions considered as verified in this work. Under additional assumptions it is also possible to prove the existence of more regular solutions [16, 29].

For the forthcoming eigenmode analysis and investigation into the stability properties of the Stokes solvers, $\mathbf{f}$ is inessential and dropped from now on. The well-known equivalent continuous formulation of this problem, where the velocity and pressure are decoupled, reads as follows:

$$\boldsymbol{\Delta} p = 0 \qquad \text{for } (\mathbf{x}, t) \in \Omega\times]0, T[, \tag{2.5}$$

$$\left(\frac{\partial}{\partial t} - \boldsymbol{\Delta}\right) \boldsymbol{\Delta}\ \mathbf{u} = 0 \qquad \text{for } (\mathbf{x}, t) \in \Omega\times]0, T[. \tag{2.6}$$

Thus, the unsteady Stokes equations lead to harmonic pressure and biharmonic velocity fields.

3. The Stokes solvers. The unsteady Stokes problem is now solved with two different approaches. The first one is the well-established fractional step algorithm, denoted KIO, and it is briefly summarized in the beginning of this section. Basically linked to a time scheme, its presentation is made in a semidiscrete framework, discretized in time and continuous in space. The second one, the PRDI, comes then, with more details.

3.1. The KIO method. Proposed in 1991 as a high-order splitting method for the Navier–Stokes equations, the KIO *time* discretized system, applied to the unsteady Stokes problem, is based on the backward Euler scheme of order J_i. Its continuous-in-space version reads

$$\frac{\hat{\mathbf{u}} - \sum_{q=0}^{J_i-1} \alpha_q \mathbf{u}^{n-q}}{\Delta t} = -\nabla p \qquad \text{for } \mathbf{x} \in \Omega, \tag{3.1}$$

$$\nabla \cdot \hat{\mathbf{u}} = 0 \qquad \text{for } \mathbf{x} \in \Omega, \tag{3.2}$$

$$\frac{\gamma_0 \mathbf{u}^{n+1} - \hat{\mathbf{u}}}{\Delta t} = \boldsymbol{\Delta} \mathbf{u}^{n+1} \qquad \text{for } \mathbf{x} \in \Omega, \tag{3.3}$$

where $\hat{\mathbf{u}}$ is an intermediate velocity field, constrained to fulfill the incompressibility condition (3.2), and $\mathbf{u}^n \equiv \mathbf{u}(n\ \Delta t)$. The weights γ_0 and α_q, up to order $J_i = 3$, may be found in [17] and are gathered with the fourth-order coefficients in Table 1. The velocity field $\mathbf{u}$ satisfies the Dirichlet boundary conditions (2.3). The equations (3.1)–(3.2) are combined to lead to the Poisson equation for the pressure, which can be solved provided boundary conditions are chosen to be compatible with the temporal order (see below). They are obtained first by taking the trace on the boundary of the normal component of the sum of (3.1) and (3.3) and second by splitting $\boldsymbol{\Delta}\mathbf{u}$ into two parts,

$$\boldsymbol{\Delta}\mathbf{u} = -\nabla \times (\nabla \times \mathbf{u}) \ + \ \nabla(\nabla \cdot \mathbf{u}), \tag{3.4}$$

the solenoidal (resp., irrotational) part being treated explicitly (resp., implicitly) in time. This decomposition was first introduced in [27] to avoid exponential growth of a nonzero divergence of the velocity field $\mathbf{u}$ on the boundaries. The irrotational part is dropped in accordance with (2.2). Section 4.1 will elucidate the basic reason why such a procedure must be applied. Finally, using the extrapolation scheme of order J_e proposed in [17], the resulting continuous-in-space pressure system is

$$\boldsymbol{\Delta} p = \nabla \cdot \left(\frac{\sum_{q=0}^{J_i-1} \alpha_q \mathbf{u}^{n-q}}{\Delta t} \right) \qquad \text{for } \mathbf{x} \in \Omega, \tag{3.5}$$

$$\left[\frac{\partial p}{\partial \mathbf{n}} \right]_{\mathbf{x} \in \partial\Omega} = \left[\left(-\left(\frac{\partial \mathbf{U}}{\partial t} \right)^{n+1} - \sum_{q=0}^{J_e-1} \beta_q \nabla \times (\nabla \times \mathbf{u}^{n-q}) \right) \cdot \mathbf{n} \right]_{\mathbf{x} \in \partial\Omega} . \tag{3.6}$$

The weights β_q are listed in Table 1 and $\mathbf{n}$ is the outward normal unit vector.

This work considers only the case $J_e = J_i$.

It is worth writing the continuous-in-time problem, with decoupled pressure and velocity fields, which is equivalent to the above split system (3.1)–(3.3), and comparing

TABLE 1
The weights γ_0, α_q, and β_q.

Weights	Order 1	Order 2	Order 3	Order 4
γ_0	1	3/2	11/6	25/12
α_0	1	2	3	4
α_1	0	–1/2	–3/2	–3
α_2	0	0	1/3	4/3
α_3	0	0	0	–1/4
β_0	1	2	3	4
β_1	0	–1	–3	–6
β_2	0	0	1	4
β_3	0	0	0	–1

it with the Stokes continuous decoupled problem (2.5)–(2.6). This interesting point has already been raised in [17], whose authors pointed out that the pressure and velocity fields are now biharmonic and triharmonic, respectively. We shall reach the same conclusion by considering a continuous-in-time formulation of the system (3.1)–(3.3). At any time order, the derivative $\frac{\partial \mathbf{u}}{\partial t}$ is approximated by a linear combination of instantaneous velocities, of which one part, $-\sum_{q=0}^{J_i-1} \alpha_q \mathbf{u}^{n-q}/\Delta t$, is taken in the first step (3.1), while the second one, $\frac{\gamma_0 \mathbf{u}^{n+1}}{\Delta t}$, belongs to the diffusion step (3.3). A simple way to write an equivalent continuous-in-time formulation of this form of splitting is to consider that each part of the time derivative $\frac{\partial \mathbf{u}}{\partial t}$ is proportional to $\frac{\partial \mathbf{u}}{\partial t}$ itself. This holds for a normal mode analysis of the unsteady Stokes problem, and leads to the equivalent set of coupled continuous equations

$$\hat{\mathbf{u}} + \eta \, \frac{\partial \mathbf{u}_\eta}{\partial t} = -\nabla p_\eta \qquad \text{for } (\mathbf{x}, t) \in \Omega \times]0, T[\,, \tag{3.7}$$

$$\nabla \cdot \hat{\mathbf{u}} = 0 \qquad \text{for } (\mathbf{x}, t) \in \Omega \times]0, T[\,, \tag{3.8}$$

$$(1 - \eta) \, \frac{\partial \mathbf{u}_\eta}{\partial t} - \hat{\mathbf{u}} = \boldsymbol{\Delta} \mathbf{u}_\eta \qquad \text{for } (\mathbf{x}, t) \in \Omega \times]0, T[\,, \tag{3.9}$$

where the intermediate $\hat{\mathbf{u}}$ is now an acceleration field, and $(\mathbf{u}_\eta, p_\eta)$ the solutions of this new problem. With modes such that $\mathbf{u}_\eta^n = \kappa^n \mathbf{u}^0$ and $\kappa = \exp \lambda \Delta t$, it is easily seen that the coefficient η is related to the temporal order of the scheme by the simple relation

$$\frac{1 - \eta}{\eta} = -\frac{\gamma_0 \, \kappa}{\sum_{q=0}^{J_i-1} \alpha_q \kappa^{-q}}. \tag{3.10}$$

Obviously, there would be no such simple relation between $\frac{\partial \mathbf{u}}{\partial t}$ and its two split parts for a field $\mathbf{u}(t)$ having any time dependence. But the one adopted here can be applied to the Stokes eigenmodes. By successive differential eliminations performed on the continuous system (3.7)–(3.9), the decoupled continuous problem then reads as follows:

$$\left((1 - \eta) \frac{\partial}{\partial t} - \boldsymbol{\Delta} \right) \boldsymbol{\Delta} p_\eta = 0 \qquad \text{for } (\mathbf{x}, t) \in \Omega \times]0, T[, \tag{3.11}$$

$$\left((1 - \eta) \frac{\partial}{\partial t} - \boldsymbol{\Delta} \right) \left(\frac{\partial}{\partial t} - \boldsymbol{\Delta} \right) \boldsymbol{\Delta} \mathbf{u}_\eta = 0 \qquad \text{for } (\mathbf{x}, t) \in \Omega \times]0, T[. \tag{3.12}$$

Compared with the initial decoupled Stokes problem (2.5)–(2.6), it can be seen that a supplementary space-time operator has appeared, the first one on the left side of

these equations, directly induced by the time splitting, as mentioned in [17]. The origin of each term of this new operator is now identified. The extra time derivative is due to the splitting of $\frac{\partial \mathbf{u}}{\partial t}$ ($\eta \neq 1$), and the supplementary degree of harmonicity ($\eta = 1$) is introduced by the splitting between $\frac{\partial \mathbf{u}}{\partial t}$ and $\boldsymbol{\Delta}\mathbf{u}$. As a consequence, the approximate pressure field is biharmonic and governed by a time evolution equation, which is not physically consistent from the $\nabla \cdot \mathbf{u} = 0$ constraint. The velocity field satisfies a triharmonic time evolution equation. Both these fields are polluted by a numerical boundary layer of dimensionless thickness $\sqrt{t/|1-\eta|}$. Finally, the case $\eta = 0$ corresponds to a nonsplitting scheme, in which the incompressibility constraint (3.8) is applied on $\hat{\mathbf{u}} = \frac{\partial \mathbf{u}}{\partial t} - \boldsymbol{\Delta}\mathbf{u}$ by (3.9). As a result of this, the supplementary space-time operator no longer appears in (3.11)–(3.12). This is precisely the main feature of the PRDI method introduced now.

3.2. The PRDI method. The PRDI method looks like this fractional step method since it proceeds with two similar steps, namely, the determination of the pressure by imposing the incompressibility constraint to some appropriate field, and then the computation of the velocity $\mathbf{u}$ by a pure diffusion stage. However, the decoupling is now defined independently of any temporal scheme. Indeed, if we introduce the acceleration $\mathbf{a}^*$, where

$$\mathbf{a}^* = \frac{\partial \mathbf{u}}{\partial t} - \boldsymbol{\Delta}\mathbf{u},$$

the original problem (2.1)–(2.2) splits, for each $t \in]0, T[$, into two steps:

(1) *first step:*

$$\mathbf{a}^* + \nabla p = 0 \qquad \text{for } \mathbf{x} \in \Omega, \tag{3.13}$$

$$\nabla \cdot \mathbf{a}^* = 0 \qquad \text{for } \mathbf{x} \in \Omega, \tag{3.14}$$

$$\mathbf{a}^* \cdot \mathbf{n} = \left(\frac{\partial \mathbf{U}}{\partial t} - \boldsymbol{\Delta}\mathbf{u} \right) \cdot \mathbf{n} \qquad \text{for } \mathbf{x} \in \partial\Omega; \tag{3.15}$$

(2) *second step:*

$$\frac{\partial \mathbf{u}}{\partial t} - \boldsymbol{\Delta}\mathbf{u} = \mathbf{a}^* \qquad \text{for } \mathbf{x} \in \Omega, \tag{3.16}$$

$$\mathbf{u} = \mathbf{U} \qquad \text{for } \mathbf{x} \in \partial\Omega. \tag{3.17}$$

As stated at the end of the previous paragraph, this scheme is consistent with (2.5)–(2.6).

The projection step (3.13)–(3.15) can also be written as a Poisson–Neumann problem similar to (3.5)–(3.6),

$$\boldsymbol{\Delta} p = 0 \qquad \text{for } \mathbf{x} \in \Omega\ , \tag{3.18}$$

$$\left[\frac{\partial p}{\partial \mathbf{n}} \right]_{\mathbf{x} \in \partial\Omega} = \left[\left(-\frac{\partial \mathbf{U}}{\partial t} - \nabla \times (\nabla \times \mathbf{u}) \right) \cdot \mathbf{n} \right]_{\mathbf{x} \in \partial\Omega}. \tag{3.19}$$

Section 4.2 will highlight the numerical equivalence between these two projection approaches. The version (3.13)–(3.15) is always the one adopted with the PRDI method.

The two stages are coupled only through the normal boundary condition (3.15). The first step (3.13)–(3.15) allows the computation of the pressure through the solution of a Darcy-type problem, assuming that the right-hand side of the normal

boundary condition $\mathbf{a}^* \cdot \mathbf{n}$ is known (see below). The background and the numerical analysis of the Darcy problem with several variational formulations, in the framework of the spectral discretizations, may be found in [2].

The space discretization proceeds by expanding the $\mathbf{u}$, $\mathbf{a}^*$, and p fields in tensor product of Chebyshev polynomials of order (N, M, L) for the (x, y, z) dependencies, respectively. The collocation method consists of exactly enforcing the differential equations, and the boundary conditions, at the Gauss–Lobatto points [13, 7]. Let us introduce the discrete spaces, $\{\Omega\}$ and $\{\partial\Omega\}$, made of the set of the Gauss–Lobatto points located inside Ω and on the boundary $\partial\Omega$, respectively. The discrete space $\{\overline{\Omega}\}$ is the union of $\{\Omega\}$ and $\{\partial\Omega\}$. From now on, $\mathbf{u}$, $\mathbf{a}^*$, and p denote the set of the nodal values in $\{\overline{\Omega}\}$ of the corresponding fields. The discretization of (3.13) proceeds in a particular way. Indeed the ith component of this equation is collocated in the discrete space which excludes the two plane boundaries normal to the ith direction, where the conditions (3.15) are imposed.

The collocated Darcy problem (3.13)–(3.15) then reads

$$\mathbf{a}^* + \tilde{\mathcal{D}} p = \mathbf{f} \qquad \text{in} \quad \{\overline{\Omega}\}, \tag{3.20}$$

$$\mathcal{D}\cdot\mathbf{a}^* = 0 \qquad \text{in} \quad \{\overline{\Omega}\}. \tag{3.21}$$

$\mathcal{D}$ is the usual gradient operator, and its restriction by the collocation of (3.13) is noted $\tilde{\mathcal{D}}$. The discrete system (3.20) gathers what comes from (3.13) and (3.15), the right-hand side $\mathbf{f}$ coming exclusively from the discretized condition (3.15) noted $\mathbf{a}^* \cdot \mathbf{n} = \mathbf{a} \cdot \mathbf{n}$ on $\{\partial\Omega\}$. It is zero everywhere except where the normal boundary conditions are imposed:

$$\mathbf{f} = \mathbf{0} \qquad \text{except} \qquad (f_i)\,|_{x_i=\pm 1} = (a_i)\,|_{x_i=\pm 1}. \tag{3.22}$$

This vectorial field $\mathbf{f}$ will be denoted in a compact way:

$$\mathbf{f} = \text{“}\mathbf{a} \cdot \mathbf{n}.\text{”} \tag{3.23}$$

By substitution of (3.20) into (3.21), we obtain finally

$$\mathcal{E}_{\mathcal{P}\mathcal{D}}\, p = \mathcal{D}\cdot\mathbf{f} \qquad \text{in } \{\overline{\Omega}\}, \tag{3.24}$$

with

$$\mathcal{E}_{\mathcal{P}\mathcal{D}} = \mathcal{D}\cdot\tilde{\mathcal{D}}.$$

Taking advantage of the tensor product form of the Uzawa operator $\mathcal{E}_{\mathcal{P}\mathcal{D}}$, which is a quasi-Poisson operator, the fast diagonalization method [15] is used to “invert” it directly and to filter out its spurious pressure modes. For the Darcy problem, their number is 2^d $(d = 2, 3)$ [2] (see section 3.3).

Because the velocity $\mathbf{u}$ is unknown at the pressure step computation, the normal boundary condition (3.15) requires the same treatment as the one leading to (3.6). The condition (3.15) is then rewritten as

$$[\mathbf{a}^* \cdot \mathbf{n}]_{\{\partial\Omega\}} = \left[\left(\frac{\partial \mathbf{U}}{\partial t} + \sum_{q=0}^{J_e} \beta_q (\mathcal{D} \times (\mathcal{D} \times \mathbf{u}^{n-q})) \right) \cdot \mathbf{n} \right]_{\{\partial\Omega\}}. \tag{3.25}$$

Once the pressure is known, $\mathbf{a}^*$ is evaluated and the second step (3.16)–(3.17), which is a pure diffusion problem, can be solved. An implicit backward Euler scheme of order J_i is employed to perform the time integration:

$$\frac{\gamma_0 \mathbf{u}^{n+1} - \sum_{q=0}^{J_i} \alpha_q \mathbf{u}^{n-q}}{\Delta t} = \mathcal{A}_D \mathbf{u}^{n+1} + \mathbf{a}^*, \tag{3.26}$$

$\mathcal{A}_D$ denoting the Dirichlet–Laplacian matrix, the system being completed by $\mathbf{u} = \mathbf{U}$ on the boundaries. The time discretized equations (3.26) contain a vectorial Helmholtz system to be solved by the fast diagonalization method [15].

3.3. About the spurious pressure modes. Let $\mathbb{P}_N$ denote the space of all polynomials of degree less than or equal to N. Both the KIO and the PRDI Stokes solvers are of $(\mathbb{P}_N, \mathbb{P}_N)$ type, in each space direction, since the velocity $\mathbf{u}$ and pressure p fields are approximated in the same polynomial space. A priori, a Stokes $(\mathbb{P}_N, \mathbb{P}_N)$ solver is expected to exhibit 8 (for a 2D case) or $4(N + M + L + 1)$ (for a 3D case) spurious pressure modes [5]. Their presence can be understood equivalently from

(a) an inf-sup, or Babuška–Brezzi, compatibility condition between the polynomial spaces approximating the velocity and the pressure fields [5],

(b) the algebraic compatibility conditions that must be fulfilled between the velocity boundary conditions and the incompressibility constraint enforced at all the nodes, boundaries included [19].

This latter statement allows us to predict easily the number of spurious pressure modes in each of these solvers. The KIO method does not impose a vanishing divergence at the boundaries. Then, the compatibility condition which remains does not involve the continuity constraint but concerns only the Neumann boundary conditions on the pressure, and corresponds to the constant pressure mode. For the PRDI method with (3.13)–(3.15), the incompressibility constraint is enforced, boundaries included, on a field required to satisfy only prescribed *normal* boundary conditions. The number of compatibility conditions, i.e., of spurious pressure modes, is then [19] reduced to 2^d $(d = 2, 3)$ [2]. Their space is spanned by the following set ($T_i(x)$ is the Chebyshev polynomial of first kind of degree i):

$$\begin{aligned} \{T_0, T_N(x)\} \otimes \{T_0, T_M(y)\} \qquad & \text{when } d = 2, \\ \{T_0, T_N(x)\} \otimes \{T_0, T_M(y)\} \otimes \{T_0, T_L(z)\} \qquad & \text{when } d = 3. \end{aligned}$$

These spurious pressure modes are the constant and checker-board modes.

4. Continuous-in-time evolution operator for the 2D Stokes solvers. The PRDI method is consistent with the continuous decoupled formulation (2.5)–(2.6) and leads to a genuinely continuous-in-time evolution operator (i.e., independent of any time integration scheme). Its spectrum will indicate to what extent the ellipticity of the Stokes problem [36] is preserved in the Chebyshev collocation approximation. Moreover, as in the case of the second order differentiation operator [7], the low spatial frequency part of the numerical spectrum can be taken as a good approximation of some eigenmodes of the continuous Stokes problem not yet analytically known for a bounded 2D or 3D simple Cartesian geometry, e.g., *(cavity flow)*. The 2D numerical spectrum, and some eigenmodes, will be given as complementary data to those reported in [3] and limited to the 2D case with one periodic direction, e.g., *(channel flow)*. Homogeneous Dirichlet boundary conditions are imposed on the velocity. From now on, $\mathbf{u}$ denotes the restriction of the discrete velocity field to its internal nodal values.

TABLE 2
Asymptotic laws for the extreme eigenvalues of the collocation operators.

2D $\mathcal{A}_D$		$\mathcal{L}$	
λ_{min}	λ_{max}	λ_{min}	λ_{max}
$-0.094 * N^4$	$\frac{-\pi^2}{2}$	$-0.075 * N^4$	$\frac{-\pi^2}{4}$

The determination of the full set of eigenvalues based on the explicit construction of the matrix is limited by the matrix size. Whenever possible, the whole spectrum is computed. Otherwise, the largest eigenvalues are computed by the power method adapted when the leading eigenvalue is a complex conjugate pair [30]. The convergence of the power method has been controlled by choosing several initial conditions.

Finally, the same polynomial approximation has been taken in both spatial directions. The corresponding number of collocation points is denoted by N^2.

4.1. The PRDI method. The time evolution equation of the velocity field comes directly from (3.16), (3.20), and (3.24). It reads

$$\frac{d\mathbf{u}}{dt} = \mathcal{A}_D\mathbf{u} - \mathcal{D}(\mathcal{E}_{\mathcal{PD}})^{-1}\mathcal{D}\cdot\mathbf{f}, \tag{4.1}$$

where $\mathbf{f}$ is defined as in (3.22) with $\mathbf{a} = (\mathcal{D}\times\mathcal{D}\times)\ \mathbf{u}$ (not in its time-extrapolated form (3.25)).

Then (4.1) takes the form

$$\frac{d\mathbf{u}}{dt} = \mathcal{L}\mathbf{u} \equiv \mathcal{A}_D\mathbf{u} - \mathcal{D}(\mathcal{E}_{\mathcal{PD}})^{-1}\mathcal{D}\cdot[\text{“}(\mathcal{D}\times\mathcal{D}\times\mathbf{u})\cdot\mathbf{n}\text{”}], \tag{4.2}$$

which defines the time evolution operator $\mathcal{L}$.

The temporal evolution is governed by the expected diffusion operator, completed with the contribution of the pressure gradient, itself fed by the normal boundary conditions (3.25).

The complete $\mathcal{L}$ spectrum has been computed with $N = 49$ polynomials. The results and their analysis are now reported.

Figure 1 displays the real and imaginary parts of the eigenvalues. The whole spectrum is made of eigenvalues with negative real parts, of which about 32% (scaled approximately by an $\mathcal{O}(\sqrt{N})$ law) represent complex modes. Most of these have an extremely small imaginary part, while the remainder have an imaginary part not exceeding 8.5% of the associated real part. The collocation PRDI Stokes operator has therefore the desirable property that it leads to A-stable time integration schemes (at best up to order 2 [10]).

Applying the power method with N up to 257, the algebraically smallest eigenvalue is found to be real, negative, and to scale with an N^4 law, while the largest one approaches asymptotically $\frac{-\pi^2}{4}$. The measured asymptotic laws of these extreme eigenvalues are reported in the second column of Table 2.

Although the 2D bounded Stokes problem can be solved analytically, by biorthogonal series based on “Papkovich–Fadle”-type functions [12], to the authors' knowledge no work has been published supplying the corresponding analytical eigenmodes to which the computed ones could be compared. An analysis of the Stokes eigenspace is well beyond the scope of this paper and we will limit ourselves to presenting some results, complementary to those reported in [3] for the periodic channel flow solved by a Legendre–Uzawa–Stokes scheme. First, the 11 largest eigenvalues (divided by $\frac{-\pi^2}{4}$)

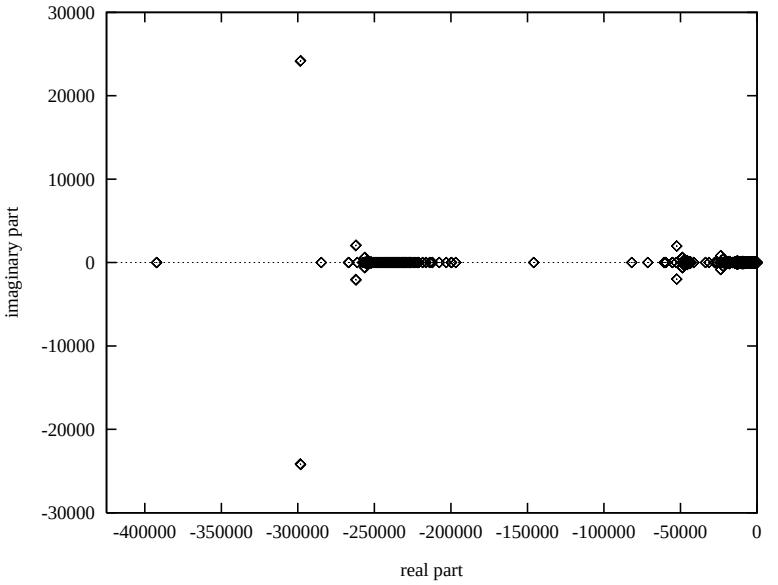


FIG. 1. *The PRDI $\mathcal{L}$ spectrum for the collocation approximation with $N = 49$.*

TABLE 3

The 11 *largest eigenvalues* (*divided by* $\frac{-\pi^2}{4}$) *for the PRDI collocation Stokes solver, with different polynomial expansions.*

$N = 7$	$N = 17$	$N = 35$	$N = 49$
1.00000800540e+00	0.99999999999e+00	1.00000000006e+00	0.99999999997e+00
1.98712673989e+00	1.99987984426e+00	1.99999713366e+00	1.99999948632e+00
4.00353214681e+00	4.00000000000e+00	4.00000000003e+00	3.99999999998e+00
5.01848849608e+00	4.99976728610e+00	4.99999430121e+00	4.99999897536e+00
5.28815867163e+00	5.30362823176e+00	5.30362606445e+00	5.30362606638e+00
8.06670784515e+00	8.00002040023e+00	8.00000010152e+00	8.00000000899e+00
8.44378285024e+00	9.00000000131e+00	9.00000000005e+00	8.99999999998e+00
9.02294796873e+00	9.33415385363e+00	9.33415263902e+00	9.33415263947e+00
9.50305383710e+00	9.99879613340e+00	9.99997135310e+00	9.99999486546e+00
9.77461792632e+00	1.00000056792e+01	1.00000000529e+01	1.00000000012e+01
1.24260709826e+01	1.29903476596e+01	1.29903468368e+01	1.29903468369e+01

are listed in Table 3 for several polynomial expansions. Since they all converge to some value (actually, an integer multiple of $\frac{-\pi^2}{4}$, except for three of them), they give a good approximation to the continuous Stokes operator eigenvalues. In fact, many numerical eigenvalues do converge with N.

Then, Figure 2 displays the leading 450 real eigenvalues λ_k (divided by $-\frac{\pi^2}{4}$) as a function of k for $N = 49$. The observed linear law agrees with the scaling estimate proposed in [9].

Finally, six of the real eigenmodes that converge with N are presented in Figure 3, thus illustrating their different structures. They correspond, respectively, to the following multiple of $\frac{-\pi^2}{4}$ eigenvalues: 1, 9, 15.616174348, 36.754375567, 64, and 276.34507007. They are candidates, among others, as eigenmodes of the continuous Stokes problem.

Once the spectra of the Stokes time-continuous evolution operator and some of its eigenmodes are presented, two related particular aspects deserve to be briefly

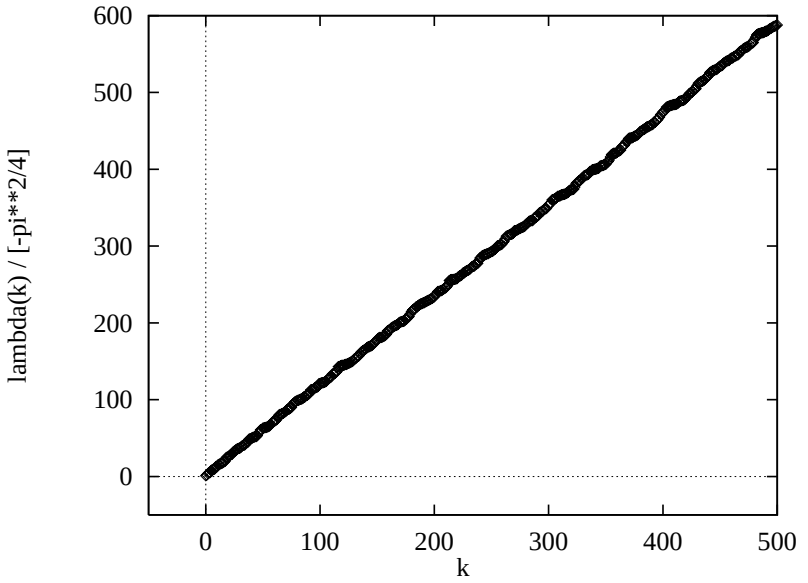


FIG. 2. *The leading* 450 *eigenvalues* (*divided by* $\frac{-\pi^2}{4}$) *for the PRDI collocation Stokes solver with* $N = 49$.

commented on.

(1) As experimentally recognized previously in [27], the implicit treatment of the $(\nabla \cdot \mathbf{u})$ term in (3.4) for the normal boundary conditions ((3.6) and (3.25)) prevents the time schemes from being numerically unstable. In an attempt to trace back the origin of this instability, the authors of [17] proposed the possible effects of "propagation and accumulation of time-differencing errors." However, it will now be shown that the source of this numerical problem is deeper than is simply due to the time discretization. Indeed, let us replace the normal boundary term ("$(\mathcal{D} \times \mathcal{D} \times \mathbf{u}) \cdot \mathbf{n}$") in the definition (4.2) of the continuous-in-time evolution operator $\mathcal{L}$ by ("$(-\mathcal{A}_D \mathbf{u}) \cdot \mathbf{n}$"). The resulting operator spectrum is given in Figure 4. Positive real parts have appeared and most of the eigenvalues are complex. This Stokes solver breaks the ellipticity of the continuous problem. Stability is therefore unreachable by any time integration scheme.

(2) Let us rewrite the $\mathcal{L}$ operator of (4.2) as $\mathcal{L} = \mathcal{A}_D + \mathcal{B}$. It must be noted that $\mathcal{B}$ handles a reduced number of independent quantities, those which are defined on the boundary nodes, $4(N-2)$ actually, as can be seen by summing the size of the $(\mathcal{E}_{\mathcal{PD}})^{-1}\mathcal{D}\cdot$ and $(\mathcal{D} \times \mathcal{D}\times) \cdot \mathbf{n}$ kernels. Therefore, the coupling of $\mathcal{B}$ with the diffusion acts in a quite reduced space. This sheds some light on the way the incompressibility constraint is actually applied (rather than strictly enforced) on the numerical velocity field. Only $4(N-2)$ independent quantities are needed to drive the solenoidality of the velocity $2(N+1)^2$ nodal values. In other words, the ability of the pressure field to ensure the incompressibility constraint depends essentially on a given reduced number of independent data, normal boundary conditions, for instance, on the pressure. This meets the influence matrix basic idea. A detailed analysis of the ($\mathcal{A}_D$, $\mathcal{B}$) coupling is reported in [21]. It brings, in particular, a way of looking at the difficulty arising with the Lanczos-τ spectral formulation (see section 5 below).

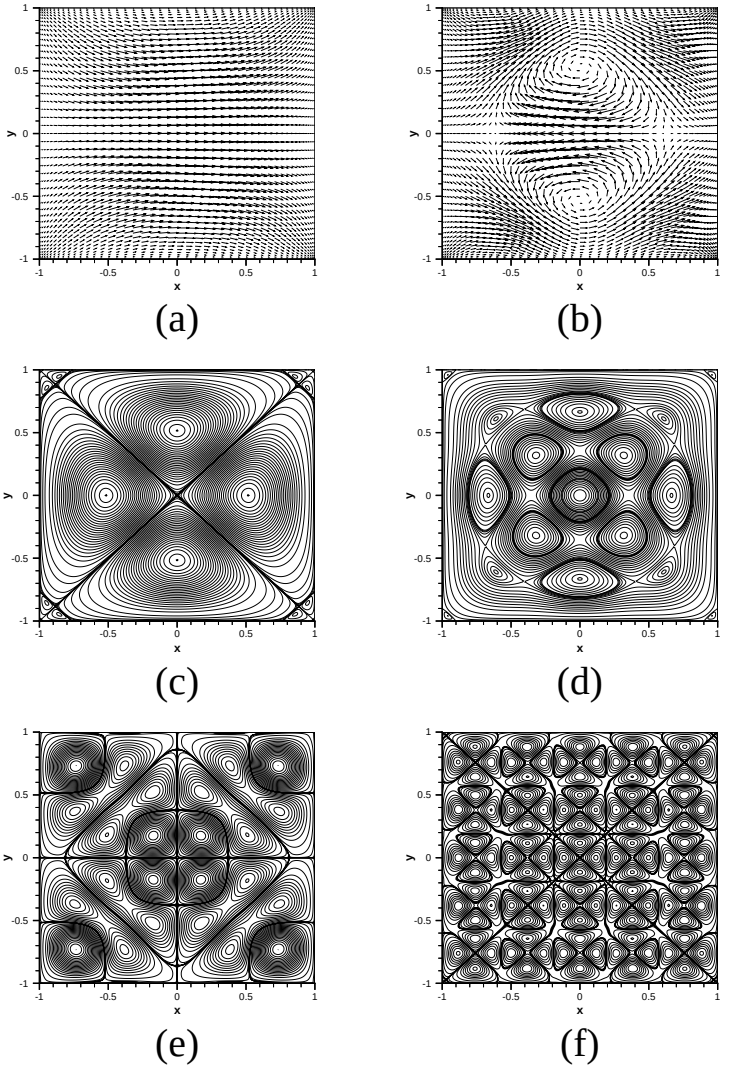


FIG. 3. *Six real eigenmodes of the PRDI $\mathcal{L}$ collocation operator with $N = 49$ corresponding to the following six eigenvalues (divided by $\frac{-\pi^2}{4}$):* (a) 1, (b) 9, (c) 15.616174348, (d) 36.754375567, (e) 64, *and* (f) 276.34507007.

4.2. The KIO method. For the PRDI solver, the time evolution equation for the discrete in space velocity field has been easily obtained. In the KIO case, because of the time splitting, this is not as straightforward. Let us start from the discretized-in-space formulation of (3.7)–(3.9):

$$\frac{d\mathbf{u}}{dt} = \mathcal{A}_D \mathbf{u} - [\mathcal{D}p]_{\{\Omega\}}\,, \tag{4.3}$$

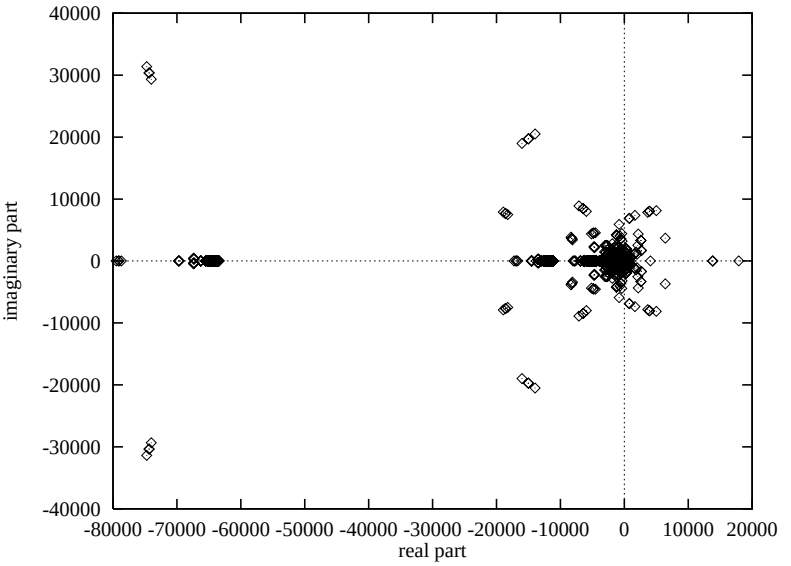


FIG. 4. *The $N = 35$ spectrum of the modified PRDI $\mathcal{L}$ operator for the collocation approximation. The $(-\mathcal{A}_D)$ term replaces the $(\mathcal{D} \times \mathcal{D}\times)$.*

with

$$[\mathcal{A}_N p]_{\{\Omega\}} = -\eta \left[\frac{d}{dt}(\mathcal{D}\cdot\mathbf{u})\right]_{\{\Omega\}}, \tag{4.4}$$

$$\left[\frac{\partial p}{\partial \mathbf{n}}\right]_{\{\partial\Omega\}} = -\left[(\mathcal{D}\times(\mathcal{D}\times\mathbf{u}))\cdot\mathbf{n}\right]_{\{\partial\Omega\}}, \tag{4.5}$$

where

(1) $\mathcal{A}_N$ is the Poisson–Neumann discrete operator,
(2) p satisfies the normal boundary conditions (4.5).

Equations (4.4)–(4.5) define the pressure as made up of two contributions, $p = p_1 + p_2$, such that

$$[\mathcal{A}_N p_1]_{\{\Omega\}} = 0, \tag{4.6}$$

$$\left[\frac{\partial p_1}{\partial \mathbf{n}}\right]_{\{\partial\Omega\}} = -\left[(\mathcal{D}\times(\mathcal{D}\times\mathbf{u}))\cdot\mathbf{n}\right]_{\{\partial\Omega\}}, \tag{4.7}$$

and

$$[\mathcal{A}_N p_2]_{\{\Omega\}} = -\eta \left[\frac{d}{dt}(\mathcal{D}\cdot\mathbf{u})\right]_{\{\Omega\}}, \tag{4.8}$$

$$\left[\frac{\partial p_2}{\partial \mathbf{n}}\right]_{\{\partial\Omega\}} = 0. \tag{4.9}$$

The p_2 contribution, induced by the splitting, forces us to take into account the temporal evolution of the velocity divergence. Each of these two pressure fields can be expressed in a concise way as

$$p_1 = -\mathcal{A}_{N1}^{-1}\left[\text{“}(\mathcal{D}\times(\mathcal{D}\times\mathbf{u})\cdot\mathbf{n})\text{”}\right], \tag{4.10}$$

TABLE 4
The 11 largest real eigenvalues (divided by $\frac{-\pi^2}{4}$) for the unsplit collocation Stokes solver based on the Poisson–Neumann pressure system, with different polynomial expansions.

$N = 7$	$N = 17$	$N = 35$	$N = 49$
1.04786449314e+00	1.00878388401e+00	1.00272485654e+00	1.00161151594e+00
2.12765504452e+00	2.03393112967e+00	2.01080401609e+00	2.00641052738e+00
4.00353214681e+00	4.00000000000e+00	4.00000000000e+00	4.00000000000e+00
5.11213178920e+00	5.01722595419e+00	5.00545338841e+00	5.00322379331e+00
5.28734082932e+00	5.30362632034e+00	5.30362606750e+00	5.30362606661e+00
8.05580355974e+00	7.99995986903e+00	7.99999927565e+00	7.99999989030e+00
8.54361599535e+00	9.00894808759e+00	9.00271135287e+00	9.00160804903e+00
9.04576061625e+00	9.33414414325e+00	9.33415453968e+00	9.33415286720e+00
9.50364060447e+00	9.99999273824e+00	1.00000000033e+01	1.00000000008e+01
1.02060636504e+01	1.00667939976e+01	1.00215687184e+01	1.00128133932e+01
1.25175033599e+01	1.29903463115e+01	1.29903468403e+01	1.29903468370e+01

$$p_2 = -\eta \mathcal{A}_{N2}^{-1} \left[\frac{d}{dt} (\mathcal{D} \cdot \mathbf{u}) \right], \tag{4.11}$$

the operator $\mathcal{A}_{N1}$ corresponding to the Poisson–Neumann problem driven only by inhomogeneous normal boundary conditions (4.7), the right-hand side being homogeneous, and $\mathcal{A}_{N2}$, to the same problem, but driven only by the right-hand side, the normal boundary conditions (4.9) being homogeneous. From this, the time evolution equation reads

$$\frac{d}{dt} \left[1 + \eta \mathcal{D} \mathcal{A}_{N2}^{-1} \mathcal{D} \cdot \right] \mathbf{u} = \left[\mathcal{A}_D \mathbf{u} + \mathcal{D} \mathcal{A}_{N1}^{-1} \text{“} (\mathcal{D} \times (\mathcal{D} \times \mathbf{u}) \cdot \mathbf{n}) \text{”} \right]. \tag{4.12}$$

In a short form, it is written as

$$\mathcal{S}(\eta) \, \frac{d\mathbf{u}}{dt} = \mathcal{L}_{PN} \mathbf{u}. \tag{4.13}$$

The right-hand side $\mathcal{L}_{PN}$ is just the analogy of the PRDI operator $\mathcal{L}$ (4.2), but from a Poisson–Neumann term instead of the quasi-Poisson one. It gives the temporal evolution of the numerical Stokes solution without splitting ($\eta = 0$). The left-hand side $\mathcal{S}(\eta)$ bears the signature of the splitting. From the knowledge of this time evolution equation, two questions are now addressed:

(1) To what extent are the Poisson-Neumann and quasi-Poisson solvers numerically equivalent?
(2) What is the long term evolution in time, given by the KIO splitting scheme, of the Stokes eigenmodes?

(1) The numerical equivalence of the Poisson–Neumann and quasi-Poisson solvers that can be established is based on the comparison of the $\mathcal{L}$ and $\mathcal{L}_{PN}$ spectra. The $\mathcal{L}_{PN}$ spectrum enjoys the properties described above (section 4.1) for the $\mathcal{L}$ operator. In particular, as shown by Table 4, the 11 largest real eigenvalues of $\mathcal{L}_{PN}$ (divided by $\frac{-\pi^2}{4}$) converge, except for 3 of them, towards an integer multiple of $\frac{-\pi^2}{4}$, but much more slowly for some of them than those of $\mathcal{L}$ (see Table 3). Furthermore, the asymptotic scaling law of the algebraically smallest eigenvalue of $\mathcal{L}_{PN}$ is in perfect agreement with the law given in Table 2.

(2) The asymptotic time evolution of the Stokes normal modes in the KIO solver. Instead of analyzing the spectrum of the evolution operator given by (4.13) for each value of the parameter η, it is more useful to characterize the long term evolution in

TABLE 5
The ranges of η for several temporal orders.

Order	1	2	3	4
η range	$[1, \infty]$	$[4, \infty]$	$[1, \infty]$	$[4.58, \infty]$

TABLE 6
The algebraically largest real eigenvalues of the generalized eigenvalue problem for several values of η, with different polynomial expansions N.

N	$\eta = 1$	$\eta = 10^1$	$\eta = 10^2$	$\eta = 10^3$	$\eta = 10^4$
7	-1.2914E+00	-2.3460E-01	-2.5547E-02	-2.5776E-03	-2.5799E-04
17	-1.2445E+00	-2.2628E-01	-2.4644E-02	-2.4866E-03	-2.4888E-04
21	-1.2413E+00	-2.2569E-01	-2.4580E-02	-2.4801E-03	-2.4824E-04
35	-1.2370E+00	-2.2492E-01	-2.4496E-02	-2.4716E-03	-2.4739E-04

time that this operator imposes to any of the Stokes normal modes. It will be shown, first, that the η parameter is not so linked to the temporal order of the scheme; and second, the splitting does not affect the stability of the scheme, i.e. any Stokes normal mode is damped through the equation (4.13).

A given Stokes normal mode is characterized by $\mathbf{u} = \mathbf{U}(x, y, z)\ \exp(\lambda t)$, with λ a real negative eigenvalue of the analytical Stokes operator, and the coefficient κ quoted in (3.10) reads as $\exp(\lambda \Delta t)$ with $0 < \kappa \leq 1$. Therefore, by relation (3.10), η is a real quantity which can be defined as well by

$$\eta = 1 - \frac{\gamma_0}{\lambda \Delta t} = -\frac{1}{\lambda \Delta t} \sum_{q=0}^{J_i - 1} \alpha_q \kappa^{-q-1}. \tag{4.14}$$

These relations are such that adding $\eta\ \frac{\partial \mathbf{u}}{\partial t}$ to $(1 - \eta)\ \frac{\partial \mathbf{u}}{\partial t}$ gives back the discrete (approximation of) $\frac{\partial \mathbf{u}}{\partial t}$. The first equation shows that η is always positive, larger than 1 actually. The allowed ranges of η are listed in Table 5.

Thus giving η any value between 1 and ∞ allows one to consider any time order of the splitting scheme; this parameter refers therefore rather to the splitting nature of the numerical system than to the chosen time order. The spectrum of the part $\left[\mathcal{D}\mathcal{A}_{N2}^{-1}\mathcal{D}\cdot\right]$ of the $\mathcal{S}(\eta)$ operator has been computed. It exhibits a large kernel (half of the complete set of the eigenvalues), the remaining ones being all positive and close to unity. The spectrum of $\mathcal{S}(\eta)$ is then always real and positive with eigenvalues $\lambda_k \approx (1 + \eta)$ for all k. This offers the possibility to the complete evolution operator (4.13) of having real negative eigenvalues for all $\eta \geq 1$. The algebraically largest real eigenvalue of the generalized eigenvalue problem (4.13) for several values of η (1 to 10000) is reported for different polynomial expansions in Table 6. Figure 5 shows the monotonic behavior of this largest eigenvalue (in absolute value) with respect to η, for a fixed $N = 21$ polynomial expansion. This eigenvalue, almost unaffected by the spatial discretization for a fixed η, remains always real, negative and goes monotonically to zero with increasing values of η. The main conclusion is that the splitting preserves the exponential decrease with time of the normal Stokes eigenmodes. The time discretization is then the only source of the possible numerical instabilities.

5. About the spectral Lanczos-τ formulation of the Stokes solvers. In this section, the first historically used spectral discretization is considered instead of the collocation approach. In this Galerkin-type method, the unknowns are the expansion coefficients in Chebyshev polynomials of the velocity and pressure fields.

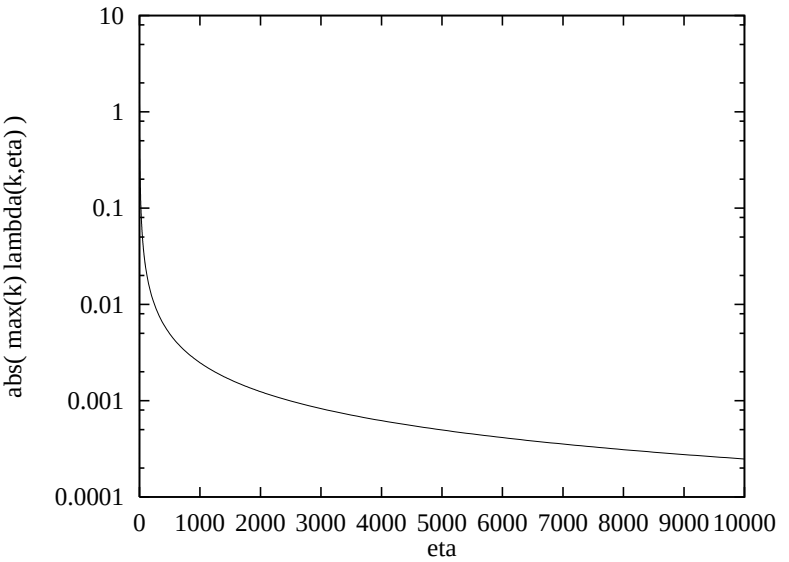


FIG. 5. *The algebraically largest real eigenvalue (in absolute value, with log-scale) of the generalized eigenvalue problem, as a function of η, with a fixed $N = 21$ polynomial expansion.*

However, the test functions (the Chebyshev polynomials themselves) do not satisfy the boundary conditions and the differential equations are enforced with a supplementary set of test functions [13].

Although commonly considered as equivalent to the collocation method, the Lanczos-τ approximation of the Stokes problem turns out to not preserve the ellipticity of the initial system. Indeed, all the previous analysis has been carried out with this spatial formulation and detailed results are reported in [21]. Only the dominant feature is quoted now, and concerns the spectrum of the time-continuous evolution operator (see (4.2)). Significantly different from the spectrum of the collocation version, it contains four eigenvalues with positive real part. The leading eigenvalue scales as $0.65 * N^4$, a result that might be related to the prediction of a very recently published analysis [11].

6. The fully discrete temporal evolution operators. Only the collocation approximation of the Stokes solvers is now considered, with a temporal discretization of order $J = J_e = J_i$ going from 1 to 4. The stability and effective order in time of their numerical solutions are addressed in this section. A 2D eigenvalue analysis and 3D numerical experiments are performed to assess these two points, respectively.

The unsteady diffusion contribution is implicitly treated with a backward Euler scheme, and the boundary condition involved in the projection stage is evaluated by an extrapolation scheme of the same order. Equations (3.1)–(3.3), (3.5)–(3.6) of section 3.1 and (3.20)–(3.26) of section 3.2 summarize the KIO and PRDI formulations, respectively.

6.1. Stability properties of the Stokes solvers. Starting from the expressions given by (4.2) and (4.13) for the PRDI and KIO continuous-in-time evolution

operators, respectively, the fully discretized evolution operators read

$$\mathbf{u}^{n+1} = (\mathcal{H})^{-1}\left[-\frac{\sum_{q=0}^{J_i}\alpha_q\mathbf{u}^{n-q}}{\Delta t} + \mathcal{D}(\mathcal{E}_{\mathcal{PD}})^{-1}\mathcal{D}\cdot\left(\sum_{q=0}^{J_e}\beta_q(\text{“}\mathcal{D}\times(\mathcal{D}\times\mathbf{u}^{n-q})\cdot\mathbf{n}\text{”})\right)\right]$$

for the PRDI method and

$$\mathbf{u}^{n+1} = (\mathcal{H})^{-1}\left[-\mathcal{I} + \mathcal{D}(\mathcal{A}_N)^{-1}\mathcal{D}\cdot\right]\left(\frac{\sum_{q=0}^{J_i-1}\alpha_q\mathbf{u}^{n-q}}{\Delta t}\right)$$

for the KIO solver. $\mathcal{H}$ is the discrete Helmholtz operator, which includes the homogeneous Dirichlet boundary conditions imposed on the velocity,

$$\mathcal{H} = \mathcal{A}_D - \frac{\gamma_0}{\Delta t}\mathcal{I},$$

$\mathcal{I}$ is the unit matrix, and $\mathcal{E}_{\mathcal{PD}}$ is the quasi-Poisson operator arising from the decoupling of the fields $\mathbf{a}^*$ and the pressure (see (3.24)). $\mathcal{A}_N$ is the Neumann–Poisson discrete operator, completed with the normal boundary condition

$$\left[\frac{\partial p}{\partial \mathbf{n}}\right]_{\{\partial\Omega\}}^{n+1} = \left[\text{“}\left(-\sum_{q=0}^{J_e-1}\beta_q\mathcal{D}\times(\mathcal{D}\times\mathbf{u}^{n-q})\right)\cdot\mathbf{n}\text{”}\right]_{\{\partial\Omega\}}.$$

These matrices are full. Their size is $[dJ(N-1)^d]^2$, where d is the space dimensionality. Thus, their complete spectrum evaluation has been limited to the 2D case with a reduced number of nodes, namely, $N = 33$ in each space direction for the first order in time discretization, and $N = 21$ otherwise. For larger N values the power method is applied to get the leading eigenvalues.

The spectra of the PRDI and KIO matrices are displayed in Figures 7 and 8 for the first to fourth order in time schemes, with time steps going from 10^{-3} to 0.1. In [21], these spectra are given for a broader range of time steps. An increase in the time order has the noticeable effect of taking the eigenvalues away from the real axis, often pushing some of them outside the unit circle when the time order reaches the third level. This effect is more pronounced with the KIO solver, as can be seen by comparing, for instance, the first order in time spectra on Figures 7 and 8, almost real with the PRDI solver and significantly complex with the KIO version. Hence, the PRDI solver is more stable for the third and fourth order in time schemes. Furthermore, both solvers are unconditionally stable with the first and second order in time schemes. This is summarized in Figure 6 which depicts, as a function of the time step, the leading eigenvalue moduli for both Stokes solvers when discretized with the four temporal schemes. The horizontal line where all the data converge, for very small time steps, is slightly below the unit value. The estimated stability criterion $\Delta t(N)$, of which some numerical values are given in Table 7, is $\Delta t(N) \simeq \mathcal{O}(N^{-4})$. This is the signature of the explicit treatment of the boundary condition used in the pressure stage.

Fixing the time step at the value 10^{+7}, the largest modulus of the eigenvalues for the first- to fourth-order time evolution operators is determined for several space discretizations N, going from $N = 7$ to $N = 129$ and given in Table 8.

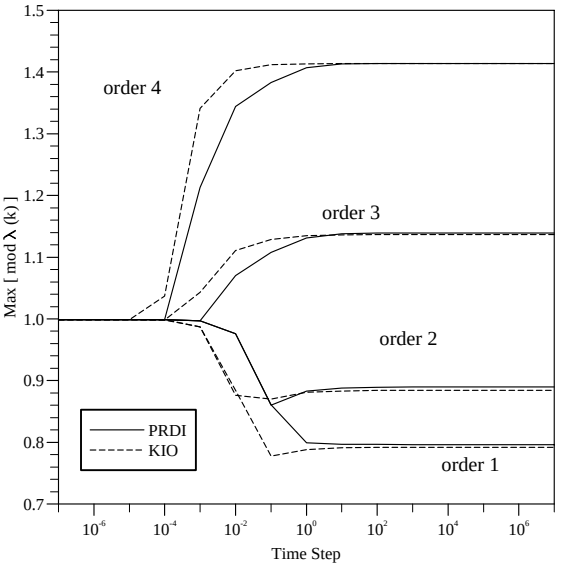


FIG. 6. *Evolution of the largest eigenvalues λ_{max} of the first- to fourth-order time evolution operators for the PRDI and KIO Stokes solvers, with the time steps going from 10^{-7} to 10^{+7} ((*mod*) means the largest eigenvalue in modulus; the number of Chebyshev collocation points in each direction is 33 for the first-order scheme, and 21 for the higher-order schemes).*

TABLE 7
The time steps Δt giving the stability limit ($|\lambda_{max}| \approx 1$) of the third- and fourth-order time schemes for the PRDI and the KIO Stokes solvers for several spatial discretizations N.

N	Order 3		Order 4	
	PRDI	KIO	PRDI	KIO
7	$1.31\ 10^{-1}$	$5.97\ 10^{-2}$	$2.10\ 10^{-2}$	$8.77\ 10^{-3}$
17	$4.05\ 10^{-3}$	$1.34\ 10^{-3}$	$5.31\ 10^{-4}$	$2.03\ 10^{-4}$
33	$2.51\ 10^{-4}$	$8.57\ 10^{-5}$	$3.32\ 10^{-5}$	$1.29\ 10^{-5}$
65	$1.56\ 10^{-5}$	$5.35\ 10^{-6}$	$2.09\ 10^{-6}$	$8.12\ 10^{-7}$
129	$9.72\ 10^{-7}$	$3.31\ 10^{-7}$	$1.30\ 10^{-7}$	$5.10\ 10^{-8}$

6.2. The effective temporal order of the Stokes solvers. A 3D test problem (inspired by the test adopted in [4]) is considered, in the cube $[-1,1]^3$, with the source term $\mathbf{f}$ of (2.1) analytically chosen so that the following velocity $\mathbf{u} = (u, v, w)$ and pressure p fields are solutions of the Stokes problem (2.1)–(2.2)

$$\begin{aligned}
u(x,y,z,t) &= -\ 0.5\sqrt{3}\sqrt{7}\sin(\sqrt{2}x+t)\cos(\sqrt{3}y+t)\cos(\sqrt{7}z+t),\\
v(x,y,z,t) &= -\ 0.5\sqrt{2}\sqrt{7}\cos(\sqrt{2}x+t)\sin(\sqrt{3}y+t)\cos(\sqrt{7}z+t),\\
w(x,y,z,t) &= \sqrt{2}\sqrt{3}\cos(\sqrt{2}x+t)\cos(\sqrt{3}y+t)\sin(\sqrt{7}z+t),\\
p(x,y,z,t) &= \sqrt{6}\sin(2x-\sqrt{5}y+\sqrt{10}z+0.7t)\sin(\sqrt{5}y++\sqrt{11}z+0.3t).
\end{aligned}$$

Note that the boundary values of the velocity are time dependent. The chosen time dependence of this test case is periodic, instead of exponentially decreasing, to allow for a nonbiased experimental confirmation of the stability criteria and of the effective temporal orders. Moreover, $N = 35$ nodes are taken for evaluating the space derivatives with enough accuracy (almost at the machine limit) to get only temporal errors in the numerical data. The Stokes problem has been integrated with the already

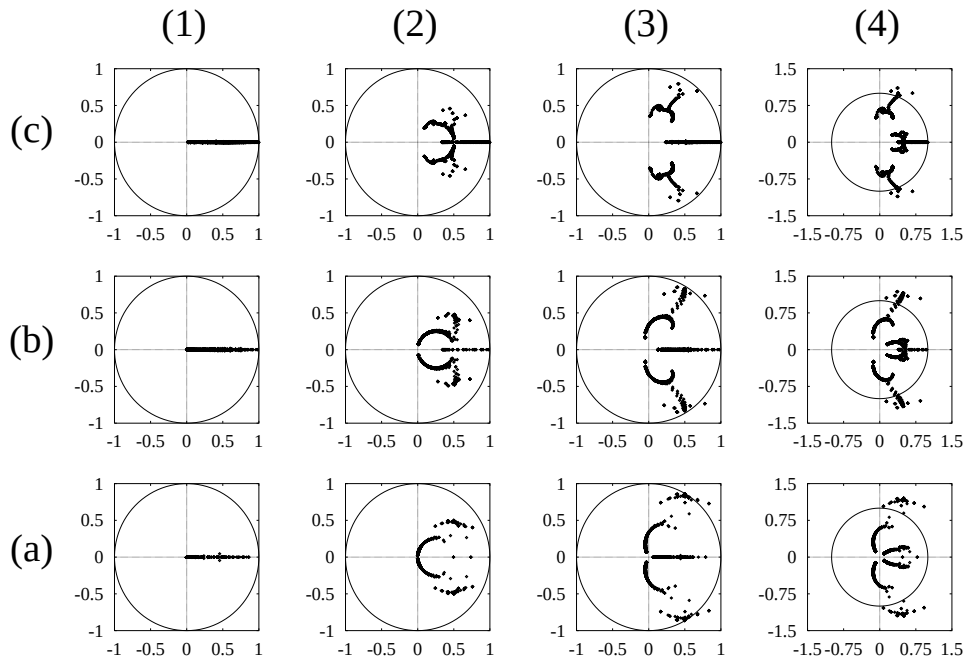


FIG. 7. *Spectra of the first- to the fourth-order time evolution operator for several time steps going from* 10^{-1} (a) *to* 10^{-3} (c) *(multiplicative increment of* 10*), with* $N = 33$ *collocation points in each direction for the first order, and* $N = 21$ *for the higher orders, PRDI method.*

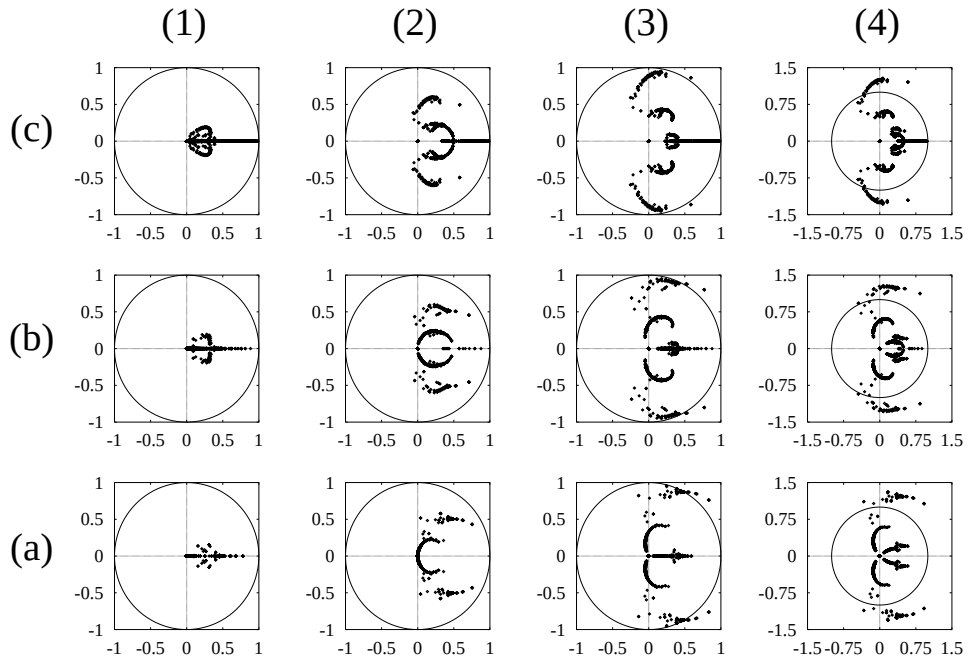


FIG. 8. *Spectra of the first- to the fourth-order time evolution operator for several time steps going from* 10^{-1} (a) *to* 10^{-3} (c) *(multiplicative increment of* 10*), with* $N = 33$ *collocation points in each direction for the first order, and* $N = 21$ *for the higher orders, KIO method.*

TABLE 8

Evolution with the spatial discretization N of the largest eigenvalue of the first- to fourth-order time evolution operators for the PRDI and KIO Stokes solvers. The time step Δt is fixed at the value 10^{+7}. (mod) *means the largest eigenvalue in modulus.*

	Order 1		Order 2 (mod)		Order 3 (mod)		Order 4 (mod)	
N	PRDI	KIO	PRDI	KIO	PRDI	KIO	PRDI	KIO
7	0.745	0.707	0.863	0.841	1.125	1.111	1.409	1.402
17	0.786	0.774	0.887	0.880	1.138	1.135	1.414	1.413
33	0.797	0.792	0.893	0.890	1.141	1.140	1.414	1.414
65	0.803	0.802	0.896	0.895	1.143	1.142	1.414	1.430
129	0.807	0.801	0.898	0.899	1.144	1.160	1.414	1.453

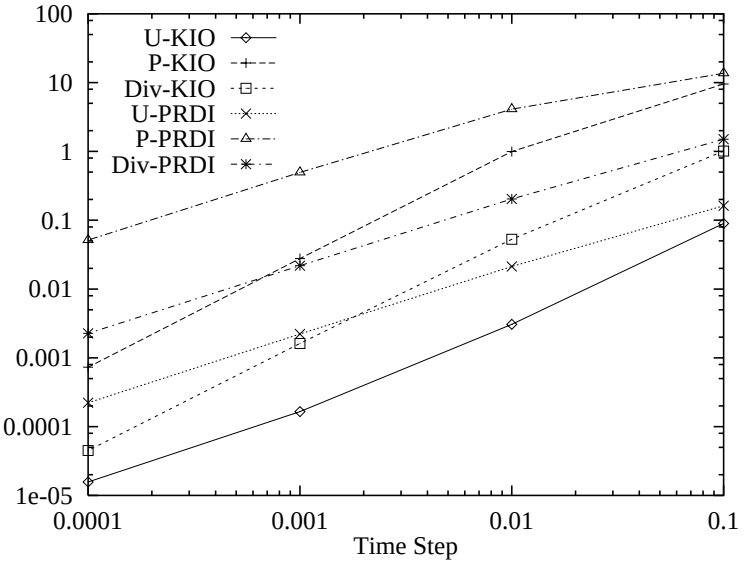


FIG. 9. *Maximum pointwise error of the computed velocity (U), the pressure (P), and* $\nabla \cdot \mathbf{u}$ (Div) *with respect to the corresponding analytical quantities for the PRDI and KIO Stokes solvers, first order time scheme.*

quoted first and second order in time schemes, during one unit of dimensionless time, with time steps ranging from 10^{-4} to 10^{-1} and the pointwise errors measured. The maximum pointwise difference obtained between the numerical and analytical velocity, the pressure, and $\nabla \cdot \mathbf{u}$ fields are plotted on Figures 9 and 10 for the first and second temporal orders, respectively. The PRDI solver supplies exactly the expected order, while the effective order coming from the KIO splitting is nonuniform on the chosen range of Δt. The expected order is effectively reached for very small time steps, $\Delta t < 10^{-3}$ actually, and beyond this approximate threshold the effective order gets higher by $\frac{1}{2}$ with respect to its expected level. For this particular test, the time accuracy is systematically in favor of the KIO method.

7. Some comparative data with the Uzawa approach. To decouple the velocity and pressure fields by the Uzawa method is, to a certain extent, a reference approach for solving accurately the Stokes problem. This has previously been attested to, for instance, by Batcho and Karniadakis [3] who choose this solver to compute the Stokes eigenmodes for the channel flow. Nonetheless, this method is very expensive when the diffusion terms are implicitly evaluated with time. The authors' point of

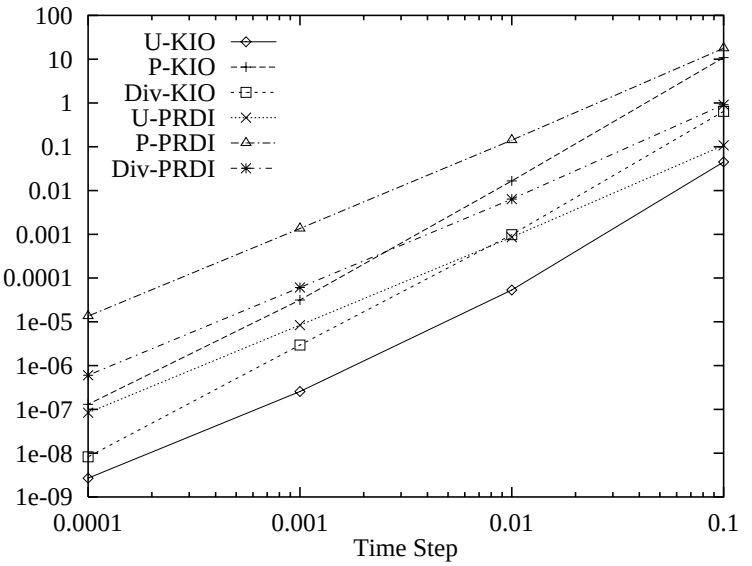


FIG. 10. *Maximum pointwise error of the computed velocity (U), the pressure (P), and* $\nabla \cdot \mathbf{u}$ (Div) *with respect to the corresponding analytical quantities for the PRDI and KIO Stokes solvers, second-order time scheme.*

TABLE 9
The operation count for the PRDI and the preconditioned Uzawa Stokes solvers.

Uzawa	$20\ d\ \mathcal{C}\mathcal{N}_{it}$
PRDI	$3\ d\ \mathcal{C}$

view is that the PRDI decoupling is an interesting alternative to get equivalent results [20] at a significantly reduced cost. This has made it possible to perform direct numerical simulation of the 3D lid-driven cavity at high Reynolds number [24, 23]. A few comparative numerical results are therefore given in this section, for the PRDI and the unique grid $(\mathbb{P}_N, \mathbb{P}_{N-2})$ Uzawa solvers.

First, for the operation count, let us introduce $\mathcal{C}$, the number of operations required to perform a matrix multiplication on a vector-data containing N^d elements, N in each direction of the dD space. Taking advantage of the tensorial structure of the matrices, one has $\mathcal{C} = N^{d+1}$.

Table 9 gives a coarse estimation of the operation count for both solvers, where $\mathcal{N}_{it}$ stands for the iteration number required to converge the Uzawa solution to machine accuracy. Indeed, an iterative method is employed in order to avoid an explicit construction of the Uzawa matrix. An efficient Navier–Stokes solver (with a biconjugate gradient [37] and a good preconditioner) has thus been implemented for moderate or high Reynolds number flow simulation, requiring small time steps because of the explicit treatment of the nonlinear terms [22].

$\mathcal{N}_{it}$ is very sensitive to the physical configuration (the Reynolds number values for instance), to the number of collocation points and the time step size [22]. As an illustration, for the two-dimensional case (to be presented hereafter), Table 10 gives $\mathcal{N}_{it}$ for several polynomial orders.

Finally, the PRDI and the preconditioned Uzawa–Stokes solvers are compared in a benchmark Navier–Stokes application, the 2D regularized lid-driven square cavity

TABLE 10
$\mathcal{N}_{it}(N, \Delta t)$ *for the* $Re = 1000$ *two-dimensional regularized lid-driven square cavity.*

N	17	35	65	129
Δt	10^{-2}	10^{-2}	$5\ 10^{-3}$	$5\ 10^{-3}$
$\mathcal{N}_{it}$	8	26	51	187

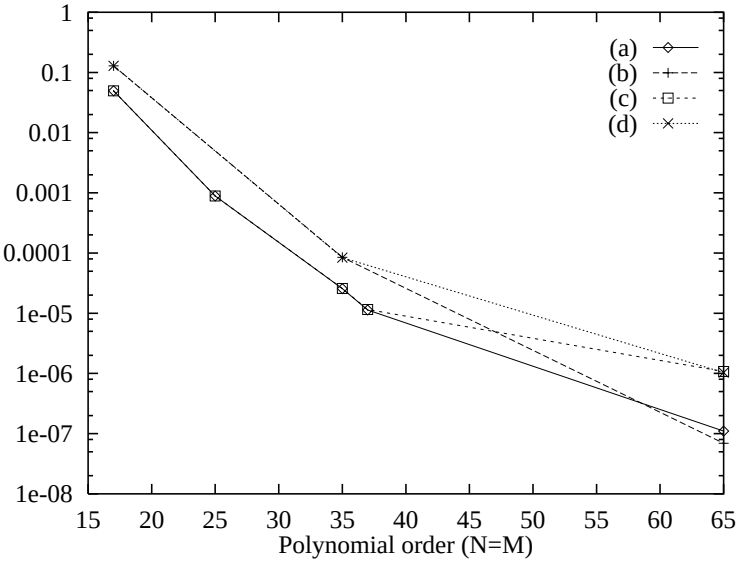


FIG. 11. *Maximum pointwise difference between the computed velocity and the reference solution:* (a) *Uzawa solutions, reference Uzawa* 129^2, (b) *PRDI solutions, reference Uzawa* 129^2, (c) *Uzawa solutions, reference PRDI* 129^2, *and* (d) *PRDI solutions, reference PRDI* 129^2.

$[0, 1]^2$ at a Reynolds value of 1000. In order to avoid the top-corners singularities, the flow is driven by a horizontal regularized velocity profile, $u(x, y = +1) = 16x^2(1-x)^2$ [32, 4, 22]. On all other sides of the cavity a zero velocity no-slip boundary condition applies. At that Reynolds number the flow is steady [32]. Both Navier–Stokes solvers have a second order accuracy in time. The steady solution is reached when the following convergence criterion is fulfilled:

$$\max_{\substack{0 \le n \le N \\ 1 \le i \le 2}} \left| \frac{u_i^{(t+\Delta t)}(x_n, y_n) - u_i^{(t)}(x_n, y_n)}{u_i^{(t)}(x_n, y_n)\ \Delta t} \right| \le 10^{-3},$$

where u_i is the ith velocity component and (x_n, y_n) are the coordinates of the collocation points. Owing to the absence of an analytical solution for this problem, the solutions based on 129^2 collocation points with the Uzawa and the PRDI solvers have been used as the references for calculating the errors. Both algorithms exhibit comparable error drops as the order of the polynomial expansion is increased. Figure 11 shows the maximum pointwise difference between the computed velocity and the reference solutions. The maximum pointwise values of the divergence of the computed velocity (in $\mathbb{P}_N$ space) are reported in Figure 12.

8. Conclusions. Projecting the complete acceleration field $\mathbf{a} = \frac{\partial \mathbf{u}}{\partial t} - \Delta \mathbf{u}$, rather than some part of it, onto the space of divergence-free polynomials, leads to a direct solver of the unsteady Stokes problem. This is consistent with the expected decoupled

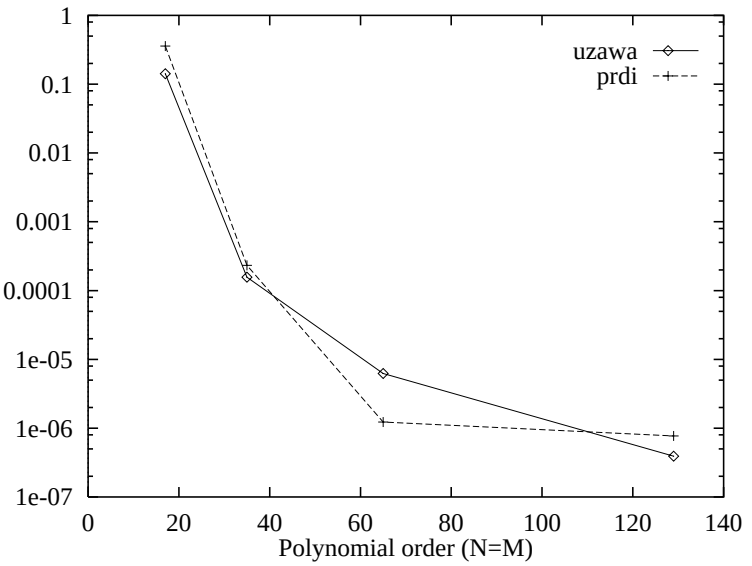


FIG. 12. *Maximum pointwise value of the divergence of the computed velocity for the PRDI and the Uzawa solvers.*

laws which govern the harmonic pressure and biharmonic velocity fields. It is as easily invertible as any other solver based on a time splitting of the velocity and pressure operators.

The main properties of its Chebyshev collocated approximations have been analyzed, comparatively to those of the fractional step approach with the ultimate improvements proposed by [17].

The basic ellipticity of the continuous Stokes system is preserved to a large extent (i.e., most of the eigenvalues are real and negative) with the new solver.

As already encountered more than ten years ago in [27], the trace of $\mathbf{a}$ on the boundary must be evaluated at the projection stage, while the velocity field has not yet been updated in time. This is the only place where the time scheme must intervene in our decoupling approach. It is shown, in this paper, that the condition to preserve the ellipticity is to reduce the $\Delta\mathbf{u}$ term to its rotational part. Otherwise, the *continuous* temporal evolution of the discrete velocity explodes exponentially.

The time explicit treatment of the rotational boundary term does not prevent the first and second order in time schemes from being unconditionally stable. However the third and fourth order schemes obey the usual explicit criterion ($\mathcal{O}(N^{-4})$) on the time step. The proposed decoupling gives this limitation a less restrictive coefficient and supplies the expected temporal orders across all the explored range of time step sizes.

Last, the effective accuracy obtained for Navier–Stokes 2D solutions has been measured with the $Re = 1000$ lid-driven cavity problem and found equivalent to what is supplied by the much more expensive Uzawa decoupling method.

Acknowledgments. The authors would like to thank Prof. M. O. Deville for helpful discussions and Dr. N. Borhani and Dr. R. Owens for proofreading the manuscript.

REFERENCES

[1] K. Arrow, L. Hurwicz, and H. Uzawa, *Studies in Nonlinear Programming*, Stanford University Press, Stanford, CA, 1968.

[2] M. Azaïez, C. Bernardi, and M. Grundmann, *Spectral methods applied to porous media equations*, East-West J. Numer. Math., 2 (1994), pp. 91–105.

[3] P. Batcho and G. Karniadakis, *Generalized Stokes eigenfunctions: A new trial basis for the solution of the incompressible Navier-Stokes equations*, J. Comput. Phys., 115 (1994), pp. 121–146.

[4] A. Batoul, H. Khallouf, and G. Labrosse, *Une méthode de résolution directe (pseudo-spectrale) du problème de Stokes 2D/3D instationnaire. Application à la cavité entrainée carrée*, C. R. Acad. Sci. Paris Sér. I Math., 319 (1994), pp. 1455–1461.

[5] C. Bernardi, C. Canuto, and Y. Maday, *Generalized inf-sup conditions for Chebyshev spectral approximation of the Stokes problem*, SIAM J. Numer. Anal., 25 (1988), pp. 1237–1271.

[6] C. Bernardi, C. Canuto, Y. Maday, and B. Metivet, *Single-grid spectral collocation for the Navier-Stokes equations*, IMA J. Numer. Analysis, 10 (1990), pp. 253–297.

[7] C. Canuto, M. Hussaini, A. Quarteroni, and T. Zang, *Spectral Methods in Fluid Dynamics*, Springer Ser. Comput. Phys., Springer-Verlag, New York, 1988.

[8] A. Chorin, *Numerical solution of the Navier-Stokes equations*, Math. Comp., 22 (1968), pp. 745–761.

[9] P. Constantin and C. Foias, *Navier-Stokes Equations*, Chicago Lectures in Math., University of Chicago Press, Chicago, IL, 1988.

[10] G. Dahlquist, *A special stability problem for linear multistep methods*, BIT, 3 (1963), pp. 27–43.

[11] P. Dawkins, S. Dunbar, and R. Douglass, *The origin and nature of spurious eigenvalues in the spectral tau method*, J. Comput. Phys., 147 (1998), pp. 441–462.

[12] P. Gaskell, J. Summers, H. Thompson, and M. Savage, *Creeping flow analyses of free surface cavity flows*, Theoret. Comput. Fluids Dynamics, 8 (1996), pp. 415–433.

[13] D. Gottlieb and S. Orszag, *Numerical Analysis of Spectral Methods: Theory and Applications*, CBMS-NSF Regional Conf. Ser. in Appl. Math. 26, SIAM, Philadelphia, 1977.

[14] J. Guermond and L. Quartapelle, *On stability and convergence of projection methods based on pressure Poisson equation*, Internat. J. Numer. Methods Fluids, 26 (1998), pp. 1039–1053.

[15] P. Haldenwang, G. Labrosse, S. Abboudi, and M. Deville, *Chebyshev 3D spectral and 2D pseudospectral solvers for the Helmholtz equation*, J. Comput. Phys., 55 (1984), pp. 115–128.

[16] J. G. Heywood and R. Rannacher, *Finite element approximation of the nonstationary Navier-Stokes problem.* I. *Regularity of solutions and second-order error estimates for spatial discretization*, SIAM J. Numer. Anal., 19 (1982), pp. 275–311.

[17] G. Karniadakis, M. Israeli, and S. Orszag, *High-order splitting methods for the incompressible Navier-Stokes equations*, J. Comput. Phys., 97 (1991), pp. 414–443.

[18] L. Kleiser and U. Schumann, *Treatment of the incompressibility and boundary conditions in 3-D numerical spectral simulation of plane channel flows*, in Proceedings of the 3rd GAMM Conf. on Numerical Methods in Fluid Mechanics, E. Hirschel, ed., Vieweg, Braunschweig, 1980, p. 165–173.

[19] G. Labrosse, *Compatibility conditions for the Stokes system discretized in 2D Cartesian domains*, Comput. Methods Appl. Mech. Engrg., 106 (1993), pp. 353–365.

[20] G. Labrosse, E. Tric, H. Khallouf, and M. Betrouni, *A direct (pseudo-spectral) solver of the 2D/3D Stokes problem: Transition to unsteadiness of natural convection flow in a differentially heated cubical cavity*, Num. Heat Transfer, 31 (1997), pp. 261–276.

[21] E. Leriche, *Direct Numerical Simulation of a Lid-Driven Cavity Flow by a Chebyshev Spectral Method*, Ph.D. thesis 1932, Ecole Polytechnique Fédérale de Lausanne, Lausanne, Switzerland, 1999.

[22] E. Leriche and M. Deville, *Uzawa-type pressure solver for the Chebyshev-tau spectral method*, Computers and Fluids, submitted.

[23] E. Leriche and S. Gavrilakis, *Direct numerical simulation of the flow in a lid-driven cubical cavity*, Phys. Fluids, 12 (2000), pp. 1363–1376.

[24] E. Leriche, S. Gavrilakis, and M. Deville, *Direct simulation of the lid-driven cavity flow with Chebyshev polynomials*, in Proceedings of the 4th European Computational Fluid Dynamics Conference (ECCOMAS), K. D. Papaillou, D. Tsahalis, J. Periaux, C. Hirsch, and M. Pandolfi, eds., John Wiley and Sons, New York, 1998, pp. 220–225.

[25] Y. Maday and C. Bernardi, *Approximations spectrales de problèmes aux limites elliptiques*,

Math. Appl. 10, Springer-Verlag, Paris, 1992.
[26] Y. Maday, D. Meiron, A. T. Patera, and E. M. Rønquist, *Analysis of iterative methods for the steady and unsteady Stokes problem: Application to spectral element discretizations*, SIAM J. Sci. Comput., 14 (1993), pp. 310–337.
[27] S. Orszag, M. Israeli, and M. Deville, *Boundary conditions for incompressible flows*, J. Sci. Comput., 1 (1986), pp. 75–111.
[28] J. Perot, *An analysis of the fractional step method*, J. Comput. Phys., 108 (1993), pp. 51–58.
[29] A. Quarteroni and A. Valli, *Numerical Approximation of Partial Differential Equations*, Springer Ser. Comput. Math. 23, Springer-Verlag, New York, 1994.
[30] Y. Saad, *Numerical Methods for Large Eigenvalue Problems*, Manchester University Press, Manchester, UK, 1992.
[31] M. Schumack, W. Schultz, and J. Boyd, *Spectral method solution of the Stokes equations on nonstaggered grids*, J. Comput. Phys., 94 (1991), pp. 30–58.
[32] J. Shen, *Hopf bifurcation of the unsteady regularized driven cavity flow*, J. Comput. Phys., 95 (1991), pp. 228–245.
[33] J. Shen, *On error estimates of the projection methods for the Navier-Stokes equations: Second-order schemes*, Math. Comp., 65 (1996), pp. 1039–1065.
[34] R. Temam, *Sur l'approximation de la solution de Navier-Stokes par la méthode des pas fractionnaires.* I, Arch. Rational Mech. Anal., 32 (1969), pp. 135–153.
[35] R. Temam, *Sur l'approximation de la solution de Navier-Stokes par la méthode des pas fractionnaires,* II, Arch. Rational Mech. Anal., 32 (1969), pp. 377–385.
[36] R. Temam, *Navier-Stokes Equations: Theory and Numerical Analysis*, 3rd ed., Stud. Math. Appl. 2, North-Holland, Amsterdam, 1984.
[37] H. Van Der Vorst, *BI-CGSTAB: A fast and smoothly converging variant of BI-CG for the solution of nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 13 (1992), pp. 631–644.

SIAM J. SCI. COMPUT.
Vol. 22, No. 4, pp. 1411–1443
© 2000 Society for Industrial and Applied Mathematics

A NEW PARADIGM FOR PARALLEL ADAPTIVE MESHING ALGORITHMS *

RANDOLPH E. BANK† AND MICHAEL HOLST†

Abstract. We present a new approach to the use of parallel computers with adaptive finite element methods. This approach addresses the load balancing problem in a new way, requiring far less communication than current approaches. It also allows existing sequential adaptive PDE codes such as PLTMG and MC to run in a parallel environment without a large investment in recoding. In this new approach, the load balancing problem is reduced to the numerical solution of a small elliptic problem on a single processor, using a sequential adaptive solver, without requiring any modifications to the sequential solver. The small elliptic problem is used to produce a posteriori error estimates to predict future element densities in the mesh, which are then used in a weighted recursive spectral bisection of the initial mesh. The bulk of the calculation then takes place independently on each processor, with no communication, using possibly the same sequential adaptive solver. Each processor adapts its region of the mesh independently, and a nearly load-balanced mesh distribution is usually obtained as a result of the initial weighted spectral bisection. Only the initial fan-out of the mesh decomposition to the processors requires communication. Two additional steps requiring boundary exchange communication may be employed after the individual processors reach an adapted solution, namely, the construction of a global conforming mesh from the independent subproblems, followed by a final smoothing phase using the subdomain solutions as an initial guess. We present a series of convincing numerical experiments which illustrate the effectiveness of this approach. The justification of the initial refinement prediction step, as well as the justification of skipping the two communication-intensive steps, is supported by some recent [J. Xu and A. Zhou, *Math. Comp.*, to appear] and not so recent [J. A. Nitsche and A. H. Schatz, *Math. Comp.*, 28 (1974), pp. 937–958; A. H. Schatz and L. B. Wahlbin, *Math. Comp.*, 31 (1977), pp. 414–442; A. H. Schatz and L. B. Wahlbin, *Math. Comp.*, 64 (1995), pp. 907–928] results on local a priori and a posteriori error estimation.

Key words. adaptivity, finite element methods, a posteriori error estimation, parallel computing

AMS subject classifications. 65M55, 65N55

PII. S1064827599353701

1. Introduction.

One of the most difficult obstacles to overcome in making effective use of parallel computers for adaptive finite element codes such as PLTMG [2] and MC [15] is the load balancing problem. As an adaptive method adjusts the mesh according to the features of the solution, elements in some areas are refined, whereas others are not. If an initial mesh is distributed quite fairly among a number of processors, a very good error estimator (coupled with adaptive refinement) quickly produces a very bad work load imbalance among the processors.

A number of static and dynamic load balancing approaches for unstructured meshes have been proposed in the literature [12, 13, 14, 17, 30, 33]; most of the dynamic strategies involve repeated application of a particular static strategy. One of the difficulties in all of these approaches is the amount of communication that must be performed both to assess the current load imbalance severity, and to redistribute the work among the processors once the imbalance is detected and an improved distribution is calculated. The calculation of the improved work distribution can be quite

*Received by the editors April 7, 1999; accepted for publication (in revised form) December 6, 1999; published electronically November 2, 2000. The work of the first author was supported by the National Science Foundation under contract DMS-9706090. The work of the second author was supported by the National Science Foundation under CAREER award 9875856.

http://www.siam.org/journals/sisc/22-4/35370.html

†Department of Mathematics, University of California at San Diego, La Jolla, CA 92093 (rbank@ucsd.edu, mholst@math.ucsd.edu).

inexpensive (such as geometric or inertia tensor-based methods), or it may be a costly procedure, with some approaches requiring the solution of an associated eigenvalue problem or evolution of a heat equation to near equilibrium [34]. These calculations may themselves require communication if they must be solved in parallel using the existing (poor) distribution.

In recent years, clusters of fast workstations have replaced the more traditional parallel computer of the past. While this type of parallel computer is now within reach of an organization with even a modest hardware budget, it is usually difficult to produce an efficient parallel implementation of an elliptic PDE solver; this is simply due to the fact that elliptic continuum mechanics problems necessarily lead to tightly coupled discrete problems, requiring substantial amounts of communication for their solution. The load balancing problem is also more pronounced on workstation clusters: even at 100 Mbit/sec speed, the cluster communication speeds are quite slow compared to modern workstation CPU performance, and the communication required to detect and correct load imbalances results in severe time penalties.

1.1. A new approach to parallel adaptive finite element methods. In this work, we present an alternative approach which addresses the load balancing problem in a new way, requiring far less communication than current approaches. This approach also allows existing sequential adaptive PDE codes such as PLTMG and MC to run in a parallel environment without a large investment in recoding.

Our approach has three main components:

1. We solve a small problem on a coarse mesh, and use a posteriori error estimates to partition the mesh. Each subregion has approximately the same error, although subregions may vary considerably in terms of numbers of elements or grid points.
2. Each processor is provided the complete coarse mesh and instructed to sequentially solve the *entire* problem, with the stipulation that its adaptive refinement should be limited largely to its own partition. The target number of elements and gridpoints for each problem is the same.
3. A final mesh is computed using the union of the refined partitions provided by each processor. This mesh is regularized and a final solution computed using a standard domain decomposition or parallel multigrid technique.

The above approach has several interesting features. First, the load balancing problem (step 1) is reduced to the numerical solution of a small elliptic problem on a single processor, using a sequential adaptive solver such as PLTMG or MC, without requiring any modifications to the sequential solver. Second, the bulk of the calculation (step 2) takes place independently on each processor and can also be performed with a sequential solver such as PLTMG or MC with no modifications necessary for communication. (In PLTMG, one line of code was added, which artificially multiplied a posteriori error estimates for elements outside a processor's partition by 10^{-6}. In MC, two lines were added to prevent elements outside the processor's partition from entering the initial refinement queue.) Step 2 was motivated by recent work of Mitchell [20, 21, 22] on parallel multigrid methods. A similar approach appeared recently in [10]. The use of a posteriori error estimates in mesh partitioning strategies has also been considered in [26].

The only parts of the calculation requiring communication are (1) the initial fanout of the mesh distribution to the processors, once the decomposition is determined by the error estimator; (2) the mesh regularization, requiring local communication to produce a global conforming mesh; and (3) the final solution phase, which might

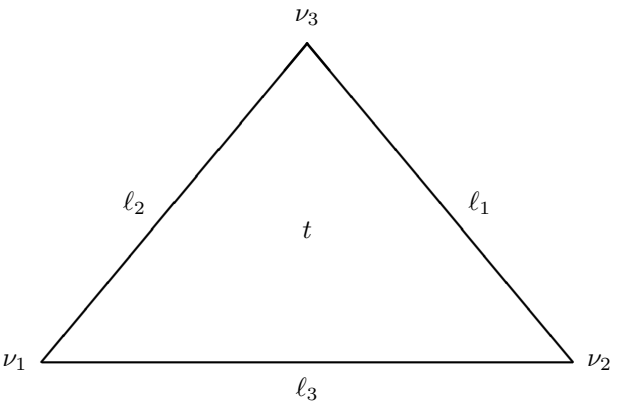


FIG. 1. *A typical element.*

require local communication (boundary exchanges). Note that a good initial guess for step 3 is provided in step 2 by taking the solution from each subregion restricted to its partition. Note also that the initial communication step to fan-out the mesh decomposition information is not actually required, since each processor can compute the decomposition independently (with no communication) as a preprocessing step.

1.2. Justification. Perhaps the largest issue arising in connection with this procedure is whether it is well founded, particularly in light of the continuous dependence of the solution of an elliptic equation on data throughout the domain. To address this issue, we first note that the primary goal of step 2 above is adaptive mesh generation. In other words, the most important issue is not how accurately the problem is solved in step 2 of this procedure, but rather the quality of the (composite) adaptively generated mesh. These two issues are obviously related, but one should note that it is not necessary to have an accurate solution in order to generate a well adapted mesh. Indeed, the ability to generate good meshes from relatively inaccurate solutions explains the success of many adaptive methods.

A secondary goal of step 2 is the generation of an initial guess for the solution on the final composite mesh. This aspect of the algorithm will be addressed in section 4. Here we focus on the primary issue of grid generation, and in particular on a posteriori error estimates, as such estimates provide the link between the computed solution and the adaptive meshing procedure. Here we consider in detail the schemes used in PLTMG and MC, but similar points can be made in connection with other adaptive algorithms.

PLTMG uses a discretization based on continuous piecewise linear triangular finite elements. The error is approximated in the subspace of discontinuous piecewise quadratic polynomials that are zero at the vertices of the mesh. In particular, let $u - u_h$ denote the error and let t denote a generic triangle in the mesh. In its adaptive algorithms, PLTMG approximates the error in triangle t using the formula

$$\|\nabla(u - u_h)\|_t^2 \equiv \int_t |\nabla(u - u_h)|^2 \, dx \approx v^t B v, \tag{1.1}$$

where (see Figure 1)

$$\nu_i = \begin{pmatrix} x_i \\ y_i \end{pmatrix} \quad \text{for } 1 \le i \le 3, \tag{1.2}$$

$$\ell_i = \nu_j - \nu_k \quad \text{for } (i,j,k) \text{ a cyclic permutation of } (1,2,3), \tag{1.3}$$

$$v = \begin{pmatrix} \ell_1^t M_t \ell_1 \\ \ell_2^t M_t \ell_2 \\ \ell_3^t M_t \ell_3 \end{pmatrix}, \qquad M_t = -\frac{1}{2}\begin{pmatrix} u_{xx} & u_{xy} \\ u_{xy} & u_{yy} \end{pmatrix}, \tag{1.4}$$

$$B = \frac{1}{48|t|}\begin{pmatrix} \ell_1^t\ell_1 + \ell_2^t\ell_2 + \ell_3^t\ell_3 & 2\ell_1^t\ell_2 & 2\ell_1^t\ell_3 \\ 2\ell_2^t\ell_1 & \ell_1^t\ell_1 + \ell_2^t\ell_2 + \ell_3^t\ell_3 & 2\ell_2^t\ell_3 \\ 2\ell_3^t\ell_1 & 2\ell_3^t\ell_2 & \ell_1^t\ell_1 + \ell_2^t\ell_2 + \ell_3^t\ell_3 \end{pmatrix}. \tag{1.5}$$

Equations (1.1)–(1.5) are derived by comparing the approximation error for linear and quadratic interpolation on t. See [2, 4] for details. The second derivatives in the 2×2 matrix M_t are taken as constants on t. The values of the second derivatives are extracted as a postprocessing step from the a posteriori error estimates. The remaining information needed to compute the right-hand side of (1.1) is generated directly from the geometry of element t.

To be effective, the approximation of the derivatives need not be extremely accurate. Many adaptive algorithms, and in particular those in PLTMG, are directed toward creating meshes in which the errors in all elements are equilibrated. Typically, adaptive algorithms develop refined meshes starting from rather coarse meshes. For reasons of efficiency, often many elements are refined between recomputing the approximate solution u_h.

MC also discretizes the solution over piecewise linear triangular or tetrahedral elements, and as in PLTMG, error estimates for each element are produced by solving a local problem using the edge-based quadratic bump functions. However, while these local problems involve inverting 3×3 matrices for scalar problems in 2D, the local problems are substantially more costly in 3D. In particular, for 3D elasticity, the local problems require the inversion of 18×18 matrices (6 bump functions and 3 unknowns per spatial point). Therefore, MC also provides an alternative less-expensive error estimator, namely, the residual of the strong form of the equation, following, e.g., [32]. In this paper, the numerical results involving MC are produced using the residual-based estimator.

While there is considerable theoretical support for the a posteriori error bounds which form the foundation for adaptive algorithms (see, e.g., the book of Verfürth [32] and its references), the adaptive algorithms themselves are largely heuristic, in particular, those aspects described above. However, there is a large and growing body of empirical evidence that such algorithms are indeed robust and effective for a wide class of problems. In particular, they are effective on coarse meshes, and on highly nonuniform meshes. The types of meshes likely to be generated in our parallel algorithm are qualitatively not very different from typical meshes where a posteriori error estimates are known to perform quite well.

In our procedure, we artificially set the errors to be very small in regions outside the subregion assigned to a given processor, so the standard refinement procedure is "tricked" into equilibrating the error on just one subregion. Since the target size of all problems solved in step 2 is the same, and each subregion initially has approximately equal error, we expect the final composite mesh to have approximately equal errors, and approximately equal numbers of elements, in each of the refined subregions created in step 2. That is, the composite mesh created in step 3 should have roughly equilibrated errors in all of its elements. This last statement is really just

an expectation, since we control only the target number of elements added in each subregion and do not control the level of error directly. This and other assumptions forming the foundation of our load balancing algorithms are discussed in more detail in section 3.2.

We note the standard adaptive procedures in PLTMG and MC have additional refinement criteria to insure conforming and shape regular meshes. Thus, some elements outside the given subregion but near its interface are typically refined in order to enforce shape regularity; the result is a smooth transition from the small elements of the refined region to larger elements in the remainder of the domain. If the target number of elements is large in comparison with the number of elements on the coarse mesh, this should be a relatively small effect.

To summarize, we expect that for any given problem, our algorithm should perform comparably to the standard algorithm applied to the same initial mesh in terms of the quality of the adaptive local mesh refinement.

2. Examples. In this section, we present some simple examples of the algorithm presented in section 1.

2.1. A convection-diffusion equation. In this example, we use PLTMG to solve the convection-diffusion equation

$$\begin{aligned} -\nabla \cdot (\nabla u + \beta u) &= 1 \qquad \text{in } \Omega, \\ u &= 0 \qquad \text{on } \partial\Omega, \end{aligned} \tag{2.1}$$

where $\beta = (0, 10^5)^t$, and Ω is the region depicted in Figure 2. This coarse triangulation has 2148 elements and 1368 vertices.

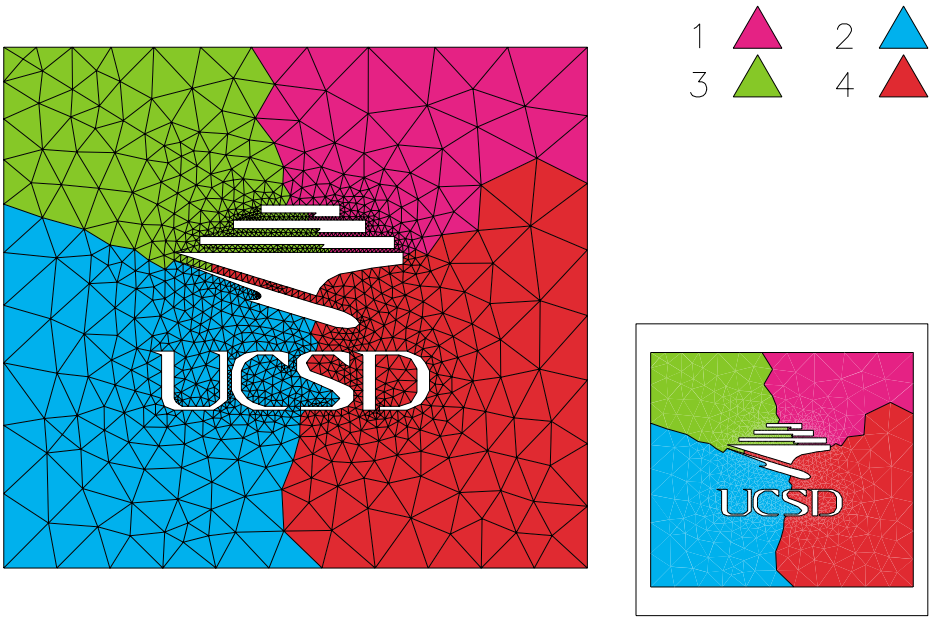
We partitioned the domain into four subregions with approximately equal error using the recursive spectral bisection algorithm, described in more detail in section 3. Then four independent problems were solved, each starting from the coarse grid and coarse grid solution. In each case the mesh is adaptively refined until a mesh with approximately 6000 unknowns (located at triangle vertices) is obtained. The mesh for one subdomain and the corresponding solution is shown in Figure 3. Notice that the refinement is largely confined to the given region, but some refinement in adjacent regions is needed in order to maintain shape regularity. We emphasize that these four problems are solved independently, by the standard sequential adaptive solver PLTMG. The only change to the code used for problem k $(1 \le k \le 4)$ was to multiply a posteriori error estimates for elements in regions $j \ne k$ by 10^{-6}, causing the adaptive refinement procedure to rarely choose these elements for refinement, except to maintain shape regularity.

The meshes from these four subproblems are combined to form a globally refined mesh with 37575 triangles and 19828 vertices. This mesh is shown in Figure 4.

The solutions from the four problems are combined to form a global solution that serves as initial guess for a global smoothing process. The final solution, as well as the a posteriori error estimates for this solution, are shown in Figure 5.

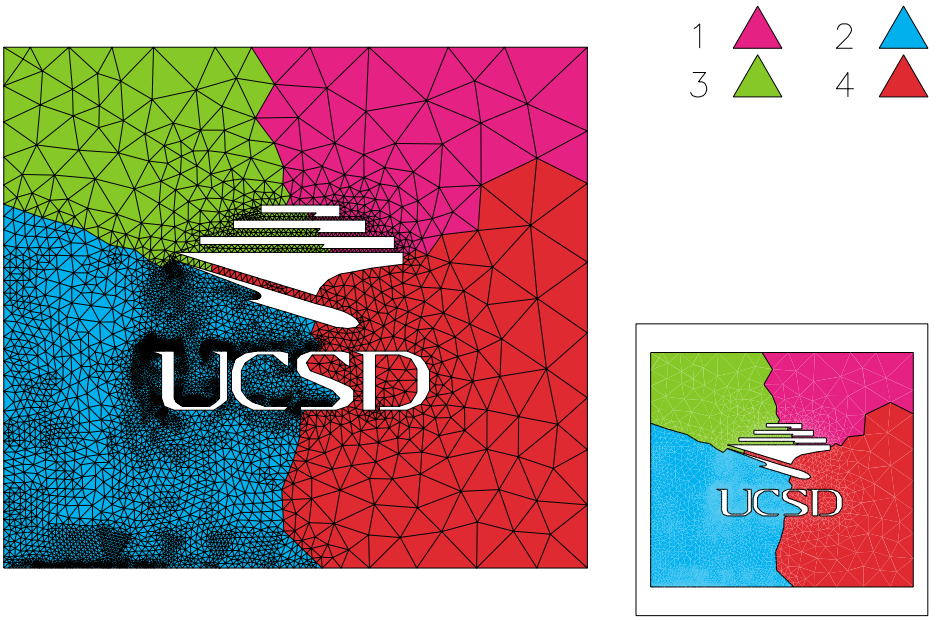
2.2. A full potential flow problem. For our second example, we use PLTMG to solve the full potential flow equation

$$\begin{aligned} -\nabla \cdot \rho(\nabla u)\nabla u &= 0 \qquad \text{in } \Omega, \\ \rho \partial u/\partial n &= g \qquad \text{on } \partial\Omega_1, \\ \rho \partial u/\partial n &= 0 \qquad \text{on } \partial\Omega_2, \end{aligned} \tag{2.2}$$



The initial triangulation, partitioned into four subregions with approximately equal error.

FIG. 2. *The coarse grid solution.*



The refined mesh for problem 2.

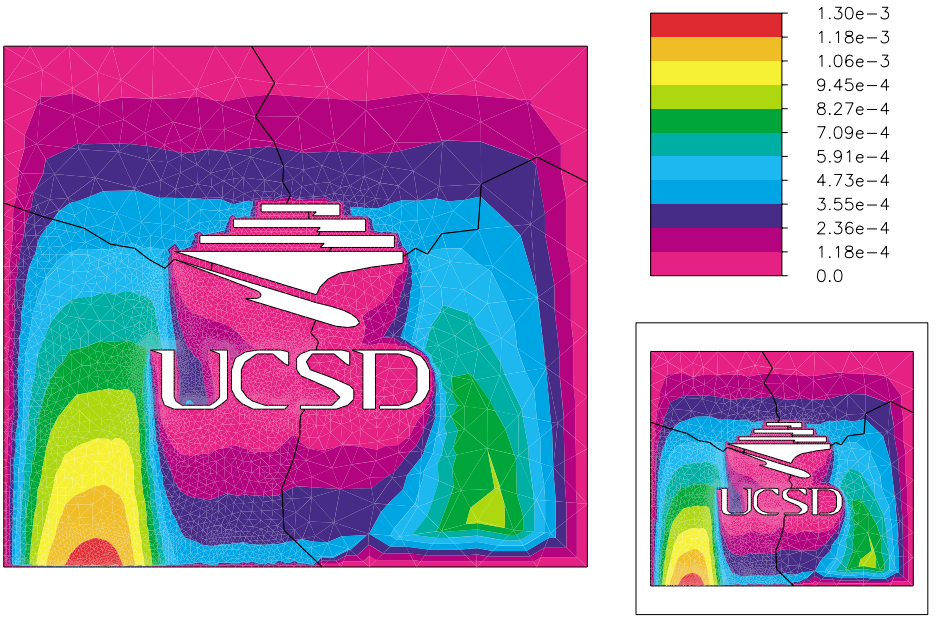
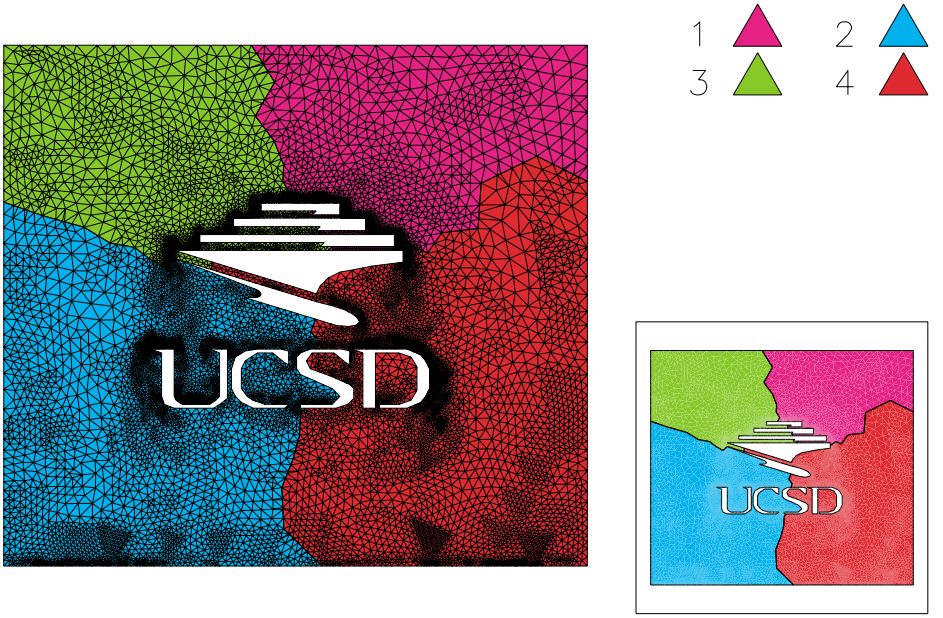


FIG. 3. *The solution for problem* 2.



The global refined mesh.

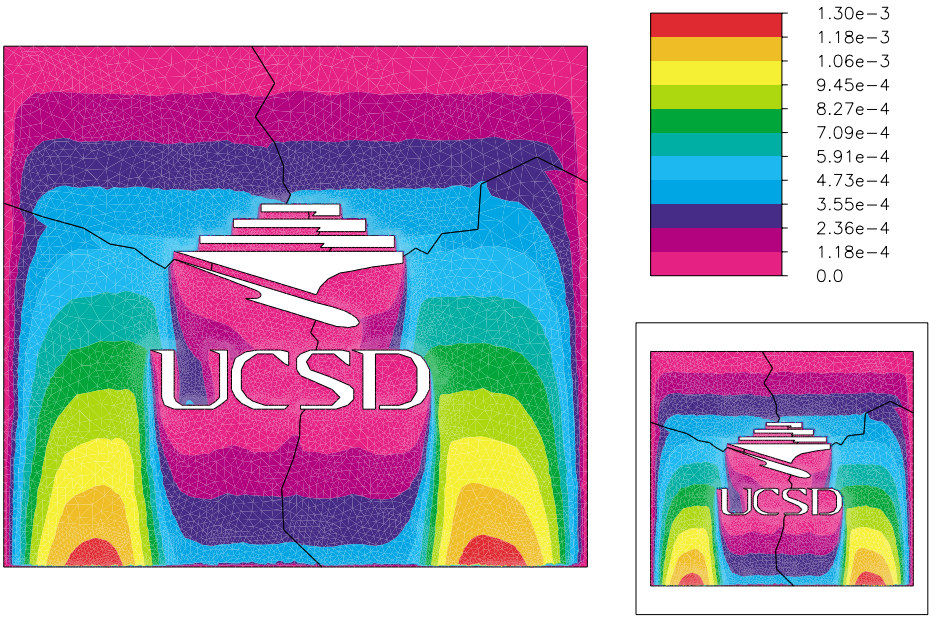


FIG. 4. *The initial guess for the global solution.*

The final global solution.

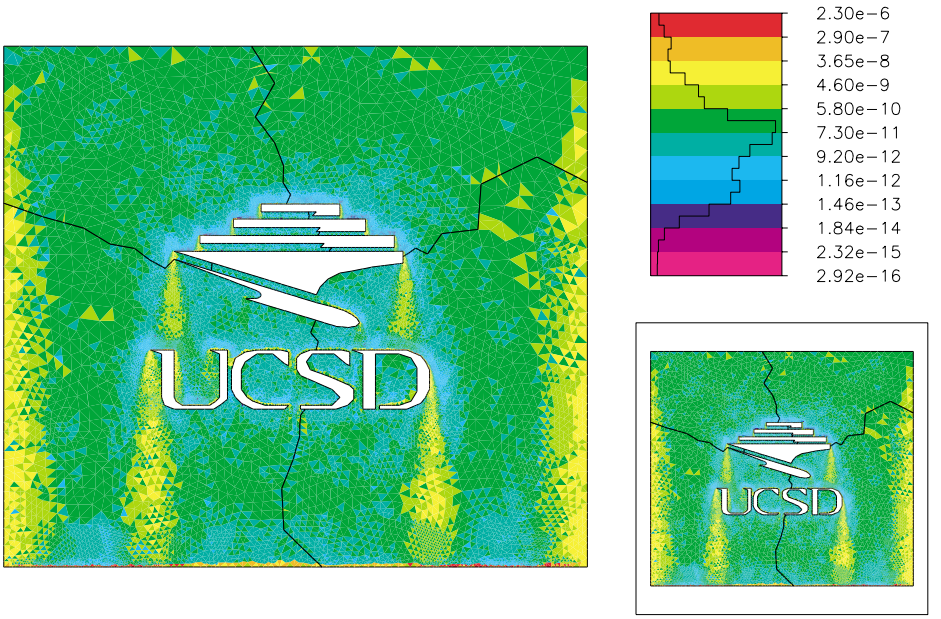


FIG. 5. *A posteriori error estimate for the final solution.*

where

$$\rho(\nabla u) = (1 - u_x^2 - u_y^2)^{\frac{1}{\gamma - 1}}$$

and $\gamma = 1.4$. $\partial\Omega_1$ is the outer boundary, where the solution assumes its asymptotic behavior, while $\partial\Omega_2$ is the inner boundary, in this example the profile of an Naca0012 airfoil. The initial mesh, generated from the geometry description, had 384 triangles and 221 vertices. In Figure 6 we illustrate the initial mesh and a detail of the local Mach number $M(\nabla u)$ given by

$$M(\nabla u) = \sqrt{\frac{2c}{\gamma - 1}},$$
$$c = \frac{1}{1 - u_x^2 - u_y^2} - 1.$$

$M(\nabla u)$ is one of the main quantities of interest in this calculation, and since it depends on ∇u, it represents a more severe test of our algorithm.

As in the first example, we partitioned the domain into four subregions with approximately equal error using the recursive spectral bisection algorithm. Then four independent problems were solved with the mesh adaptively refined to one with approximately 5000 unknowns. The mesh for one subdomain and the corresponding $M(\nabla u)$ are shown in Figure 7.

The meshes from the four subproblems are combined to form a globally refined mesh with 37913 triangles and 18723 vertices. This mesh is shown in Figure 8, along with $M(\nabla u)$ computed from the initial guess for the global solution. $M(\nabla u)$ computed from the global solution, as well as the a posteriori error estimates for the final global solution, are shown in Figures 9 and 10, respectively.

2.3. A 3D elasticity problem. The two previous examples demonstrated the effectiveness of the parallel algorithm for linear and nonlinear scalar problems in 2D. To illustrate that the parallel algorithm works equally well for coupled elliptic systems and for 3D problems, we will use MC to solve the elasticity equations for our third example:

$$-\nabla \cdot T(\nabla u) = f \quad \text{in} \Omega \subset \mathrm{R}^3, \tag{2.3}$$
$$n \cdot T(\nabla u) = g \quad \text{on } \Gamma_1, \tag{2.4}$$
$$u = 0 \quad \text{on } \Gamma_0, \qquad \partial\Omega = \Gamma_0 \cup \Gamma_1, \qquad \emptyset = \Gamma_0 \cap \Gamma_1. \tag{2.5}$$

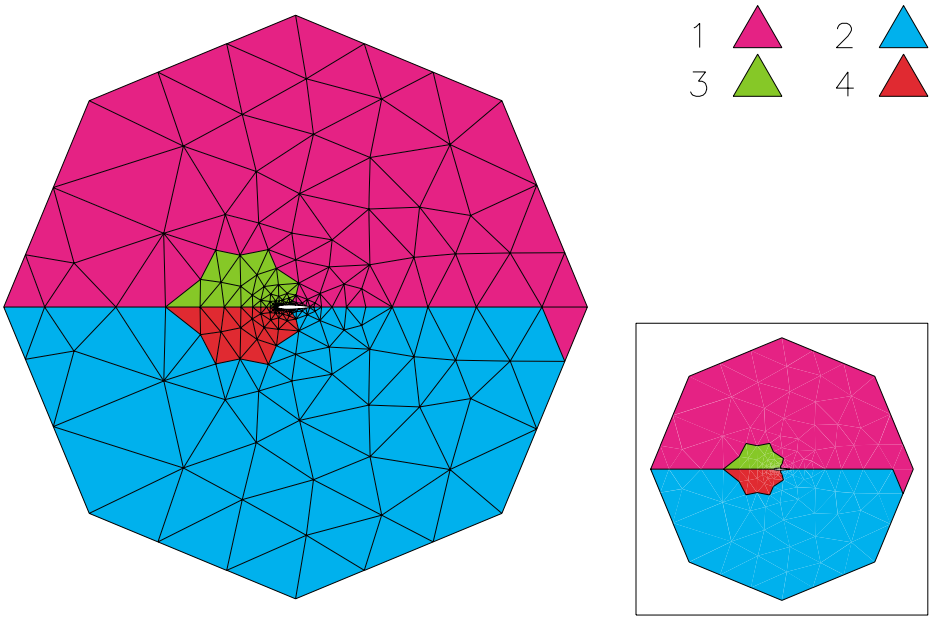
The stress tensor T is a function of the gradient ∇u of the unknown displacement u, and the corresponding deformation mapping φ and deformation gradient $\nabla\varphi$ are given by

$$\varphi = id + u : \overline{\Omega} \mapsto \mathrm{R}^3, \qquad \nabla u : \overline{\Omega} \mapsto \mathrm{M}^3, \qquad \nabla\varphi = I + \nabla u : \overline{\Omega} \mapsto \mathrm{M}^3.$$

We will use a linearized strain tensor and a linear stress-strain relation:

$$E(\nabla u) = \frac{1}{2}(\nabla u + \nabla u^T) : \overline{\Omega} \mapsto \mathrm{S}^3, \qquad T(\nabla u) = \lambda(\mathrm{tr}E)I + 2\mu E,$$

where the Lamé constants λ and μ are taken to be those of steel ($\lambda \approx 10.4403$, $\mu \approx 8.20312$). The solid object forming the domain is depicted in Figure 11. We apply traction forces to the back and top of each letter in a horizontal and slightly



The initial triangulation, partitioned into four subregions with approximately equal error.

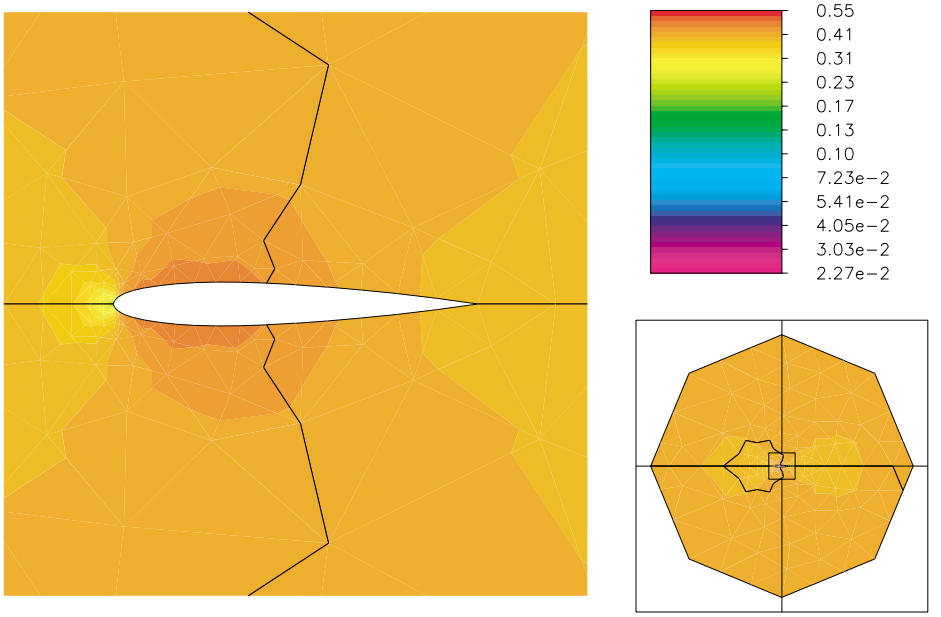
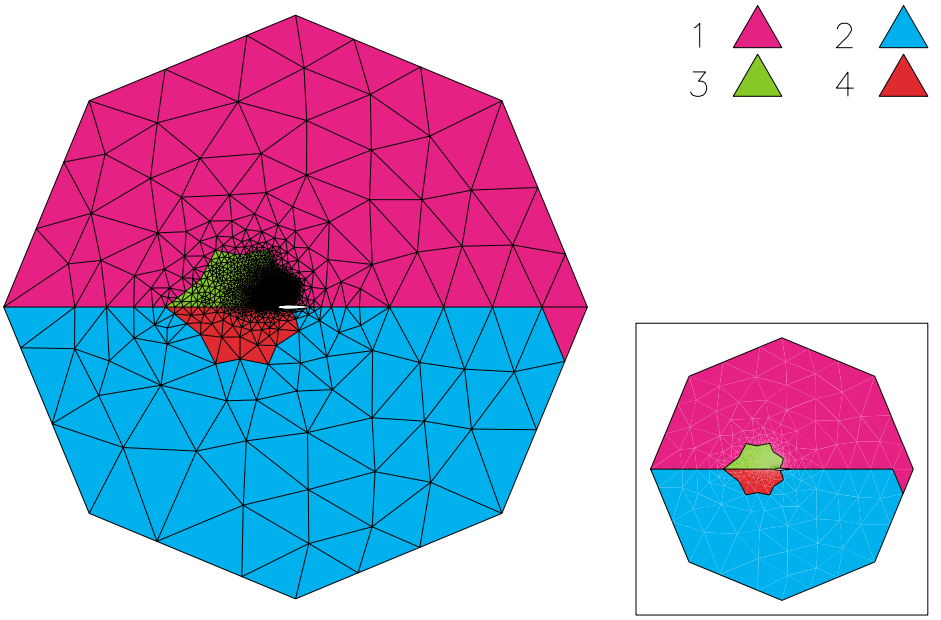


FIG. 6. $M(\nabla u)$ *computed from the coarse grid solution.*



The refined mesh for problem 3.

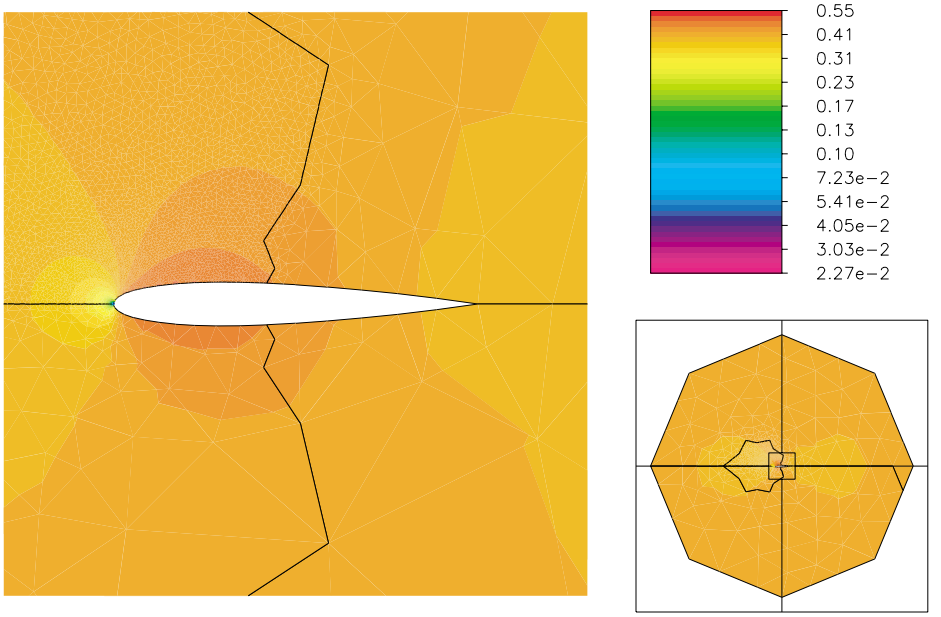
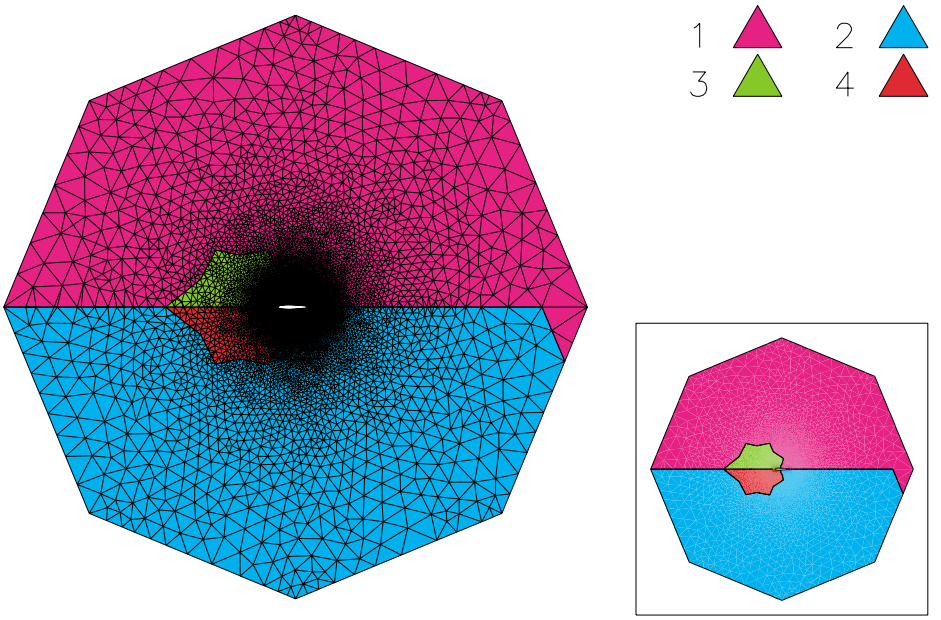


FIG. 7. $M(\nabla u)$ *computed from the solution for problem* 3.



The global refined mesh.

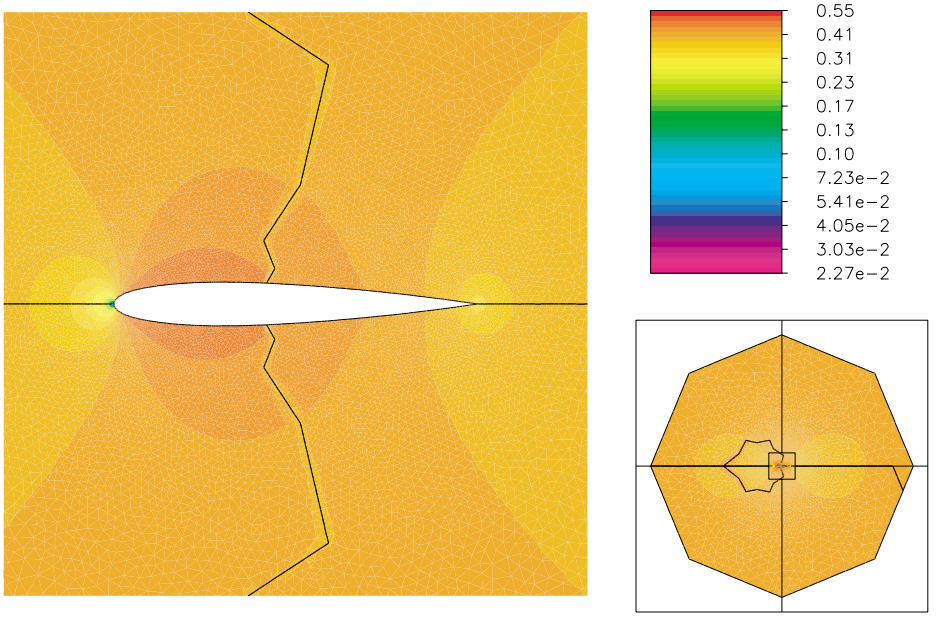


FIG. 8. $M(\nabla u)$ *computed from the initial guess for the global solution.*

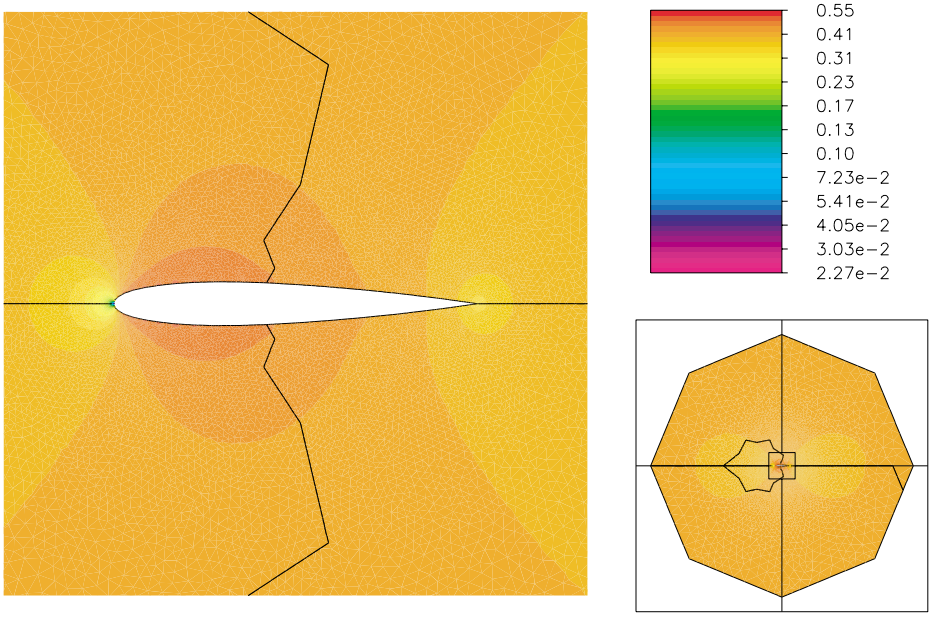


FIG. 9. $M(\nabla u)$ *computed from the final global solution.*

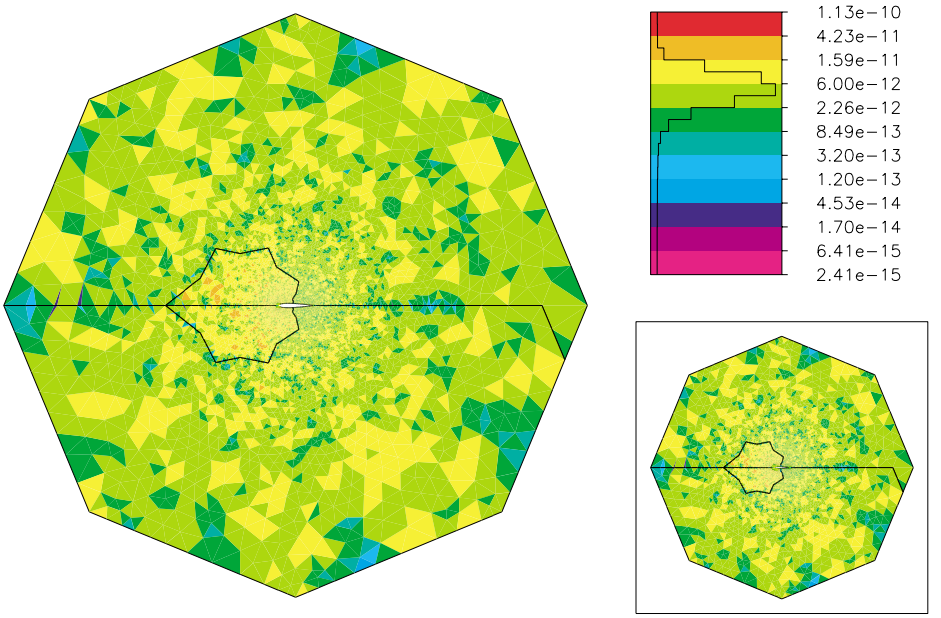


FIG. 10. *a posteriori error estimate for the final solution.*

upward direction, through the function g above. The base of each letter is fixed to a solid foundation, through the essential boundary condition above. No volume forces are present so that the function f above is zero. We then solve the resulting equations (2.3)–(2.5) adaptively using MC.

The initial coarse mesh in the left picture of Figure 11 has 296 tetrahedral elements and 159 vertices. As in the previous examples, we partition the domain into four subdomains with approximately equal error using the recursive spectral bisection algorithm described in section 3 below. The four subdomain problems are then solved independently by MC, starting from the complete coarse mesh and coarse mesh solution. In this example, the mesh is adaptively refined in each subdomain until a mesh with about 5000 vertices is obtained.

The resulting refined subdomain mesh for one subdomain and the corresponding solution (the deformation mapping φ) is shown in Figure 12. (The deformation is taken to be larger than the range of validity of the linear elasticity equations for visualization purposes.) As in the PLTMG examples, the refinement performed by MC is confined primarily to the given region, with some refinement into adjacent regions due to the closure algorithm which maintains conformity and shape regularity. The four problems are solved completely independently by the sequential adaptive code MC. This decomposition approach is especially effective for this particular problem due to the structure of the object, which leads to very weak couplings between the deformations of the individual letters, and due to the fact that the spectral bisection algorithm happens to decomposes the mesh close to the boundaries of the letters.

2.4. A 3D nonlinear elliptic system arising in gravitation. The fourth and final example involves the use of MC to solve a coupled nonlinear elliptic system in R^3, namely, the elliptic constraints in the Einstein equations (cf. [38]):

$$\hat{\gamma}^{ab}\hat{D}_a\hat{D}_b\phi = \frac{1}{8}\hat{R}\phi - \frac{1}{8}\phi^{-7}(\hat{A}^*_{ab} + (\hat{l}W)_{ab})^2 + \frac{1}{12}(\mathrm{tr}K)^2\phi^5 - 2\pi\hat{\rho}\phi^{-3}, \tag{2.6}$$

$$\hat{D}_b(\hat{l}W)^{ab} = \frac{2}{3}\phi^6\hat{D}^a\mathrm{tr}K + 8\pi\hat{j}^a. \tag{2.7}$$

The unknowns are the "conformal factor" ϕ and the vector potential W^b. The $(\hat{l}W)^{ab}$ operator above is a certain symmetrized gradient operator for tensors:

$$(\hat{l}W)^{ab} = \hat{D}^aW^b + \hat{D}^bW^a - \frac{2}{3}\hat{\gamma}^{ab}\hat{D}_cW^c. \tag{2.8}$$

The Einstein summation convention is used above, so that all repeated symbols in products imply a sum over that index. The gradient operator $\hat{D}^a$ is covariant, meaning that its application requires the use of Christoffel symbols due to the curvilinear nature of the coordinate system required to represent the underlying domain manifold. The Christoffel symbols are formed with respect to an underlying background metric $\hat{\gamma}^{ab}$, so that the left-hand side of the first equation for the conformal factor ϕ is essentially a covariant Laplace operator.

To use MC to calculate the initial bending of space and time around two massive black holes separated by a fixed distance by solving the above constraint equations, we place two spherical objects in space, the first object having unit radius (after appropriate normalization), the second object having radius 2, separated by a distance of 20. Infinite space is truncated with an enclosing sphere of radius 100. (This outer boundary may be moved further from the objects to improve the accuracy of boundary condition approximations.) Physically reasonable functions for remaining parameters

appearing in the equations are used to completely specify the problem (cf. [16] for details).

We then generate an initial (coarse) mesh of tetrahedra inside the enclosing sphere, exterior to the two spherical objects within the enclosing sphere. The mesh is generated by adaptively bisecting an initial mesh consisting of an icosahedron volume filled with tetrahedra. The bisection procedure simply bisects any tetrahedron which touches the surface of one of the small spherical objects. When a reasonable approximation to the surface of the spheres is obtained, the tetrahedra completely inside the small spherical objects are removed, and the points forming the surfaces of the small spherical objects are projected to the spherical surfaces exactly. This projection involves solving a linear elasticity problem (nearly identical to the problem solved in Example 3 above), together with the use of a shape-optimization-based smoothing procedure. The smoothing procedure locally optimizes the following shape measure function for a given d-simplex s, in an iterative fashion, similar to the approach taken in [4]:

$$\eta(s,d) = \frac{2^{2(1-\frac{1}{d})} 3^{\frac{d-1}{2}} |s|^{\frac{2}{d}}}{\sum_{0 \le i < j \le d} |e_{ij}|^2}.$$

The quantity $|s|$ represents the (possibly negative) volume of the d-simplex s, and $|e_{ij}|$ represents the length of the edge that connects vertex i to vertex j in the simplex. For $d = 2$, this is the shape-measure used in [4], with a slightly different normalization. For $d = 3$, this is the shape-measure given in [18], again with a slightly different normalization. Unlike Laplace smoothing, this local shape optimization approach is guaranteed to improve the shape of elements locally at each step, and always maintains a mesh of simplices with positive volumes.

The initial coarse mesh in Figures 13 and 14, generated using the procedure described above, has 31786 tetrahedral elements and 5809 vertices. As in the previous examples, we partition the domain into four subdomains (shown in Figures 15 and 16 with approximately equal error using the recursive spectral bisection algorithm described in section 3 below. The four subdomain problems are then solved independently by MC, starting from the complete coarse mesh and coarse mesh solution. The mesh is adaptively refined in each subdomain until a mesh with roughly 50000 vertices is obtained (yielding subdomains with about 250000 simplices each).

The resulting refined subdomain meshes are shown in Figures 17 and 18. As in the previous examples, the refinement performed by MC is confined primarily to the given region, with some refinement into adjacent regions due to the closure algorithm which maintains conformity and shape regularity. The four problems are solved completely independently by the sequential adaptive code MC. One component of the solution (the conformal factor ϕ) of the elliptic system is depicted in Figure 19 (the subdomain zero solution) and in Figure 20 (the subdomain two solution).

3. Computational considerations.

3.1. A spectral bisection algorithm. In this section we describe the algorithm we use for partitioning the coarse mesh so that each subregion has approximately equal error. This algorithm is a variant of the recursive spectral bisection algorithm [11, 27, 31]. While this particular mesh partitioning algorithm is one of the more expensive of the choices that we could make, we emphasize that it is used in our algorithm *only once*, and it is only used on a very small coarse grid problem. As

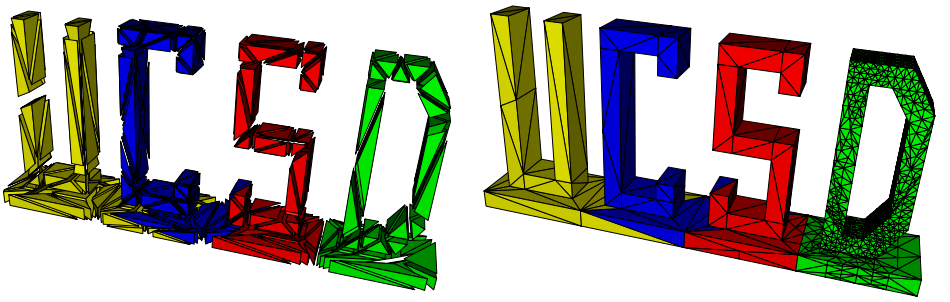


FIG. 11. *Initial spectrally bisected UCSD domain and the D subdomain mesh.*

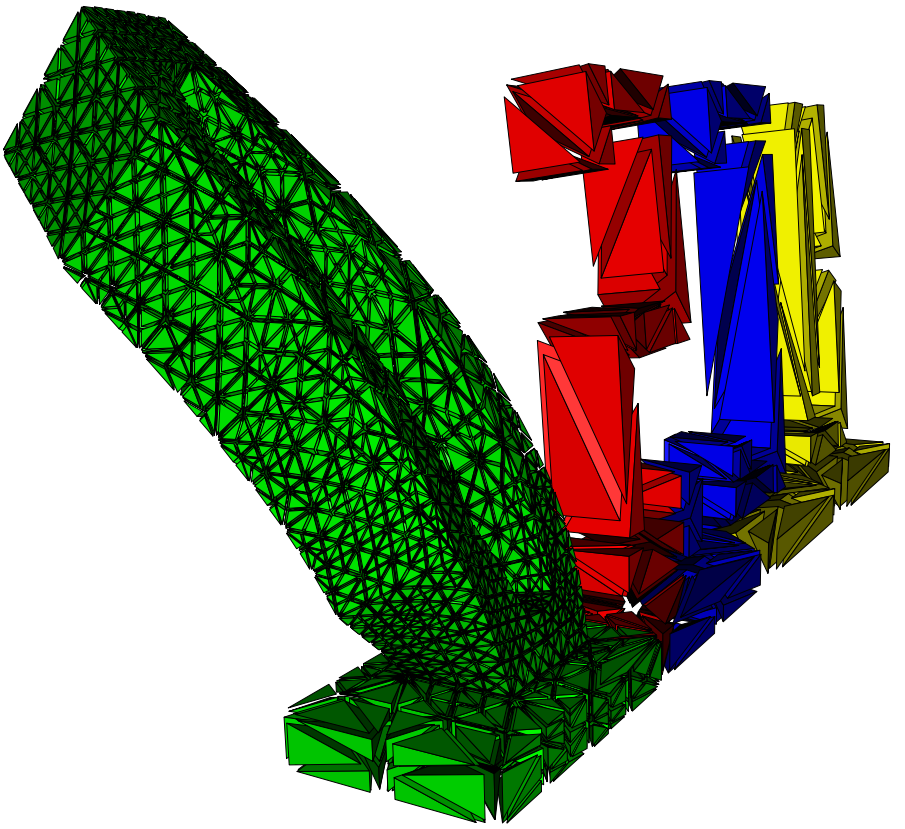


FIG. 12. *The exploded deformation of the D subdomain mesh (*4838 *vertices and* 20905 *simplices).*

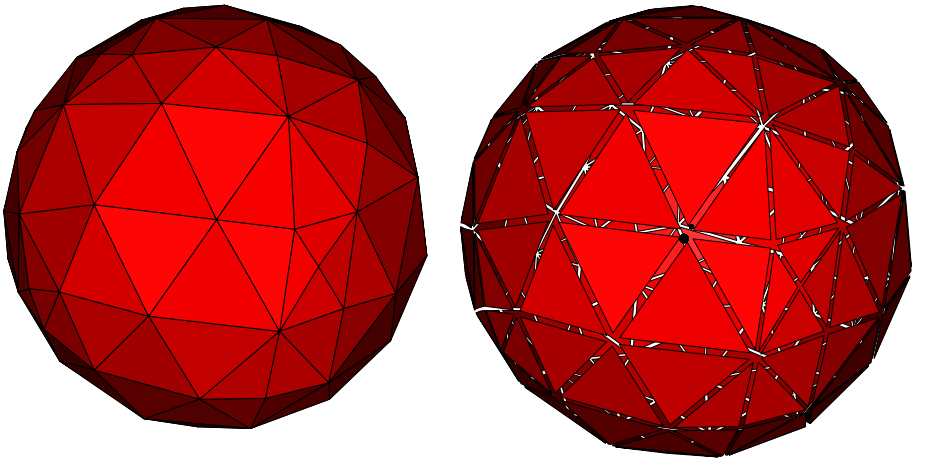


FIG. 13. *The coarse binary black hole mesh (*5809 *vertices and* 31786 *simplices).*

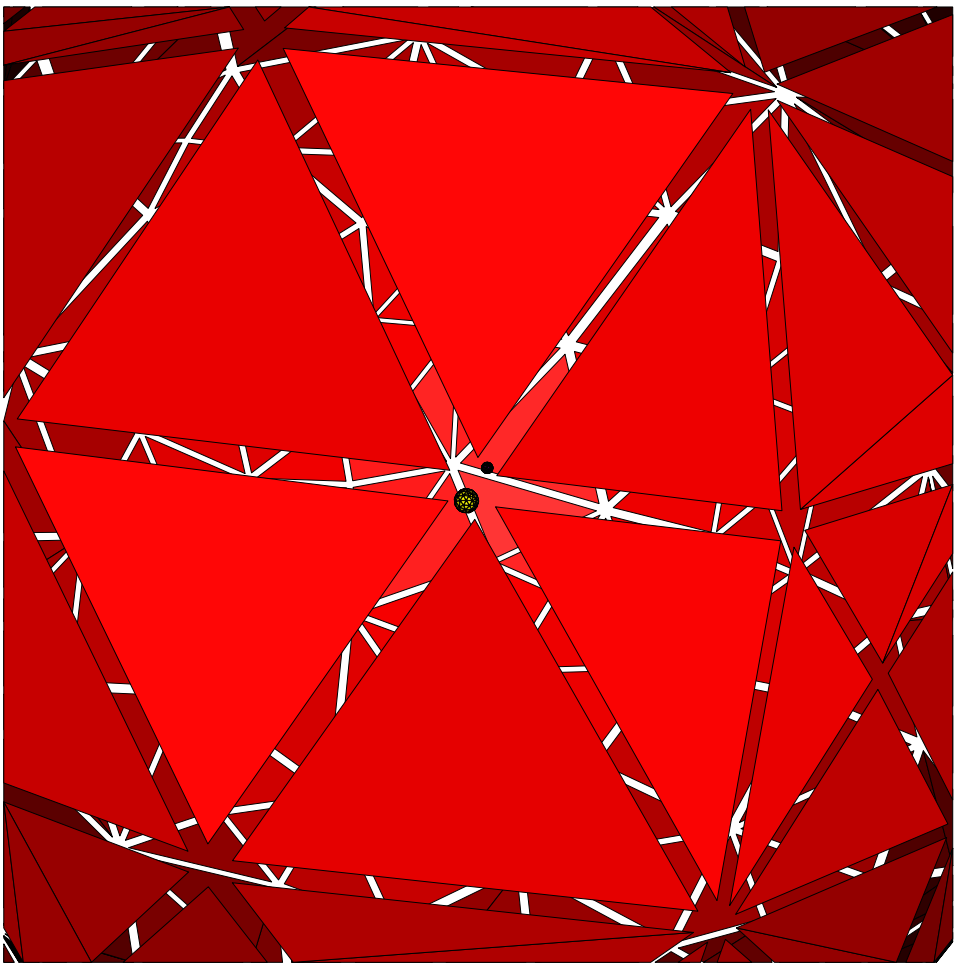


FIG. 14. *Exploded view of the coarse binary black hole mesh showing the two interior hole boundaries.*

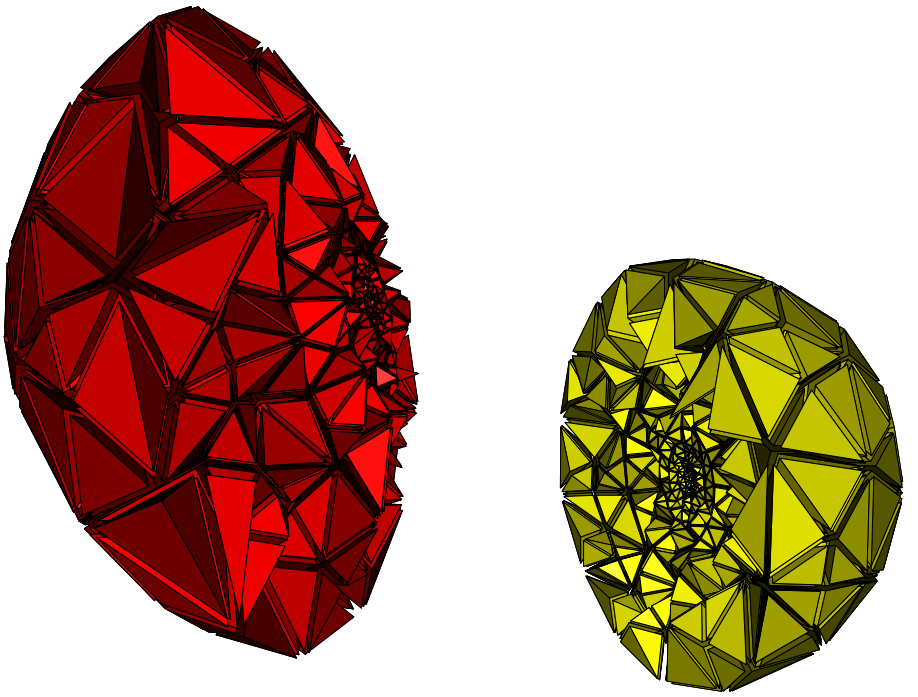


FIG. 15. *Subdomains* 2 *(red) and* 4 *(yellow) from spectral bisection of the coarse binary black hole mesh; these subdomains enclose two smaller subdomains that contain the inner holes.*

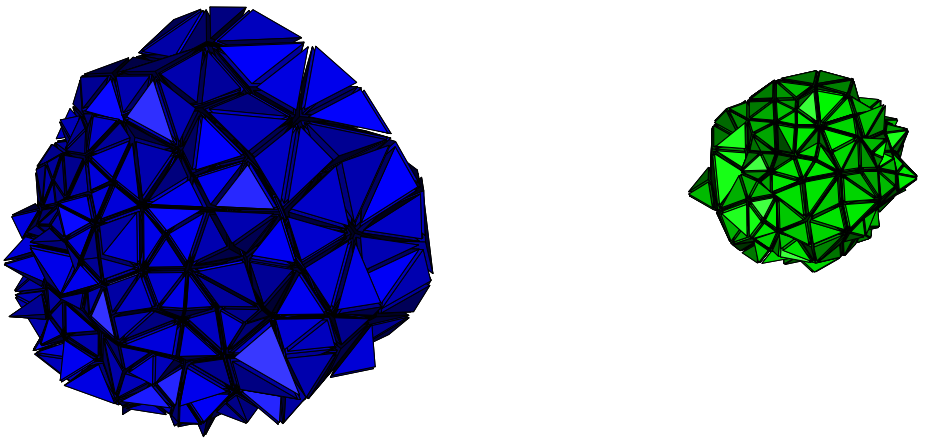


FIG. 16. *Subdomains* 3 *(blue) and* 1 *(green) from spectral bisection of the coarse binary black hole mesh; these subdomains each contain one of the inner holes.*

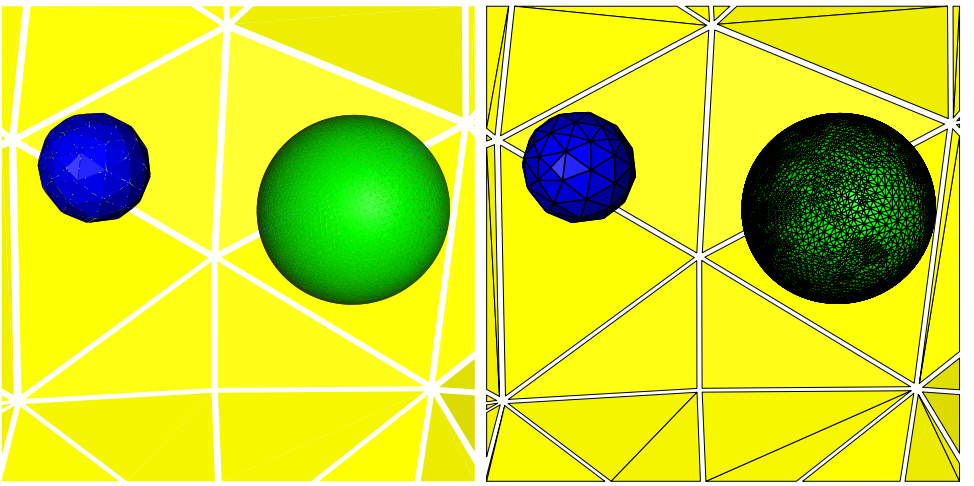


FIG. 17. *Refined mesh for subdomain* 1 *(*51915 *vertices and* 266114 *simplices; only faces of tetrahedra on the boundary surfaces are shown).*

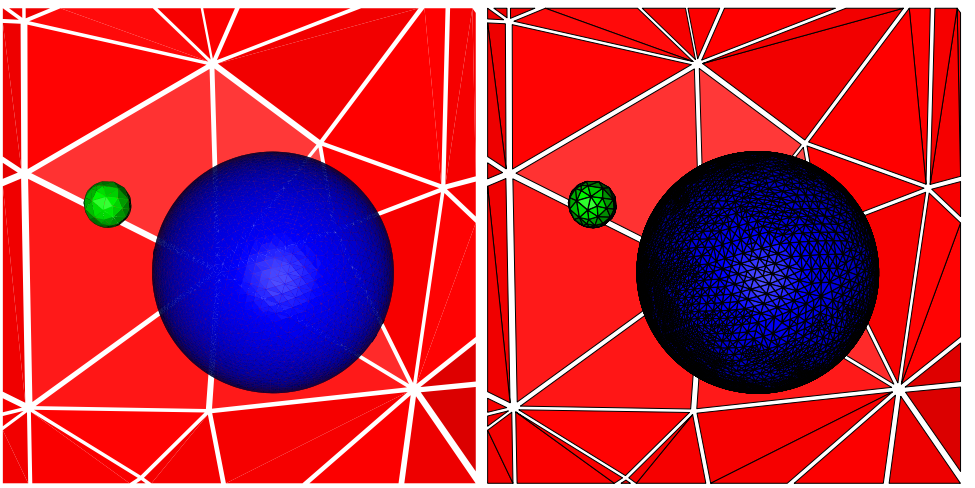


FIG. 18. *Refined mesh for subdomain* 3 *(*44550 *vertices and* 228194 *simplices; only faces of tetrahedra on the boundary surfaces are shown).*

a result, the initial partitioning cost is much smaller than the solve time for a single subdomain problem (see Tables 1 and 2 below).

Let $\mathcal{T}$ denote a triangulation of the domain Ω with triangular elements $t_i \in \mathcal{T}$, $1 \le i \le N$, and let e_i denote the a posteriori error estimate for t_i,

$$e_i \approx \|\nabla(u - u_h)\|^2_{t_i}.$$

Define the $N \times N$ symmetric, positive semidefinite M-matrix A by

$$A_{ij} = \begin{cases} -1, & i \neq j \text{ and } t_i \text{ and } t_j \text{ share a common edge,} \\ 0, & i \neq j \text{ and } t_i \text{ and } t_j \text{ do not share a common edge,} \\ s_i, & i = j,\ s_i = -\sum_{k \neq i} A_{ik}, \end{cases}$$

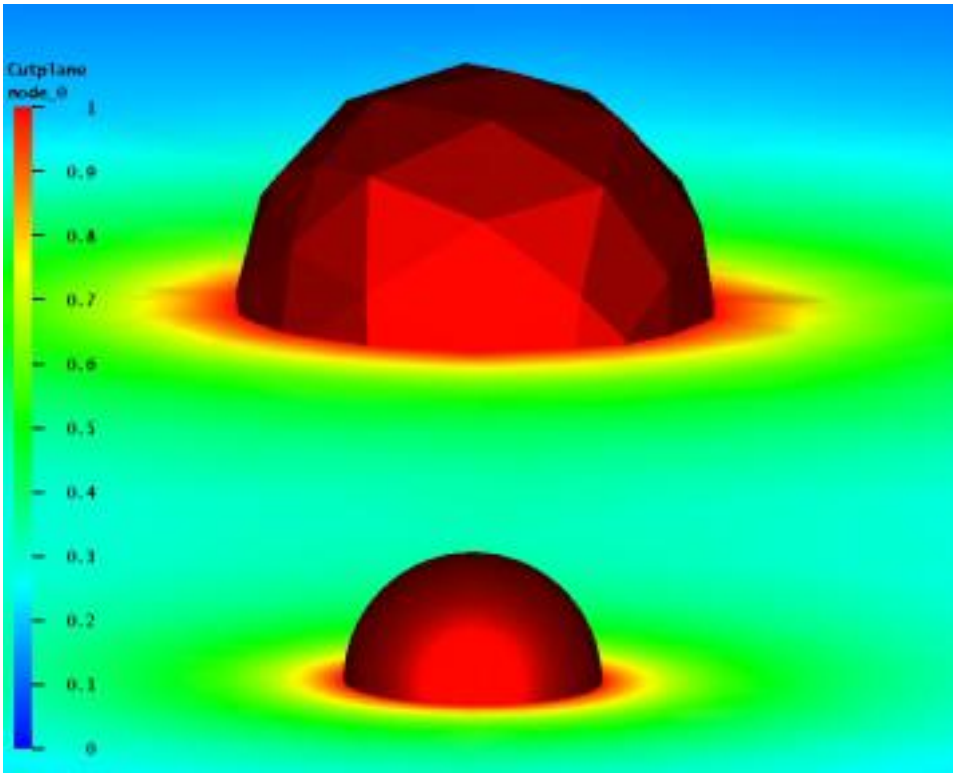


FIG. 19. *The conformal factor ϕ from the adapted subdomain* 1 *solve.*

and the positive diagonal matrix D by

$$D_{ii} = e_i / e_{max},$$

where

$$e_{max} = \max_i e_i.$$

A typical row of A will have three nonzero off-diagonal elements and $A_{ii} = 3$; this is the so-called *discrete Laplacian* for the dual graph for the triangulation. (The triangles themselves are the nodes of the dual graph, and the edges are defined by the adjacency relation.) We consider the generalized eigenvalue problem

$$A\psi = \lambda D\psi. \tag{3.1}$$

Our approach is standard; by construction, the smallest eigenvalue for (3.1) is $\lambda_1 = 0$ and $\psi_1 = (1, 1, \ldots, 1)^t$. Our interest is in the second eigenvector ψ_2, known as the Fiedler vector.

Let $\mathcal{S}^+$ and $\mathcal{S}^-$ denote the index sets corresponding to positive and nonpositive components of ψ_2. Then from the orthogonality relation

$$\psi_1^t D\psi_2 = 0$$

we have

$$\sum_{i\in\mathcal{S}^+} e_i\psi_{2,i} = -\sum_{i\in\mathcal{S}^-} e_i\psi_{2,i}.$$

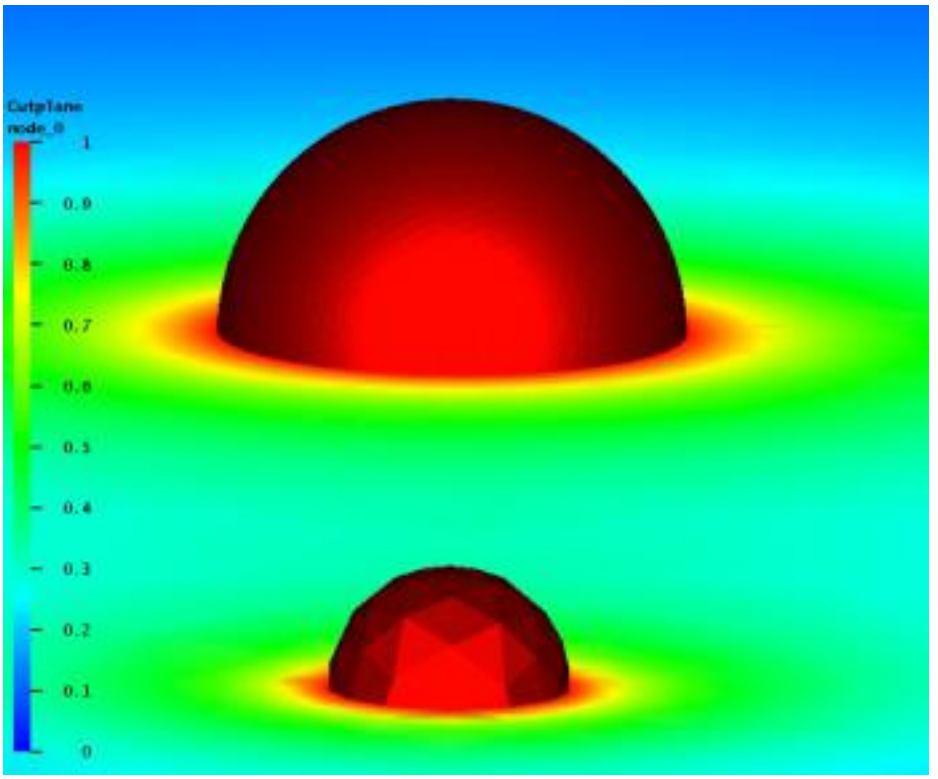


FIG. 20. *The conformal factor ϕ from the adapted subdomain* 3 *solve.*

This indicates that the total error for the two groups of triangles is approximately balanced (and would be exactly balanced if all entries of ψ_2 were ± 1). Using this observation as motivation, to construct a partition of the elements we first find a permutation of the elements $\{p_i\}$ such that

$$\psi_{2,i} \leq \psi_{2,j} \quad \text{if and only if} \quad p_i < p_j.$$

We then find the index k which provides the best partition of the form

$$\sum_{p_i \leq k} e_i \approx \sum_{p_i > k} e_i. \tag{3.2}$$

Often this partition is very close to the partition based on $\mathcal{S}^{\pm}$. This is similar to the strategy suggested by Chan, Ciarlet, and Szeto [11].

As usual, we apply this approach recursively, at each level dividing each group of elements into two smaller groups by solving an eigenvalue problem of the type (3.1) restricted to that group of elements. Thus after k steps, we have created a partition of the elements into 2^k groups of roughly equal error.

We now briefly describe some details of our procedure for computing the second eigenvector of (3.1). Our procedure is essentially just a classical Rayleigh quotient iteration [25], modified both to bias convergence to λ_2, and to account for the fact that the linear systems arising in the inverse iteration substep are solved (approximately) by an iterative process. To simplify notation and avoid multiple subscripts, we let $\phi_k \approx \psi_2$, where k now denotes the iteration index.

We suppose that we have a current iterate ϕ_k which satisfies

$$\phi_k^t D\phi_k = 1 \qquad \text{and} \qquad \psi_1^t D\phi_k = 0. \tag{3.3}$$

Using ϕ_k we compute the approximate eigenvalue $\sigma_k \approx \lambda_2$ from the Rayleigh quotient

$$\sigma_k = \phi_k^t A \phi_k$$

and form the residual vector

$$r_k = \sigma_k D\phi_k - A\phi_k.$$

Note that by construction $\psi_1^t r_k = \phi_k^t r_k = 0$. Next, we approximately solve the linear system

$$(A - \theta_k \sigma_k D)\tilde{\delta}_k = r_k, \tag{3.4}$$

where $0 \le \theta_k \le 1$ is chosen to try to insure $0 \le \theta_k \sigma_k \le \lambda_2$. A simple strategy such as $\theta_k = 1 - 2^{-k}$ is often effective (although, of course, offers no guarantees). From $\tilde{\delta}_k$, we form the vector δ_k satisfying

$$\delta_k^t D\delta_k = 1 \qquad \text{and} \qquad \psi_1^t D\delta_k = \phi_k^t D\delta_k = 0.$$

The inverse iteration step uses a conservative shift policy in order to strongly bias the calculation in favor of convergence to the desired second eigenvector. The residual appears as right-hand side, since this system is to be solved by iteration rather than by direct Gaussian elimination. In this circumstance only a few iterations are used, and the effect is mainly to attenuate unwanted eigenvectors rather than "blow up" the desired eigenvector. For our iterative method, we use a conjugate gradient procedure with symmetric Gauss–Seidel preconditioner. So far, this has proven to be simple and effective, but the issue of the most efficient solver in this context is presently open.

Finally, we solve the 2×2 eigenvalue problem

$$\hat{A}v = \lambda v,$$

where

$$\hat{A} = \begin{pmatrix} \phi_k^t \\ \delta_k^t \end{pmatrix} A \begin{pmatrix} \phi_k & \delta_k \end{pmatrix}.$$

If $v = (\alpha, \beta)^t$ is an eigenvector corresponding to the smaller eigenvalue, we form

$$\tilde{\phi}_{k+1} = \alpha\phi_k + \beta\delta_k,$$

and then ϕ_{k+1} is formed from $\tilde{\phi}_{k+1}$ by imposing conditions (3.3). The use of this subspace-iteration–like calculation rather than a simple eigenvector update provides a second means to bias the overall Rayleigh quotient iteration towards convergence to ψ_2. It also compensates to some extent for the loss in convergence rate due to the conservative shift policy and incomplete solution of (3.4) by iteration.

This completes the description of a single Rayleigh quotient iteration. Note that continually imposing orthogonality conditions with respect to ψ_1 is mathematically unnecessary, but is important in practice because this direction is reintroduced by roundoff error. Without systematically and continually excluding this eigenvector, the Rayleigh quotient iteration could easily converge to ψ_1.

3.2. Load balancing. Partitioning the domain to achieve approximately equal error in each subregion is not really the optimal approach. The optimal strategy is to partition the domain such that each subregion requires equal *work* for the ensuing calculation, and the errors are approximately equal in each element of the global composite mesh. Estimating the work is problematic for many reasons, among them:

- The cost of function evaluations and numerical integration used in computing matrices, right-hand sides, and a posteriori error estimates might vary significantly in various regions. Moreover, the number of such quadratures depends on the number of elements, and the number of instances when such assembly steps are required.
- The number of iterations of Newton's method for nonlinear systems and linear iterative methods for linear systems may vary slightly from problem to problem, even though all are derived from the same continuous problem. Because the number of iterations is usually small, the percentage change in the work can be quite large (e.g., 3 rather than 2 multigrid cycles represents a 50% increase in the work for that part of the computation). It should also be clear that such small differences are difficult to predict in advance of the actual calculation. The cost per iteration will also vary due to differing orders of the problems.
- The cost of grid management (refining, unrefining, moving the mesh points, and maintaining the relevant data structures) will vary with the number of elements involved and the particular mix of tasks.
- The number of major iterations through the adaptive feedback loop may differ from problem to problem, even if the final meshes all have about the same number of unknowns.

Creating subregions of approximately equal error for the initial partition really amounts to the fragile assumption that this corresponds to approximately equal work for each processor. Although one can hope that more sophisticated models of work will lead to improvement, it seems certain that the overall flexibility and complexity of current adaptive solvers will still make this aspect of the initial load balancing phase problematic.

On the positive side, despite all the dangers mentioned above, our load balancing procedure using a posteriori error estimates has empirically been observed to be much better than one might at first expect, at least for the classes of problems we address. For example, in Table 1, we give the overall execution times (in seconds) for the two PLTMG example calculations described in section 2 (examples 1 and 2 in sections 2.1 and 2.2, respectively). These times are for an SGI Octane 195mhz R10000, using the f77 compiler options -O -32.

In Table 2, we give the execution times (in seconds) for the two MC example calculations described in section 2 (examples 3 and 4 in sections 2.3 and 2.4, respectively). These times are for a single 195Mhz R10000 processor of an SGI Octane, using the IRIX C compiler with optimization -O2. While the numbers of vertices and elements in the subdomain meshes of the 3D UCSD example are similar to the two PLTMG examples, there are actually three unknowns at each vertex (the three displacement degrees of freedom), so the discrete problem sizes are triple the number of vertices.

The initialization times in both tables includes generating a coarse mesh from the geometry description, solving the coarse mesh problem, computing a posteriori error estimates, and partitioning the domain. The initialization was done on one processor.

TABLE 1
PLTMG execution times (seconds) for convection-diffusion example (UCSD) and the potential flow example (NACA).

	UCSD	NACA
Initialization	2.49	0.82
Solve subproblem 1	6.58	12.9
Solve subproblem 2	6.75	13.0
Solve subproblem 3	6.92	13.0
Solve subproblem 4	6.78	12.9
Postprocessing	0.89	0.81

TABLE 2
MC execution times (seconds) for the elasticity example (UCSD3D) and the binary black hole example (BHOLE).

	UCSD3D	BHOLE
Initialization	6	74
Solve subproblem 1	71	1905
Solve subproblem 2	95	1523
Solve subproblem 3	73	1811
Solve subproblem 4	82	1467
Global test solution	422	—

The times for each of the subproblems include all aspects of the adaptive feedback loop, including matrix and right-hand side assembly, solution of linear and nonlinear systems, a posteriori error estimation, and adaptive refinement and mesh smoothing. In Table 1, this also includes some relatively inexpensive clean-up performed by PLTMG, mainly removing unwanted parts of each mesh and solution in preparation for creating the global composite mesh. (These calculations are not performed by MC and do not appear in the timing figures in Table 2.)

The postprocessing costs given in Table 1 for the PLTMG implementation includes the creation of a global conforming mesh and forming an initial guess for the solution on that mesh using the procedures described in section 4.1. In our present code, this is also done on one processor. In contrast, the MC implementation does not form a global problem; this is justified to some extent in section 4.3. The global test solution costs listed in Table 2 for the first MC example reflects the cost of solving the entire problem sequentially on one processor without decomposition. (The refined black hole subdomain problems were so large that it was not possible to solve a single global problem on an SGI Octane with roughly four times the number of simplices of each subdomain problem.) What is interesting to note is that the overhead required to decompose the problem is amortized by the gain due to the reduced subproblem sizes; solving the subproblems sequentially is actually faster than solving the global problem. In other words, if the solution quality of the decomposition algorithm is reliable, then the decomposition algorithm actually reduces the sequential solution time when viewed purely as a sequential algorithm. Note that if the decomposition algorithm is used in conjunction with solvers with less favorable complexities than the $O(N \log N)$ complexities in PLTMG and MC, then the decomposition algorithm would show an even larger gain over solving the global problem.

From Tables 1 and 2 we see that although the mix of calculations was different for each subproblem, the overall times do not vary too much. And since these problems were solved completely independently, there was no time spent in communication between processors, synchronization, etc. Thus to some extent, the time potentially

saved by not having the processor communicate during the bulk of the computation offsets the time potentially lost by imperfect load balancing. The potential flow problem has a smaller initialization time than the convection-diffusion problem, mainly due to the smaller size of its coarse mesh problem (221 vertices compared to 1368). The solution times for the subspace problems are larger, due mainly to the facts that the potential flow problem is nonlinear and involves a Newton iteration, and it has more expensive coefficient function evaluations in the assembly phases and the a posteriori error estimates. In the 3D examples, the black hole calculation has a much larger initialization time than the elasticity problem due to the size of the initial coarse mesh (159 vertices compared with 5,809 vertices).

3.3. Scalability. We now consider some aspects of the scalability of our procedure. Let N_c denote in the size of the coarse grid problem (number of elements or grid points). Let N_f denote the target size of the global fine grid problem, p denote the number of processors, and N_p denote the target problem size for each of the processors.

We have, approximately,

$$N_p \approx N_c + \frac{N_f - N_c}{p}. \tag{3.5}$$

Relation (3.5) does not take into account the fact that the processor given the task of refining subregion Ω_i will refine some elements *not* in Ω_i. This will occur mainly near the interface boundaries of Ω_i, where the mesh will be graded in a smooth fashion from the smaller elements of Ω_i to larger elements that cover the remainder of Ω. Such grading is necessary to maintain a conforming, shape regular mesh. In a typical situation, one would expect this to be an effect of order $O(N_p^{1/2})$ in 2D, and of order $(N_p^{2/3})$ in 3D. Nevertheless, in practical situations, choosing $N_p > N_c + (N_f - N_c)/p$ is generally needed to achieve a fine grid problem size of N_f.

It seems clear that generally one should have

$$N_c \gg p. \tag{3.6}$$

A requirement like (3.6) is important to give the partitioning algorithm enough flexibility to construct regions of approximately equal error. For example, in the extreme case where the number of elements and the number of processors are equal, then the only partition is to provide one element to each processor, regardless of the error. This would likely result in a very uneven distribution of the error and poor performance of our adaptive refinement strategy.

It also is important to have

$$N_p \gg N_c. \tag{3.7}$$

This will marginalize the cost of redundant computations. For example, if $N_p = 2N_c$, then one could expect that about half of the computation on each processor would be redundant, which is a significant fraction of the total cost. By solving the problem on the entire domain, using a coarse mesh in all but one subregion, we are in effect substituting computation for communication. This trade-off will be most effective in situations where N_p is much larger than N_c (e.g., $N_p > 10N_c$) so that the redundant computation represents a small fraction of the total cost.

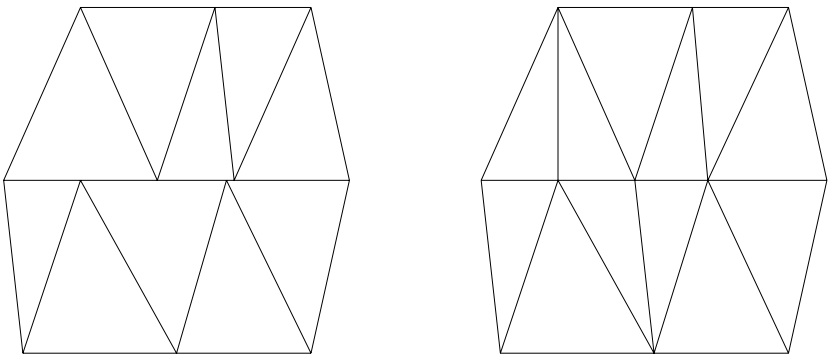


FIG. 21. *Nonmatching interface, shown on the left, is made conforming by refining elements and adjusting vertex locations. The resulting conforming interface is shown on the right.*

Taken together (3.5)–(3.7) indicate that for good performance, the difficulty of the problem must in some sense scale in proportion to the number of processors. That is, "simple" problems with only a few significant features that need to be resolved through refinement can most efficiently be solved using a few processors. Such problems could be handled on a small network of powerful workstations. Using our procedure effectively with, say, $p = 128$, would require a more difficult problem which could be decomposed into at least 128 regions, with most including some interesting behavior to resolve.

4. The global solution. In this section we discuss some options for combining the independent calculations to form a global composite mesh and solution.

4.1. Conforming meshes and global smoothing. In PLTMG, we have implemented an option where the independently generated meshes are glued together to form a global conforming mesh. We begin by simply deleting unwanted parts of the mesh from each of the independently solved problems. The union of the remaining submeshes forms a global mesh. Mesh points on the interfaces are restricted to move only along the interface during mesh smoothing phases of the independent solution process, so the overall geometry of the subregions remains conforming. However, along the interfaces the meshes will generally be nonmatching, leading to a global nonconforming triangulation, as shown in Figure 21. This mesh is made globally conforming through a combination of the refinement of triangles with boundary edges on the interface, and adjusting the locations of some interface vertices. The result is a matching grid along the interface and a globally conforming triangulation. These alternatives are illustrated in Figure 21. Once the mesh is globally conforming, local mesh improvement routines of PLTMG (e.g., edge swapping and mesh smoothing) can be employed to improve the shape regularity of the elements as needed.

We remark that the element sizes on both sides of an interface should be approximately equal if the load balancing algorithm is working properly, so that this gluing process, although technically complicated in terms of data structures and implementation, is both simple and natural at a more abstract level.

If one uses a refined element tree data structure for the refinement process [3, 5], as in previous versions of PLTMG, then this procedure is greatly simplified, since each independent problem starts from the same element tree. A simple postprocessing step

can enforce equal (and hence conforming) levels of refinement for elements sharing an interface edge. Also, if mesh smoothing is disallowed for vertices on the interface, the post processing is further simplified, since each interface vertex created during the independent solution process will be exactly the midpoint of the edge it refines. Assuming the refinement levels are approximately the same on both sides of an interface, it is likely that most vertices along an interface from one independent problem will have matching counterparts on the other side of the interface from another independent problem. Furthermore, the locations of nonmatching vertices cannot be arbitrary, but must lie, e.g., at midpoints of unrefined edges. Such additional constraints, if present in the refinement algorithm, greatly simplify the algorithm for rendering a conforming mesh.

In any event, once a conforming mesh is created, a solution on the globally refined mesh can be defined as follows. For vertices not on the interface, we use the solution from the independent problem corresponding to that region. For vertices on the interface, the solution is defined by a simple averaging of the solutions from the relevant independent problems. This procedure seems adequate if the resulting solution is to serve only as an initial guess for a subsequent global solution technique. However, one could conceive of more sophisticated and more accurate procedures. For example, one could assemble and solve a problem for the interface values, holding all other values fixed. This results in a low-dimensional system consisting mainly of tridiagonal blocks, with some interblock coupling due to so-called *cross points*. One could also include points adjacent or near to the interfaces as well, yielding larger, but still easily solved systems of equations.

In our present code, all postprocessing is done on one processor, following the solution phase. There are several ways one could make this parallel. For example, one could initially have several processors each glue together the results from two subproblems. These larger pieces then could be combined (in parallel) to make composite meshes arising from four subproblems. Continuing in this fashion leads to an overall algorithm with logarithmic complexity. However, we prefer a simple alternative strategy in which some processors solve subproblems while other processors simultaneously postprocess results from earlier computations.

An example is given in Table 3. Here we assume that the domain is partitioned into 16 subproblems ($P_1 - P_{16}$) to be distributed among four processors. Rather than provide four problems to each processor, we provide five problems to three of the processors and only one to the fourth. After solving its subproblem, this processor then begins the task of postprocessing the previously computed solutions. We remark that the cost of postprocessing a problem is usually less than one-third the cost of solving it. Indeed it was much smaller in both examples given in Table 1. This ratio was assumed for convenience in this illustration; in practice, one might assign Processor 0 several subproblems before switching to postprocessing calculations.

To compute the global solution, one can of course simply assemble and solve the global problem on a single processor. This was the choice made in PLTMG, but it was made mainly to maximize the use of the existing code, rather than for reasons of efficiency. More in keeping with the overall strategy of parallel processing, there are a variety of standard domain decomposition approaches that could be applied in this situation, both with and without overlap. The relevant decomposition and communication channels can be based on the same partitioning used to create the independent problems, which greatly simplifies the implementation. One could also apply parallel multigrid or another parallel iterative method to this situation. Here

TABLE 3
One possible strategy for overlapping the postprocessing and the solution phases.

Processor 0	Processor 1	Processor 2	Processor 3
Load balance	–	–	–
⇓	⇓	⇓	⇓
Solve P_1	Solve P_2	Solve P_3	Solve P_4
⇓	⇓	⇓	⇓
Glue $P_2 - P_4$	Solve P_5	Solve P_6	Solve P_7
⇓	⇓	⇓	⇓
Glue $P_5 - P_7$	Solve P_8	Solve P_9	Solve P_{10}
⇓	⇓	⇓	⇓
Glue $P_8 - P_{10}$	Solve P_{11}	Solve P_{12}	Solve P_{13}
⇓	⇓	⇓	⇓
Glue $P_{11} - P_{13}$	Solve P_{14}	Solve P_{15}	Solve P_{16}
⇓	⇓	⇓	⇓
Glue $P_{14} - P_{16}$	–	–	–
⇓	⇓	⇓	⇓
Global solve			

the parallel multigrid method of Mitchell [20, 21, 22] seems particularly appropriate. In any event, by creating a good initial guess from the solutions of the independent problems, very little work (e.g., few iterations) should be required to compute the global solution.

4.2. Mortar elements. In the 3D case, producing a global conforming mesh is much more problematic, in that face matching simply through bisection is not achievable as it is in the 2D case. This is a well-known problem, and impacts adaptive tetrahedral subdivision algorithms based on octasection of tetrahedra [39, 24, 9]. We consider two approaches for dealing with this difficultly in the present context. The first of these is the mortar elements, which is discussed here; the second approach is described in the next section.

One approach for making a global solution from subdomain solutions on nonconforming subdomains is to simply use the global nonconforming mesh and establish weak continuity of the solution on the interface using so-called *mortar elements* [8, 7]. Although originally developed as a technique to couple spectral and finite element methods, it can be used to couple finite element discretizations which are conforming within subdomains but have nonmatching meshes at the interfaces of the subdomains.

To keep the discussion simple, suppose that there are only two subregions with a single interface Γ. Let $u_h^{(1)}$ and $u_h^{(2)}$ denote the approximate solutions for the two subregions. Rather than forcing the mesh along the interface to become conforming, we impose continuity of the computed solution weakly, as

$$\int_\Gamma (u_h^{(1)} - u_h^{(1)})\phi \, dx = 0 \quad \text{for all } \phi \in \mathcal{V}, \tag{4.1}$$

where $\mathcal{V}$ is some suitably chosen mortar space. See [8, 7, 1, 6] for details on the selection of $\mathcal{V}$. When assembled, the resulting system of linear equations will have the classic saddle point structure

$$\begin{pmatrix} A_1 & 0 & B_1 \\ 0 & A_2 & B_2 \\ B_1^t & B_2^t & 0 \end{pmatrix} \begin{pmatrix} U_1 \\ U_2 \\ \Lambda \end{pmatrix} = \begin{pmatrix} R_1 \\ R_2 \\ 0 \end{pmatrix}, \tag{4.2}$$

where, as usual, A_i correspond to individual subregions and B_i corresponds to the coupling of the subregions through the mortar space $\mathcal{V}$. The space $\mathcal{V}$ plays the role of

Lagrange multipliers in the saddle point problem, and the element of $\mathcal{V}$ corresponding to the solution of (4.2) has a physical interpretation in terms of an approximation to the normal component of ∇u on Γ.

From its structure, it is clear that one may apply classical domain decomposition techniques to the solution of (4.2) (and the more general case of many subregions). All of the information related to each subdomain is already present on the processor responsible for adaptively creating that portion of the composite mesh. Communication between processors is necessary for coupling through the mortar elements, but as usual, the required information is related to the solution and the structure of the mesh only along the interface, generally a small amount of data in comparison with the size of the subdomains.

4.3. Overlapping decompositions and interior estimates. The simplest way to form a global solution is not to form one, meaning that the subdomain solutions themselves are taken to be the final discrete solution represented subdomainwise. To evaluate the solution at any point $x \in \bar{\Omega}$, one simply must determine which subdomain contains the point x and then fetch the solution value from the particular subdomain. While this approach seems naive, some recent [37] and not so recent [23, 28, 29] theoretical results actually support this.

The principle idea underlying the results in [37] is that while elliptic problems are globally coupled, this global coupling is essentially a "low-frequency" coupling, and can be handled on a mesh which is much coarser than that required for approximation accuracy considerations. This idea has been exploited for example in [19, 36], and is, in fact, why the construction of a coarse problem in overlapping domain decomposition methods is the key to obtaining convergence rates which are independent of the number of subdomains (cf. [35]).

The key results in [37] for our purposes are the following a priori and a posteriori error estimates. To explain, let Ω_k be the collection of disjoint subregions of the domain Ω defined by the weighted spectral bisection algorithm of the previous section, and let Ω_k^0 to be an extension of the disjoint Ω_k, such that $\Omega_k \subset\subset \Omega_k^0$, and so that the sizes of the overlap regions $\Omega_i^0 \bigcap \Omega_j^0$ are on the order of the sizes of the regions Ω_k. Under some reasonable assumptions about the approximation properties of a finite element space S_0^h defined over Ω (existence of superapproximation, inverse, and trace inequalities), the following a priori error estimate holds for the global Galerkin solution u_h to a Poisson-like linear elliptic equation:

$$\|u - u_h\|_{H^1(\Omega_k)} \leq C \left(\inf_{v \in S_0^h} \|u - v\|_{H^1(\Omega_k^0)} + \|u - u_h\|_{L^2(\Omega)} \right),$$

and the following a posteriori error estimate holds (where $\eta(u_h)$ is a locally computable jump function):

$$\|u - u_h\|_{H^1(\Omega_k)} \leq C \left(\|h\eta(u_h)\|_{L^2(\Omega_k^0)} + \|u - u_h\|_{L^2(\Omega)} \right).$$

The a priori result states that the error in the global Galerkin solution u_h restricted to a subdomain Ω_k can be bounded by the error in the best approximation from the finite element space S_0^h measured only over the extended subdomain Ω_k^0, plus a higher-order global term (the global L^2-norm of the error). In the context of the algorithm in this paper, if the global coarse mesh is quasi-uniform and shape regular with element diameter H, and if we assume that $u \in H^2(\Omega)$, then standard

interpolation theory and L^2-lifting can be used to bound the global term by an $O(H^2)$ factor. Moreover, if the mesh produced by adaptivity in the extended subdomain Ω_k^0 is again quasi-uniform and shape regular, but now has element diameter h, then the local term can be bounded using standard interpolation theory by an $O(h)$ factor. If our adaptive method respects the relationship $h = O(H^2)$ between the coarse and refined regions, then asymptotically the global term based on the much coarser mesh outside Ω_k^0 does not pollute the accuracy of the adapted solution in the subdomain Ω_k. A similar argument can be applied in the less regular case of $u \in H^{1+\alpha}(\Omega)$.

The a posteriori estimate states that the error in the global Galerkin solution restricted to a subdomain Ω_k can be estimated in terms of a (computable) jump function $\eta(\cdot)$ over the extended subdomain Ω_k^0, plus a higher-order term (the global L^2-norm of the error). Through the same argument above, this means that asymptotically, the global error can be controlled by the local computable jump function estimate in each subdomain, so that reliable a posteriori error estimates can be computed in isolation from the other subdomains.

The a priori and a posteriori estimates from [37] outlined above were derived for self-adjoint linear problems in the plane, and as a result they do not apply directly to the examples presented in this paper (nonlinear scalar problems and elliptic systems in both 2D and 3D). Moreover, the local refinement strategy described here tends to produce very little overlap of the extended subdomains Ω_k^0, violating one of the key assumptions in [37], and we do not explicitly enforce a refinement limitation such as $h = O(H^2)$. However, the estimates in [37] indicate that this approach will likely provide a very good initial approximation to an overlapping domain decomposition procedure for solving the final global problem, and in an ideal situation (certain classes of elliptic problems with large subdomain overlap), it might be possible to skip the global solution altogether.

REFERENCES

[1] Y. Achdou, Y. Kuznetsov, and O. Pironneau, *Substrucuring preconditioners for the Q1 mortar element method*, Numer. Math., 71 (1995), pp. 419–449.

[2] R. E. Bank, *PLTMG: A Software Package for Solving Elliptic Partial Differential Equations, Users' Guide* 8.0, Software, Environments, and Tools, 5, SIAM, Philadelphia, 1998.

[3] R. E. Bank, A. H. Sherman, and A. Weiser, *Refinement algorithms and data structures for regular local mesh refinement*, in IMACS Transactions on Scientific Computation I, R. S. Stepleman, ed., North–Holland, Amsterdam, 1983, pp. 3–17.

[4] R. E. Bank and R. K. Smith, *Mesh smoothing using a posteriori error estimates*, SIAM J. Numer. Anal., 34 (1997), pp. 979–997.

[5] M. W. Beall and M. S. Shephard, *A general topology-based mesh data structure*, Internat. J. Numer. Methods Engrg., 40 (1997), pp. 1573–1596.

[6] F. Belgacem, *The mortar finite element method with Lagrange multipliers*, Numer. Math. 84 (1999), pp. 173–197.

[7] C. Bernardi, N. Debit, and Y. Maday, *Coupling finite element and spectral methods: First results*, Math. Comp., 54 (1990).

[8] C. Bernardi, Y. Maday, and A. Patera, *A new nonconforming approach to domain decomposition: the mortar element method*, in Nonlinear Partial Differential Equations and Their Applications, H. B. and J.L. Lions, eds., Pitman Res. Notes Math. Ser. 302, Longman Scientific & Technical, Harlow; copublished with John Wiley, New York, 1994, pp. 13–51.

[9] J. Bey, *Tetrahedral grid refinement*, Computing, 55 (1995), pp. 355–378.

[10] X. Cai and K. Samuelsson, *Parallel Multilevel Methods with Adaptivity on Unstructured Grids*, preprint, 1999.

[11] T. F. Chan, P. Ciarlet, Jr., and W. K. Szeto, *On the optimality of the median cut spectral bisection graph partitioning method*, SIAM J. Sci. Comput., 18 (1997), pp. 943–948.

[12] H. L. deCougny, K. D. Devine, J. E. Flaherty, R. M. Loy, C. Ozturan, and M. S. Shephard, *Load balancing for the parallel adaptive solution of partial differential equations*,

Appl. Numer. Math., 16 (1994), pp. 157–182.
[13] J. E. Flaherty, R. M. Loy, C. Ozturan, M. S. Shephard, B. K. Szymanski, J. D. Teresco, and L. H. Ziantz, *Parallel structures and dynamic load balancing for adaptive finite element computation*, Appl. Numer. Math., 26 (1998), pp. 241–263.
[14] G. Fox, R. Williams, and P. Messina, *Parallel Computing Works!*, Morgan-Kaufmann, San Francisco, 1994.
[15] M. Holst, *Adaptive multilevel finite element methods on manifolds and their implementation in MC*, in preparation; currently available as a technical report and User's Guide to the MC software.
[16] M. Holst and D. Bernstein, *Finite element solution of the initial-value problem in general relativity: Theory and algorithms*, Comm. Math. Phys., 1999, submitted.
[17] S. Kohn, J. Weare, M. E. Ong, and S. B. Baden, *Software abstractions and computational issues in parallel structured adaptive mesh methods for electronic structure calculations*, in Proceedings of the Workshop on Structured Adaptive Mesh Refinement Grid Methods, Institute for Mathematics and Its Applications, University of Minnesota, Minneapolis, MN, 1997.
[18] A. Liu and B. Joe, *Relationship between tetrahedron shape measures*, BIT, 34 (1994), pp. 268–287.
[19] M. Marion and J. Xu, *Error estimates on a new nonlinear Galerkin method based on two-grid finite elements*, SIAM J. Numer. Anal., 32 (1995), pp. 1170–1184.
[20] W. Mitchell, *The full domain partition approach to distributing adaptive grids*, Appl. Numer. Math., 26 (1998), pp. 265–275.
[21] W. Mitchell, *A parallel multigrid method using the full domain partition*, Electron. Trans. Numer. Anal., 6 (1998), pp. 224–233.
[22] W. Mitchell, *The full domain partition approach to parallel adaptive refinement*, in Grid Generation and Adaptive Algorithms, IMA Vol. Math. Appl. 113, Springer-Verlag, New York, 1998, pp. 151–162.
[23] J. A. Nitsche and A. H. Schatz, *Interior estimates for Ritz-Galerkin methods*, Math. Comp., 28 (1974), pp. 937–958.
[24] M. E. G. Ong, *Uniform refinement of a tetrahedron*, SIAM J. Sci. Comput., 15 (1994), pp. 1134–1144.
[25] B. N. Parlett, *The Symmetric Eigenvalue Problem*, Prentice–Hall, Englewood Cliffs, NJ, 1980.
[26] A. K. Patra and D. W. Kim, *Efficient mesh partitioning for adaptive hp finite element meshes*, in Proceedings of the Eleventh International Conference on Domain Decomposition Methods, Greenwich, UK, 1998, C.-H. Lai, P. E. Björstad, M. Cross, and O. B. Widlund, eds., 1998.
[27] A. Pothen, H. D. Simon, and K.-P. Liou, *Partitioning sparse matrices with eigenvectors of graphs*, SIAM J. Matrix Anal. Appl., 11 (1990), pp. 430–452.
[28] A. H. Schatz and L. B. Wahlbin, *Interior maximum-norm estimates for finite element methods*, Math. Comp., 31 (1977), pp. 414–442.
[29] A. H. Schatz and L. B. Wahlbin, *Interior maximum-norm estimates for finite element methods* II, Math. Comp., 64 (1995), pp. 907–928.
[30] P. M. Selwood, M. Berzins, and P. M. Dew, *3D parallel mesh adaptivity: Data structures and algorithms*, in the Proceedings of the Eigth SIAM Conference on Parallel Processing for Scientific Computing, Minneapolis, MN, 1997, CD-ROM, SIAM, Philadelphia, PA, 1997.
[31] H. D. Simon and S.-H. Teng, *How good is recursive bisection?*, SIAM J. Sci. Comput., 18 (1997), pp. 1436–1445.
[32] R. Verfürth, *A Posteriori Error Estimation and Adaptive Mesh Refinement Techniques*, Teubner Skripten zur Numerik, G. B. Teubner, Stuttgart, 1995.
[33] C. Walshaw and M. Berzins, *Dynamic load balancing for pde solvers on adaptive unstructured meshes*, Concurrency: Practice and Experience, 7 (1995), pp. 17–28.
[34] R. Williams, *Performance of dynamic load balancing algorithms for unstructured mesh calculations*, Concurrency, 3 (1991), p. 457.
[35] J. Xu, *Iterative methods by space decomposition and subspace correction*, SIAM Rev., 34 (1992), pp. 581–613.
[36] J. Xu, *Two-grid discretization techniques for linear and nonlinear PDEs*, SIAM J. Numer. Anal., 33 (1996), pp. 1759–1777.

[37] J. Xu and A. Zhou, *Local and parallel finite element algorithms based on two-grid discretizations*, Math. Comp., to appear.
[38] J. W. York, *Kinematics and dynamics of general relativity*, in Sources of Gravitational Radiation, L. L. Smarr, ed., Cambridge University Press, Cambridge, MA, 1979, pp. 83–126.
[39] S. Zhang, *Multi-level Iterative Techniques*, Ph.D. thesis, Department of Mathematics, Pennsylvania State University, College Park, PA, 1988.

SIAM J. SCI. COMPUT.
Vol. 22, No. 4, pp. 1444–1460
© 2000 Society for Industrial and Applied Mathematics

FLEXIBLE CONJUGATE GRADIENTS*

YVAN NOTAY†

Abstract. We analyze the conjugate gradient (CG) method with preconditioning slightly variable from one iteration to the next. To maintain the optimal convergence properties, we consider a variant proposed by Axelsson that performs an explicit orthogonalization of the search directions vectors. For this method, which we refer to as *flexible* CG, we develop a theoretical analysis that shows that the convergence rate is essentially independent of the variations in the preconditioner as long as the latter are kept sufficiently small. We further discuss the real convergence rate on the basis of some heuristic arguments supported by numerical experiments. Depending on the eigenvalue distribution corresponding to the fixed reference preconditioner, several situations have to be distinguished. In some cases, the convergence is as fast with truncated versions of the algorithm or even with the standard CG method, whereas quite large variations are allowed without too much penalty. In other cases, the flexible variant effectively outperforms the standard method, while the need for truncation limits the size of the variations that can be reasonably allowed.

Key words. iterative methods for linear systems, acceleration of convergence, preconditioning

AMS subject classifications. 65F10, 65B99, 65N20

PII. S1064827599362314

1. Introduction.

As is well known, the conjugate gradient (CG) method combined with a suitable preconditioning is a choice method to solve large sparse $n \times n$ linear systems

$$A\,\mathbf{u} \;=\; \mathbf{b} \tag{1.1}$$

whose coefficient matrix A is symmetric and positive definite (e.g., [4, 9, 19, 21]). Ideally, a preconditioner is a symmetric and positive definite matrix B such that solving a system

$$B\,\mathbf{x} \;=\; \mathbf{y} \tag{1.2}$$

is relatively cheap, whereas the eigenvalues of $B^{-1}A$ are nicely clustered, favoring a rapid convergence.

Now, as preconditioning techniques become more sophisticated and apply to a wider class of problems, it is not unusual that the solution of (1.2) itself requires solving one or more subproblems for which, again, the preconditioned CG method is best suited. Although, from a theoretical point of view, it is more comfortable to assume that the system (1.2) is nevertheless solved accurately, in practice, the overall efficiency of the scheme may then critically depend on the ability to allow loose stopping criteria for the inner iterative solutions. In this case, the preconditioning step seen by the outer process is no longer $\mathbf{x} = B^{-1}\,\mathbf{y}$, but is

$$\mathbf{x} \;=\; \mathcal{B}(\mathbf{y}), \tag{1.3}$$

where $\mathcal{B}$ is a mapping from $\mathcal{R}^n$ to $\mathcal{R}^n$, in general nonlinear.

Early observation has indeed been made by Golub and Overton [17, 18] that the convergence rate of the outer CG process could be maintained even with very

*Received by the editors October 14, 1999; accepted for publication (in revised form) May 10, 2000; published electronically November 2, 2000. This research was supported by the "Fonds National de la Recherche Scientifique," Maître de recherches.

http://www.siam.org/journals/sisc/22-4/36231.html

†Service de Métrologie Nucléaire, Université Libre de Bruxelles (C.P. 165), 50, Av. F.D. Roosevelt, B-1050 Brussels, Belgium (ynotay@ulb.ac.be).

loose accuracy for the inner iterations. Unfortunately, much less is known from a theoretical point of view. An interesting analysis has been recently developed by Golub and Ye [20], but, as will be seen in section 3, the bound derived there tends to strongly overestimate nonlinearities effects. On the other hand, some insight can be gained by reexamining the studies on CG in the presence of errors [22, 23, 31, 26], letting the parameter governing the size of the errors be much larger than the roundoff unit considered in these works. However, only heuristic conclusions can be derived on this basis, and the latter display that the method may lose part of its efficiency when the extremal eigenvalues of $B^{-1}A$ are well separated.

In [4, p. 549], Axelsson proposed a variant of the CG algorithm that is specifically designed to accommodate variable preconditioners (see also Algorithm 5.3 in [8]). This algorithm automatically truncates to standard CG when using a fixed symmetric and positive definite preconditioner, whereas, in the general case, optimal convergence property in A-norm is preserved at the price of performing a full orthogonalization of the search direction vectors (with respect to the $(\cdot, A\,\cdot)$ inner product). Although this method is a particular case of the generalized CG (GCG) algorithm developed in [1, 2, 3, 7], we refer to it as *flexible* CG (FCG).[1]

Similar to any method not based on a short recurrence, FCG has to be combined in practice with a truncation or a restart strategy. In this respect, it is interesting to note that with maximal truncation one recovers the steepest descent algorithm, whereas adding one orthogonality constraint delivers an implementation of the usual CG method, in fact the one observed in [20] to be the most stable with respect to variations in the preconditioner.

Concerning the theoretical analysis, the results in the latter paper prove fast convergence only when the perturbations are below some given threshold, and the analysis fails otherwise. One may then resort to the general results on GCG with variable preconditioning [4, 7], which prove the well definiteness of the method and its convergence under rather weak assumptions on $\mathcal{B}$. However, the proved speed of convergence is clearly much too pessimistic in the case of FCG; see section 3 for details. On the other hand, from a practical point of view, no discussion seems available on which restart or truncation strategy is advisable, and on which benefit can be hoped by using this flexible variant instead of the standard CG algorithm.

In the present paper, we aim at filling these gaps. We first propose in section 2 a mixed truncation–restart strategy that combines the advantages of both pure truncation and pure restart. We next consider in section 3 individual steps of FCG and prove a bound on the decrease of the error between two successive steps that tends to the optimal bound as $\mathcal{B} \to B^{-1}$, while allowing quite large variations in the preconditioner. We further consider in section 4 several successive steps of the untruncated algorithm and show that there exists a matrix $\widehat{B}$ such that $\mathcal{B}(\mathbf{y}) = \widehat{B}\,\mathbf{y}$ for all vectors $\mathbf{y}$ to which $\mathcal{B}$ is effectively applied during these steps. Moreover, when $\mathcal{B}$ is close to some symmetric and positive definite matrix B^{-1}, the spectrum of $\widehat{B}\,A$ is only a slight perturbation of that of $B^{-1}A$. Finally, we discuss in section 5 the effects of truncation and compare FCG with the standard CG algorithm in view of some illustrative numerical examples. Our conclusions are summarized in section 6.

Concerning the use of variable preconditioning in combination with other methods for the outer iterations, we refer the reader to the literature: Chebyshev and Richardson iterations are considered in [16, 17, 18], the Uzawa algorithm is analyzed

[1] *Generalized* CG mostly refers to methods designed to solve unsymmetric systems, whereas the specific variant considered here solely permits the use of variable preconditioners in CG.

in [15], and a flexible variant of GMRES is discussed in [28, 34] (see also [29]).

Notation. Throughout this paper, A and B are $n \times n$ symmetric and positive definite matrices, and $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$ are the eigenvalues of $B^{-1}A$ in increasing order, whereas

$$\kappa = \frac{\lambda_n}{\lambda_1} \tag{1.4}$$

is the spectral condition number.

For any symmetric and positive definite matrix matrix C, $\|\cdot\|_C$ is the C-norm, that is, the norm associated with the $(\cdot, C\,\cdot)$ inner product: $\|\mathbf{v}\|_C = \sqrt{(\mathbf{v},\, C\,\mathbf{v})}$ for all $\mathbf{v}$. When G is a matrix, $\|G\|_C$ is the matrix norm induced by this vector norm: $\|G\|_C = \max_{\mathbf{z}\neq 0} (\|G\,\mathbf{z}\|_C / \|\mathbf{z}\|_C)$.

2. Algorithm and basic properties. We recall below (Algorithm 2.1) the method referred to as FCG in the introduction, that is, Algorithm 5(a) from [4, p. 549] or Algorithm 5.3 from [8]. As already stated, the main difference with standard CG lies in the explicit orthogonalization of the search direction vectors $\mathbf{d}_i$; more precisely, $\mathbf{d}_i$ is orthogonalized with respect to $(\cdot, A\,\cdot)$ against the m_i previous vectors, where $\{m_i\}_{i=0,1,\ldots}$ is a sequence of truncation parameters. The untruncated version corresponds to $m_i = i$ for all i, whereas the natural choice for m_i would be $m_i = \min(i,\, m_{\max})$ (pure truncation) or sequences like $m_i = 0, 1, \ldots, m_{\max}, 0, 1 \ldots, m_{\max}$, which would mean restarting the (untruncated) algorithm at each $m_{\max}+1$ iteration. However, as we discuss below, one should avoid using m_i less than 1 for any $i > 0$. This observation leads to the mixed truncation–restart strategy advised in Algorithm 2.1, which we generally observed to be more cost efficient than pure truncation (see section 5.3). In Algorithm 2.1 this specific truncation–restart strategy will be referred to as FCG($m_{\max}$).

ALGORITHM 2.1 (FCG).
Flexible Conjugate Gradient

- *Initialization*

$$\begin{aligned} \mathbf{u}_0 &\quad \text{arbitrary} \\ \mathbf{r}_0 &= \mathbf{b} - A\,\mathbf{u}_0 \end{aligned}$$

- *Iteration* $(i = 0, 1, \ldots)$

$$\begin{aligned} \mathbf{w}_i &= \mathcal{B}(\mathbf{r}_i) \\ \mathbf{d}_i &= \mathbf{w}_i - \sum_{k=i-m_i}^{i-1} \frac{(\mathbf{w}_i,\, A\,\mathbf{d}_k)}{(\mathbf{d}_k,\, A\,\mathbf{d}_k)}\,\mathbf{d}_k \\ \mathbf{u}_{i+1} &= \mathbf{u}_i + \frac{(\mathbf{d}_i, \mathbf{r}_i)}{(\mathbf{d}_i,\, A\,\mathbf{d}_i)}\,\mathbf{d}_i \\ \mathbf{r}_{i+1} &= \mathbf{r}_i - \frac{(\mathbf{d}_i, \mathbf{r}_i)}{(\mathbf{d}_i,\, A\,\mathbf{d}_i)}\,A\,\mathbf{d}_i \end{aligned}$$

Advised truncation strategy (FCG($m_{\max}$)):
$m_0 = 0;\ m_i = \max\big(1,\, \mathrm{mod}(i,\, m_{\max}+1)\big) \quad (i > 0)$
$m_{\max}$: *nonnegative integer.*

Assuming A symmetric and positive definite and $0 \leq m_{i+1} \leq m_i + 1$ for all i, it is easy to prove that (see [4, 7])

$$(\mathbf{d}_k,\, A\,\mathbf{d}_j) = 0 \quad \text{for all } j,\, k \text{ such that } i - m_i \leq j < k \leq i, \tag{2.1}$$

$$(\mathbf{r}_k,\, \mathbf{d}_j) = 0 \quad \text{for all } j,\, k \text{ such that } i - m_i \leq j < k \leq i + 1 \tag{2.2}$$

and that, letting $\mathbf{u} = A^{-1}\,\mathbf{b}$,

$$\|\mathbf{u} - \mathbf{u}_{i+1}\|_A \;=\; \min_{\mathbf{d} \in \mathrm{span}\{\mathbf{d}_{i-m_i},\, \ldots,\, \mathbf{d}_i\}} \|\mathbf{u} - \mathbf{u}_{i-m_i} - \mathbf{d}\|_A. \tag{2.3}$$

If $m_i = i$ (no truncation), these are the optimality properties satisfied by the standard CG method. Moreover, when $\mathcal{B} \equiv B^{-1}$ is a symmetric and positive definite matrix, it can be proved that the recursion defining $\mathbf{d}_i$ automatically truncates because $(\mathbf{w}_i,\, A\,\mathbf{d}_k) = 0$ for all $k < i - 1$. Algorithm 2.1 simplifies then to a particular implementation of the standard CG method, in fact, one of the variants discussed in the early works on CG [24, 27]. This variant is also referred to as inexact preconditioned CG (IPCG) in [20], because there it is observed to be the most stable with respect to variations in the preconditioner. Note, however, that it requires computing one more inner product per iteration than the standard algorithm described in most textbooks (e.g., [4, 9, 19, 21]). In the remainder of this paper, we use "standard CG" to refer to the latter algorithm, whereas the alternative implementation offered by Algorithm 2.1 with $m_{\max} = 1$ will be referred to as FCG(1).

Note that our motivation to propose a strategy in which m_i is never less than 1 originates in this close relation between FCG and CG. Indeed, we then have the guarantee that one will recover the usual CG convergence when $\mathcal{B}$ is a fixed preconditioner, whereas something would irremediably be lost with the pure restart that results when setting $m_i = 0$ for some $i > 0$.

3. Convergence analysis. Equation (2.3) proves that the error measured in the A-norm cannot increase, but it says nothing about the convergence rate. In [4, 7], it is proved that

$$\frac{\|\mathbf{u} - \mathbf{u}_{i+1}\|_A}{\|\mathbf{u} - \mathbf{u}_i\|_A} \;\leq\; \sqrt{1 - \left(\frac{\delta_2}{\delta_1}\right)^2}, \tag{3.1}$$

where

$$\delta_1 = \max_{\mathbf{v} \neq 0} \frac{(\mathbf{v},\, \mathcal{B}(\mathbf{v}))}{(\mathbf{v},\, A^{-1}\,\mathbf{v})}, \qquad \delta_2 = \min_{\mathbf{v} \neq 0} \sqrt{\frac{(\mathcal{B}(\mathbf{v}),\, A\,\mathcal{B}(\mathbf{v}))}{(\mathbf{v},\, A^{-1}\,\mathbf{v})}}.$$

In this analysis, the requirements on $\mathcal{B}$ are thus quite weak, but, unfortunately, the proved speed of convergence is likely to be much too pessimistic. This is easily seen by inspecting the case $\mathcal{B} \equiv B^{-1}$. Then, $\delta_1 = \lambda_n$, $\delta_2 = \lambda_1$, and (3.1) writes

$$\frac{\|\mathbf{u} - \mathbf{u}_{i+1}\|_A}{\|\mathbf{u} - \mathbf{u}_i\|_A} \;\leq\; \sqrt{1 - \kappa^{-2}}, \tag{3.2}$$

whereas the best available bound on the "local" decrease of the error is

$$\frac{\|\mathbf{u} - \mathbf{u}_{i+1}\|_A}{\|\mathbf{u} - \mathbf{u}_i\|_A} \;\leq\; \frac{\kappa - 1}{\kappa + 1}. \tag{3.3}$$

By way of comparison, for $\kappa = 9$, the latter results prove $\|\mathbf{u}-\mathbf{u}_{i+1}\|_A \leq 0.8\,\|\mathbf{u}-\mathbf{u}_i\|_A$, whereas (3.2) yields only $\|\mathbf{u}-\mathbf{u}_{i+1}\|_A \leq 0.994\,\|\mathbf{u}-\mathbf{u}_i\|_A$.

Except for very well conditioned problems, the bound (3.1) is therefore of little help to understand the real effects of the nonlinearities in $\mathcal{B}$. In the following theorem, we also bound the "local" decrease of the error, but with an expression that tends to (3.3) as $\mathcal{B} \to B^{-1}$.

THEOREM 3.1. *Let A, B be $n\times n$ symmetric and positive definite matrices and $\mathcal{B}$ a mapping from $\mathcal{R}^n$ to $\mathcal{R}^n$. Let $\mathbf{b}$, $\mathbf{u}_0$ be vectors of $\mathcal{R}^n$, and let $\{\mathbf{r}_i\}_{i=0,1,\ldots}$, $\{\mathbf{d}_i\}_{i=0,1,\ldots}$, $\{\mathbf{u}_i\}_{i=1,2,\ldots}$ be the sequences of vectors generated by applying Algorithm 2.1 to A, $\mathcal{B}$, $\mathbf{b}$, and $\mathbf{u}_0$ with some given sequence of nonnegative integer parameters $\{m_i\}_{i=0,1,\ldots}$.*

If, for any i,

$$\frac{\|\mathcal{B}(\mathbf{r}_i) - B^{-1}\,\mathbf{r}_i\|_B}{\|B^{-1}\,\mathbf{r}_i\|_B} \;\leq\; \varepsilon_i \;<\; 1, \tag{3.4}$$

then

$$\frac{\|\mathbf{u}-\mathbf{u}_{i+1}\|_A}{\|\mathbf{u}-\mathbf{u}_i\|_A} \;\leq\; \sqrt{1-\frac{4\,\kappa\,(1-\varepsilon_i)^2}{(\kappa+\varepsilon_i^2(\kappa-1)+(1-\varepsilon_i)^2)^2}} \;\leq\; \frac{\kappa\,\frac{1+\varepsilon_i}{1-\varepsilon_i}\,\frac{\left(1+\varepsilon_i^2\right)^2}{1-\varepsilon_i^2}-1}{\kappa\,\frac{1+\varepsilon_i}{1-\varepsilon_i}\,\frac{(1+\varepsilon_i^2)^2}{1-\varepsilon_i^2}+1}. \tag{3.5}$$

Proof. Clearly, by (2.3), for any real α,

$$\|\mathbf{u}-\mathbf{u}_{i+1}\|_A \;\leq\; \|\mathbf{u}-\mathbf{u}_i-\alpha\,\mathcal{B}(\mathbf{r}_i)\|_A \;=\; \|\mathbf{r}_i-\alpha\,A\,\mathcal{B}(\mathbf{r}_i)\|_{A^{-1}}.$$

Then let $\mathbf{t}_i = B^{-1}\,\mathbf{r}_i - \mathcal{B}(\mathbf{r}_i)$. One has

$$(\mathbf{t}_i,\,A\,\mathbf{t}_i) \;=\; \frac{(\mathbf{t}_i,\,A\,\mathbf{t}_i)}{(\mathbf{t}_i,\,B\,\mathbf{t}_i)}\,(\mathbf{t}_i,\,B\,\mathbf{t}_i) \;\leq\; \lambda_n\,\varepsilon_i^2\,(\mathbf{r}_i,\,B^{-1}\,\mathbf{r}_i),$$

whereas

$$|\,\big((I-\alpha\,A\,B^{-1})\mathbf{r}_i,\,\mathbf{t}_i\big)\,| \;\leq \|\mathbf{t}_i\|_B\,\|B^{-1}\,(I-\alpha\,A\,B^{-1})\mathbf{r}_i\|_B \;\leq\; \varepsilon_i\,\beta\,(\mathbf{r}_i,\,B^{-1}\,\mathbf{r}_i),$$

where $\beta = \|I-\alpha\,A\,B^{-1}\|_{B^{-1}} = \max(|1-\alpha\,\lambda_1|,\,|1-\alpha\,\lambda_n|)$. Hence, for $\alpha \geq 0$,

$$\begin{aligned}\|\mathbf{r}_i-\alpha\,A\,\mathcal{B}(\mathbf{r}_i)\|_{A^{-1}}^2 &= \|(I-\alpha\,A\,B^{-1})\mathbf{r}_i\|_{A^{-1}}^2 + 2\,\alpha\,\big((I-\alpha\,A\,B^{-1})\mathbf{r}_i,\,\mathbf{t}_i\big) + \alpha^2\,\|\mathbf{t}_i\|_A \\ &\leq \big(\mathbf{r}_i,\,(I-\alpha\,B^{-1}\,A)A^{-1}(I-\alpha\,A\,B^{-1})\,\mathbf{r}_i\big) \\ &\quad + \big(2\,\alpha\,\beta\,\varepsilon_i+\alpha^2\,\lambda_n\,\varepsilon_i^2\big)\;\;\big(\mathbf{r}_i,\,B^{-1}\,\mathbf{r}_i\big)\,.\end{aligned}$$

Clearly, the right-hand side of the latter inequality cannot be larger than $(1-\eta_i)\|\mathbf{u}-\mathbf{u}_i\|_A^2 = (1-\eta_i)(\mathbf{r}_i,\,A^{-1}\,\mathbf{r}_i)$ if the matrix

$$(1-\eta_i)\,A^{-1} \;-\; (I-\alpha\,B^{-1}\,A)A^{-1}(I-\alpha\,A\,B^{-1}) \;-\; \big(2\,\alpha\,\beta\,\varepsilon_i+\alpha^2\,\lambda_n\,\varepsilon_i^2\big)\;B^{-1}$$

is nonnegative definite. This holds if and only if

$$-\alpha^2\,\lambda^2 \;+\; \big(2\,\alpha-2\,\alpha\,\beta\,\varepsilon_i-\alpha^2\,\lambda_n\,\varepsilon_i^2\big)\;\lambda \;-\; \eta_i \;\geq 0 \quad \text{for all } \lambda \in \sigma\left(B^{-1}A\right). \tag{3.6}$$

Consider now

$$\alpha = 2\,\left(\tfrac{\lambda_n+\varepsilon_i^2(\lambda_n-\lambda_1)}{1-\varepsilon_i}+(1-\varepsilon_i)\,\lambda_1\right)^{-1}.$$

One has $0 < \alpha \leq 2(\lambda_n + \lambda_1)^{-1}$ and therefore $\beta = 1 - \alpha\,\lambda_1$, yielding

$$2\,\alpha - 2\,\alpha\,\beta\,\varepsilon_i - \alpha^2\,\lambda_n\,\varepsilon_i^2 \;=\; \alpha\,(2\,(1-\varepsilon_i) + \alpha\,\varepsilon_i\,(2\lambda_1 - \varepsilon_i\,\lambda_n)) \;=\; \alpha^2\,(\lambda_n + \lambda_1).$$

Hence, the sum of the roots of the polynomial in the left-hand side of (3.6) is just $(\lambda_n + \lambda_1)$; (3.6) therefore holds for $\eta_i = \lambda_1\,\lambda_n\,\alpha^2$, since these two roots are then precisely equal to λ_1 and λ_n, respectively; the left inequality (3.5) readily follows. To prove the right one, note that

$$\frac{\kappa\,(1-\varepsilon_i)^2}{\left(\kappa + \varepsilon_i^2(\kappa-1) + (1-\varepsilon_i)^2\right)^2} \;=\; \frac{\kappa\,\frac{1+\varepsilon_i}{1-\varepsilon_i}\,\frac{\left(1+\varepsilon_i^2\right)^2}{1-\varepsilon_i^2}}{\left(\kappa\,\frac{1+\varepsilon_i}{1-\varepsilon_i}\,\frac{(1+\varepsilon_i^2)^2}{1-\varepsilon_i^2} \;+\; \frac{(1+\varepsilon_i^2)(1-2\,\varepsilon_i)}{(1-\varepsilon_i)^2}\right)^2}$$

whereas $\frac{(1+\varepsilon_i^2)(1-2\,\varepsilon_i)}{(1-\varepsilon_i)^2} \leq 1$ holds since $(1-2\,\varepsilon_i) + \varepsilon_i^2\,(1-2\,\varepsilon_i) \leq 1 - 2\,\varepsilon_i + \varepsilon_i^2$. ∎

We now comment on the assumption (3.4), which will also be used in the next section. It will require that the relative error for the approximate solution of (1.2) is less than ε_i in the B-norm. This latter norm is the most natural if a fixed number of inner CG iterations is performed with B as system matrix, since standard analysis will then result in a bound on this measure of the error. Even when the stopping criterion for the inner iterations is based on the relative residual error, it remains that the error in B-norm is guaranteed to decrease monotonically. Further, in practice, it often happens that all relative measures of the error decrease more or less similarly. Hence, *on average*, one may expect ε_i to be close to the user prescribed tolerance. In this respect, note that we do not require ε_i to be less than 1 at *each* iteration. Thus, even from a purely theoretical point of view, it is not necessary to use a tolerance on the relative residual error that is small enough to guarantee some uniform bound on ε_i.

Now, one should remember that, in general, inner iterations will not be performed directly to solve $B\,\mathbf{x} = \mathbf{y}$, but rather will be involved in some subproblem(s) whose solution is needed for the computation of $\mathbf{x}$. Then an analysis is required to check that a small error for this (these) subproblem(s) cannot be magnified and result in a large global error on $\mathbf{x}$. Of course, such an analysis is application dependent and lies therefore outside the scope of the present paper.

Finally, it is interesting to compare our bound (3.5) with the one recently obtained by Golub and Ye [20, Theorem 3.6]. This comparison is possible because they measure the errors with a criterion similar to (3.4), which is even identical when $\mathcal{B}$ is defined by inner CG iterations with B as the system matrix. However, they consider *two* successive steps of FCG(1) (referred to as IPCG), and their bound, for $\varepsilon \to 0$, tends to the standard bound for two CG iterations. The latter is of course better than the bound for the two steepest descent iterations, and our result therefore cannot compete for small ε. Nevertheless, as seen in Figure 1, Golub–Ye analysis strongly overestimates perturbations effects, so that our bound becomes better for ε larger than approximately 0.05.

4. Equivalent linear operator. The analysis of inexact preconditioning is intrinsically difficult because $\mathcal{B}$ is in general nonlinear and therefore cannot be represented in standard matrix form. Indeed, it would be too restrictive to assume $\mathcal{B}(\mathbf{x}+\mathbf{y}) = \mathcal{B}(\mathbf{x}) + \mathcal{B}(\mathbf{y})$ for any $\mathbf{x}$, $\mathbf{y}$. However, we do not in fact need a matrix representation of $\mathcal{B}(\mathbf{x})$ valid for any $\mathbf{x}$: Our interest is restricted to the vectors $\mathbf{r}_0$, $\mathbf{r}_1$, ... to which $\mathcal{B}$ is effectively applied during the course of the iterations. Moreover, to

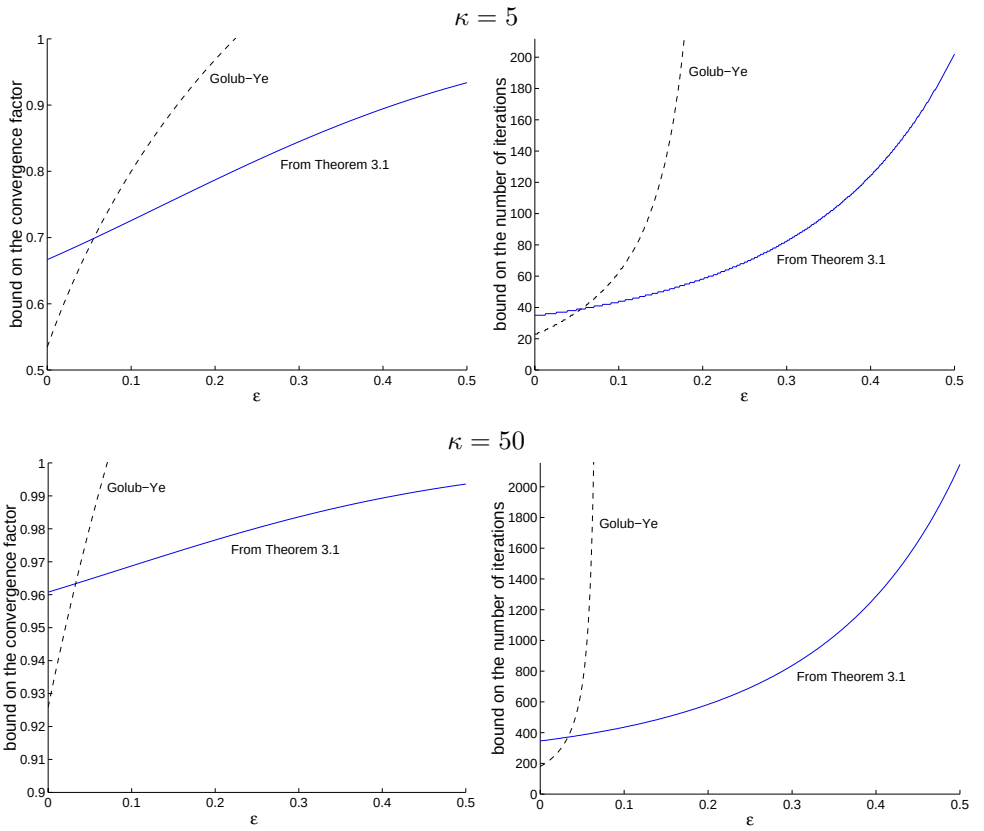


FIG. 1. *Left: convergence factor as estimated by* (3.5) *(solid line) and by Golub–Ye analysis (dashed line). Right: corresponding bound on the number of iterations to reduce the relative error in A-norm by* 10^{-6}.

analyze $\|\mathbf{u} - \mathbf{u}_{i+1}\|$ in view of (2.3), it is not useful to know something about $\mathcal{B}(\mathbf{r}_j)$ for all j. We have only to find a matrix $\widehat{B}_{i,\ell}$ such that

$$\mathcal{B}(\mathbf{r}_j) \;=\; \widehat{B}_{i,\ell}\,\mathbf{r}_j \qquad \text{for } j = i-\ell,\ldots,i \tag{4.1}$$

with $\ell \le m_i$.

It is rather obvious that such a matrix exists as long as the vectors $\mathbf{r}_{i-\ell}, \ldots, \mathbf{r}_i$ are linearly independent (there are even infinitely many possibilities if $\ell + 1 < n$). When using a fixed preconditioner, it is known that this linear independence holds because $(\mathbf{r}_j, B^{-1}\mathbf{r}_k) = 0$ for all $j \neq k$. In the general case, the linear independence is easily proved as soon as Theorem 3.1 (or (3.1)) applies to guarantee $\|\mathbf{r}_i\|_{A^{-1}} < \|\mathbf{r}_{i-1}\|_{A^{-1}} < \cdots < \|\mathbf{r}_{i-\ell}\|_{A^{-1}}$. Indeed, if $\sum_{k=j}^{i} \xi_k \mathbf{r}_k = 0$ with $\xi_j \neq 0$ for some $j \ge i-\ell$, one should have $(\mathbf{r}_j, \mathbf{d}_j) = 0$ (since by (2.2) $(\mathbf{r}_k, \mathbf{d}_j) = 0$ for $k = j+1, \ldots, i$). But this would imply $\mathbf{r}_{j+1} = \mathbf{r}_j$ (see Algorithm 2.1), and this stagnation contradicts the assumption that the error is strictly decreasing.

Now, to gain a further insight, one has to analyze how close the eigenvalues of $\widehat{B}_{i,\ell}A$ are to those of the "target" preconditioned system matrix $B^{-1}A$. This is the purpose of the next theorem. For technical reasons, we assume $m_i = i$ for all i; that is, we limit the analysis to the untruncated version of Algorithm 2.1. We believe that

this is not too severe a limitation because, as discussed below, one should expect from the theorem more qualitative than quantitative insight.

THEOREM 4.1. *Let the assumptions of Theorem* 3.1 *hold, and assume in addition that* $m_i = i$ *for all* i *in Algorithm* 2.1. *Let* i, ℓ *be some nonnegative integers such that* $\ell \leq i$. *For* $j = i - \ell, \ldots, i$, *let*

$$\mathbf{t}_j \;=\; B^{-1}\,\mathbf{r}_j - \mathcal{B}(\mathbf{r}_j), \tag{4.2}$$

and let ε, γ_t, *and* γ_r *be such that*

$$\frac{\|\mathbf{t}_j\|_B}{\|B^{-1}\,\mathbf{r}_j\|_B} \leq \varepsilon, \tag{4.3}$$

$$\frac{|(\mathbf{t}_j,\, B\,\mathbf{t}_k)|}{\|\mathbf{t}_j\|_B\,\|\mathbf{t}_k\|_B} \leq \gamma_t, \qquad k = j+1, \ldots, i, \tag{4.4}$$

$$\frac{|(\mathbf{t}_j,\, \mathbf{r}_k)|}{\|\mathbf{t}_j\|_B\,\|B^{-1}\,\mathbf{r}_k\|_B} \leq \gamma_r, \qquad k = j+1, \ldots, i. \tag{4.5}$$

If

$$\widehat{\varepsilon} \;=\; \varepsilon\sqrt{\frac{1+\gamma_t\,\ell}{1-\varepsilon\,\gamma_r\,\ell}} \;<\; 1, \tag{4.6}$$

then there exists a matrix $\widehat{B}_{i,\ell}$ *satisfying*

$$\widehat{B}_{i,\ell}\,\mathbf{r}_j \;=\; \mathcal{B}(\mathbf{r}_j), \qquad j = i-\ell, \ldots, i, \tag{4.7}$$

and such that, noting $W(C)$ *the field of value of the matrix* C, *one has*

$$W\left(A^{1/2}\widehat{B}_{i,\ell}A^{1/2}\right) \;\subset\; \left\{\overline{\lambda}\in\mathcal{C} \;:\; \left|\frac{\overline{\lambda}}{\lambda}-1\right| \leq \widehat{\varepsilon} \text{ for } \lambda \in W\left(A^{1/2}B^{-1}A^{1/2}\right)\right\} \tag{4.8}$$

and

$$\sigma\left(\widehat{B}_{i,\ell}\,A\right) \;\subset\; \bigcup_{k=1}^{n}\mathcal{D}_k, \tag{4.9}$$

where

$$\mathcal{D}_k \;=\; \left\{\overline{\lambda}\in\mathcal{C} \;:\; \left|\overline{\lambda}-\lambda_k\right| \leq \widehat{\varepsilon}\,\lambda_n\right\} \tag{4.10}$$

is the disk of radius $\widehat{\varepsilon}\,\lambda_n$ *centered at the* k*th eigenvalue of* $B^{-1}A$. *Moreover, if* s *such disks form a connected domain isolated from the remainder, there are precisely* s *eigenvalues in this domain.*

Finally, letting $\mathbf{z}_k$ *be an eigenvector of* $A^{1/2}\,B^{-1}\,A^{1/2}$ *associated with eigenvalue* λ_k, *there holds*

$$\left|\frac{\left(\mathbf{z}_k,\, A^{1/2}\,\widehat{B}_{i,\ell}\,A^{1/2}\,\mathbf{z}_k\right)}{(\mathbf{z}_k,\,\mathbf{z}_k)} - \lambda_k\right| \;\leq\; \widehat{\varepsilon}\,\lambda_k. \tag{4.11}$$

Proof. With (4.3), one easily sees that Theorem 3.1 applies, and therefore that $\mathbf{r}_{i-\ell}, \ldots, \mathbf{r}_i$ are linearly independent as shown above. There exists then a unique $n \times n$ matrix $\widehat{B}_{i,\ell}$ satisfying (4.7) and

$$\widehat{B}_{i,\ell}\,\mathbf{v} \;=\; B^{-1}\,\mathbf{v} \qquad \text{for all } \mathbf{v}:\ (\mathbf{v},\, B^{-1}\mathbf{r}_j) = 0,\ j = i-\ell, \ldots, i. \tag{4.12}$$

Let $E = \widehat{B}_{i,\ell} - B^{-1}$. First, (4.9) and the subsequent assertion on the localization of the eigenvalues follow from a standard result of perturbation theory [35, p. 88] if $\|A^{1/2}\,E\,A^{1/2}\|_2 \leq \widehat{\varepsilon}\,\lambda_n$. Now,

$$\begin{aligned}
\|A^{1/2}\,E\,A^{1/2}\|_2^2 &= \max_{\mathbf{z}\neq 0} \frac{(\mathbf{z},\, A^{1/2}\,E^T\,A\,E\,A^{1/2}\,\mathbf{z})}{(\mathbf{z},\,\mathbf{z})} \\
&= \max_{\mathbf{z}\neq 0} \frac{(E\,\mathbf{z},\, A\,E\,\mathbf{z})}{(\mathbf{z},\, A^{-1}\,\mathbf{z})} \\
&\leq \lambda_n^2\ \max_{\mathbf{z}\neq 0} \frac{(E\,\mathbf{z},\, B\,E\,\mathbf{z})}{(\mathbf{z},\, B^{-1}\,\mathbf{z})}
\end{aligned}$$

and (4.9) holds if

$$\max_{\mathbf{z}\neq 0} \sqrt{\frac{(E\,\mathbf{z},\, B\,E\,\mathbf{z})}{(\mathbf{z},\, B^{-1}\,\mathbf{z})}} \;=\; \|B\,E\|_{B^{-1}} \;\leq\; \widehat{\varepsilon}. \tag{4.13}$$

Likewise, letting $\mathbf{v} = A^{1/2}\mathbf{z}$,

$$\frac{(\mathbf{z},\, A^{1/2}\,\widehat{B}_{i,\ell}\,A^{1/2}\,\mathbf{z})}{(\mathbf{z},\,\mathbf{z})} \;=\; \frac{(\mathbf{v},\, \widehat{B}_{i,\ell}\,\mathbf{v})}{(\mathbf{v},\, A^{-1}\,\mathbf{v})} \;=\; \frac{(\mathbf{v},\, B^{-1}\,\mathbf{v})}{(\mathbf{v},\, A^{-1}\,\mathbf{v})}\left(1 + \frac{(\mathbf{v},\, E\,\mathbf{v})}{(\mathbf{v},\, B^{-1}\,\mathbf{v})}\right),$$

and, noting that $\frac{(\mathbf{v},\, B^{-1}\,\mathbf{v})}{(\mathbf{v},\, A^{-1}\,\mathbf{v})} \in [\lambda_1,\, \lambda_n] = W\left(A^{1/2}B^{-1}A^{1/2}\right)$, we have that (4.8) follows if (4.13) holds since

$$\max_{\mathbf{v}\neq 0} \frac{|(\mathbf{v},\, E\,\mathbf{v})|}{(\mathbf{v},\, B^{-1}\,\mathbf{v})} \;\leq\; \max_{\mathbf{v}\neq 0} \frac{\|\mathbf{v}\|_{B^{-1}}\,\|B\,E\,\mathbf{v}\|_{B^{-1}}}{\|\mathbf{v}\|_{B^{-1}}^2} \;=\; \|B\,E\|_{B^{-1}}. \tag{4.14}$$

Further, noting that $\mathbf{v}_k = A^{1/2}\,\mathbf{z}_k$ satisfies $A^{-1}\,\mathbf{v}_k = \lambda_k^{-1}\,B^{-1}\,\mathbf{v}_k$, one has

$$\begin{aligned}
\left|\frac{\left(\mathbf{z}_k,\, A^{1/2}\,\widehat{B}_{i,\ell}\,A^{1/2}\,\mathbf{z}_k\right)}{(\mathbf{z}_k,\,\mathbf{z}_k)} - \lambda_k\right| &= \frac{|\left(\mathbf{z}_k,\, A^{1/2}\,E\,A^{1/2}\,\mathbf{z}_k\right)|}{(\mathbf{z}_k,\,\mathbf{z}_k)} \\
&= \frac{|\,(\mathbf{v}_k,\, E\,\mathbf{v}_k)\,|}{(\mathbf{v}_k,\, A^{-1}\,\mathbf{v}_k)} \\
&= \lambda_k\ \frac{|\,(\mathbf{v}_k,\, E\,\mathbf{v}_k)\,|}{(\mathbf{v}_k,\, B^{-1}\,\mathbf{v}_k)},
\end{aligned}$$

showing with (4.14) that (4.11) also holds when (4.13) is satisfied.

Hence we are left with proving (4.13). In this view, note first that by (4.12) we may restrict the maximum in the left-hand side of (4.13) to vectors of the form $\mathbf{z} = \sum_{k=i-\ell}^{i} \xi_k \mathbf{r}_k$. Let then, for $j_0 = 1, \ldots, \ell+1$,

$$y_{j_0} \;=\; \|\mathbf{r}_{j_0+i-\ell-1}\|_{B^{-1}}\,\xi_{j_0+i-\ell-1}$$

and set $y = \begin{pmatrix} y_1 & y_2 & \dots & y_{\ell+1} \end{pmatrix}^T$. Note that, for $i - \ell \le k \le i$, $E\,\mathbf{r}_k = \widehat{B}_{i,\ell}\,\mathbf{r}_k - B^{-1}\,\mathbf{r}_k = -\mathbf{t}_k$, one has

$$\frac{(E\,\mathbf{z},\, B\,E\,\mathbf{z})}{(\mathbf{z},\, B^{-1}\,\mathbf{z})} \;=\; \frac{y^T\,G\,y}{y^T\,H\,y},$$

where G, H are the symmetric $(\ell+1) \times (\ell+1)$ matrices with components

$$G_{j_0k_0} = \frac{(\mathbf{t}_j,\, B\,\mathbf{t}_k)}{\|\mathbf{r}_j\|_{B^{-1}}\,\|\mathbf{r}_k\|_{B^{-1}}},$$
$$H_{j_0k_0} = \frac{(\mathbf{r}_j,\, B^{-1}\,\mathbf{r}_k)}{\|\mathbf{r}_j\|_{B^{-1}}\,\|\mathbf{r}_k\|_{B^{-1}}}$$

(letting $j = j_0 + i - \ell - 1$ and $k = k_0 + i - \ell - 1$).

Now, (4.3) and (4.4) straightforwardly yield $G_{j_0j_0} \le \varepsilon^2$ and $G_{j_0k_0} \le \varepsilon^2\,\gamma_t$ for $k_0 \ne j_0$. Hence, the largest eigenvalue of G is bounded by

$$\lambda_{\max}(G) \;\le\; \|G\|_\infty \;\le\; \varepsilon^2\,(1 + \gamma_t\,\ell).$$

Concerning H, we note that $H_{j_0j_0} = 1$ for all j_0, whereas, when $k > j$,

$$(\mathbf{r}_k,\, B^{-1}\,\mathbf{r}_j) \;=\; (\mathbf{r}_k,\, \mathcal{B}(\mathbf{r}_j)\,) + (\mathbf{r}_k,\, \mathbf{t}_j) \;=\; (\mathbf{r}_k,\, \mathbf{t}_j)$$

because $\mathcal{B}(\mathbf{r}_j)$ is a linear combination of $\mathbf{d}_0$, $\mathbf{d}_1$, …, $\mathbf{d}_j$, and (2.2) applies therefore to show $(\mathbf{r}_k,\, \mathcal{B}(\mathbf{r}_j)\,) = 0$.[2] One then readily obtains with (4.5) that $|H_{k_0j_0}| = |H_{j_0k_0}| \le \varepsilon\,\gamma_r$ for $k_0 > j_0$. The Gerschgorin theorem then yields

$$\lambda_{\min}(H) \;\ge\; 1 - \varepsilon\,\gamma_r\,\ell.$$

$\frac{y^T\,G\,y}{y^T\,H\,y} \le \widehat{\varepsilon}^{\,2}$ and therefore the required result readily follows. ∎

Note that, with (2.3), Theorem 4.1 yields

$$\frac{\|\mathbf{u} - \mathbf{u}_{i+1}\|_A}{\|\mathbf{u} - \mathbf{u}_{i-\ell}\|_A} \;\le\; \min_{\substack{\mathcal{P}_{\ell+1}\text{pol. of degree }(\ell+1)\\ \mathcal{P}_{\ell+1}(0)=1}} \left\|\mathcal{P}_{\ell+1}\left(A^{1/2}\,\widehat{B}_{i,\ell}\,A^{1/2}\right)\right\|. \tag{4.15}$$

A bound on the convergence rate can then be deduced by selecting shifted Chebyshev polynomials for which, thanks to (4.8), we may apply Eiermann analysis based on the field of value [14, Theorem 1]. Clearly, the resulting bound on the asymptotic convergence factor will tend to the usual bound $\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}$ as $W(A^{1/2}\widehat{B}_{i,\ell}A^{1/2})$ gets closer to $[\lambda_1,\, \lambda_n]$, that is, for ε "sufficiently small."

We did not pursue this direction, however, because the requirements on ε would be too stringent and no longer compatible with our general goal of allowing loose accuracy for the inner solutions. Indeed, in theory, we can only say about γ_t and γ_r that they do not exceed 1 by virtue of the Cauchy–Schwarz inequality (with respect to the $(\cdot,\, B\,\cdot)$ inner product). Hence, for realistic ε, the theoretical bound on $\widehat{\varepsilon}$ grows quickly with the number of iterations, even if in practice γ_t and γ_r are both expected to be small, because we act in a very large space and the error vectors $\mathbf{t}_j$ have at first sight no serious reason to be aligned in one preferred direction.

[2] This is the technical detail that motivates the restriction to the untruncated version of Algorithm 2.1.

Theorem 4.1 is also too pessimistic when it requires no truncation, whereas FCG(1) or even standard CG is observed to be quite robust in practice; see, e.g., [17, 18, 20]. This is not completely surprising if one takes into account the general analysis of perturbed CG sequences [22, 23, 31, 26], which indeed display that the *linear* convergence properties of the method are essentially preserved, at least when such perturbations are tiny. Extrapolating these conclusions to larger perturbations gives then a heuristic explanation of the observed robustness.

Now, fast convergence is often possible only on account of the so-called *superlinear* convergence of CG. This arises when the condition number is relatively large, but with a good separation of the extremal eigenvalues, so that interesting bounds on the convergence can nevertheless be proved by selecting polynomials that vanish exactly at these isolated eigenvalues [6, 32]. Since (4.9) and (4.11) show that the spectrum of $\widehat{B}_{i,\ell}A$ is only a slight perturbation of that of $B^{-1}A$,[3] we expect a similar superlinear convergence when applying untruncated FCG, even if, here again, trying to prove this rigorously would lead to too stringent requirements on ε.

Concerning truncated FCG, however, we cannot put forward in this context the heuristic argument above based on the general results about perturbed CG sequences. Indeed, detailed analysis reveals that these superlinear bounds are then essentially unreliable, especially when the perturbations are not that small [26]. Hence, we rather expect a big difference between truncated and untruncated versions when the extremal eigenvalues are well separated.

5. Numerical results. Theorem 3.1 delivers a bound that, at first order in ε, corresponds to the usual bound for the steepest descent in which the condition number is magnified by a factor $\frac{1+\varepsilon}{1-\varepsilon}$. As this does not take into account global convergence effects, this is clearly much too pessimistic for (sufficiently) small ε when using FCG(m) with $m > 0$. On the other hand, attempts to improve the analysis as in Theorem 4.1 or in [20] lead to too stringent requirements on ε.

We therefore resorted to numerical tests to further assess the real convergence rate. In section 5.1, we conducted very simple experiments with artificial examples to investigate to what extent the number of iterations can be realistically estimated by making the substitution $\kappa \to \kappa\frac{1+\varepsilon}{1-\varepsilon}$ in the standard bounds for the CG method with a fixed preconditioner. On the other hand, in sections 5.2 and 5.3, we considered more realistic computations, aiming principally at comparing FCG with standard CG while assessing the real effects of truncation.

5.1. Some artificial experiments. Here, we let $A = \mathrm{diag}(\lambda_i)$ with

Case 1: $\lambda_i = 1 + \kappa\,\frac{i-1}{n-1}$, $\kappa = 5$, $i = 1, \ldots, n$;
Case 2: $\lambda_i = 1 + \kappa\,\frac{i-1}{n-1}$, $\kappa = 50$, $i = 1, \ldots, n$;
Case 3: $\lambda_1 = 10^{-2}$, $\lambda_i = 1 + \kappa_2\,\frac{i-2}{n-2}$, $\kappa_2 = 10$, $i = 2, \ldots, n$.

We further set $B = I$ and consider two kinds of perturbations for the "solution" of $I\,\mathbf{w}_i = \mathbf{r}_i$:

(a) $\mathbf{w}_i = \mathbf{r}_i + \varepsilon\,\frac{\|\mathbf{r}_i\|}{\|\mathbf{f}\|}\,\mathbf{f}$, the components of $\mathbf{f}$ being pseudorandom numbers uniformly distributed in $[-1, 1]$;
(b) $\mathbf{w}_i$ computed from inner CG iterations with the zero vector as initial approximation, $B = I$ as system matrix, and ε as prescribed accuracy on the relative residual error (which is also equal to the relative error measured in

[3] The eigenvalues of $\widehat{B}_{i,\ell}A$ satisfy $\lambda_k(\varepsilon) = \frac{\left(\mathbf{z}_k,\, A^{1/2}\,\widehat{B}_{i,\ell}\,A^{1/2}\,\mathbf{z}_k\right)}{(\mathbf{z}_k,\,\mathbf{z}_k)} + \mathcal{O}\left(\varepsilon^2\right)$; see [35, p. 69].

TABLE 1
Number of iterations for the artificial examples; FCG(1) is used for Cases 1 and 2 and FCG(∞) is used for Case 3.

ε	0	10^{-2}	10^{-1}	$\frac{1}{7}$	$\frac{1}{4}$	$\frac{1}{3}$	$\frac{1}{2}$	1
Case 1:	$\lambda_i = 1 + \kappa\,(i-1)/(n-1)$, $\kappa = 5$							
Estimate (5.1)	17	17	18	19	21	23	29	
Random pert.	15	15	16	17	19	22	28	
Inner PCG		15	16	17	19	21	24	
(# inner it.)		(117)	(64)	(66)	(56)	(52)	(47)	(49)
Case 2:	$\lambda_i = 1 + \kappa\,(i-1)/(n-1)$, $\kappa = 50$							
Estimate (5.1)	52	52	57	60	67	73	89	
Random pert.	49	49	55	59	69	81	116	
Inner PCG		50	54	56	64	75	71	
(# inner it.)		(397)	(216)	(222)	(191)	(153)	(141)	(155)
Case 3:	$\lambda_1 = 10^{-2}$, $\lambda_i = 1 + \kappa_2\,(i-2)/(n-2)$, $\kappa_2 = 10$, $i > 1$							
Estimate (5.2)	35	35	38	39	45	48	59	
Random pert.	31	31	32	33	37	40	49	
Inner PCG		31	33	33	40	41	42	
(# inner it.)		(246)	(132)	(130)	(119)	(90)	(83)	(99)

the B-norm); since $B = I$, this inner CG process is nontrivial only if one uses a "preconditioner" C such that $C^{-1}B \neq I$, and we actually selected C defined by $C = \mathrm{diag}(c_i)$ with $c_i^{-1} = 1 + 10\,\frac{i-1}{n-1}$ (hence $\kappa(C^{-1}B) = 10$).

For Cases 1 and 2, the heuristic estimate of the number of iterations is

$$k_\delta \;=\; \mathrm{int}\left[\frac{1}{2}\,\sqrt{\kappa\tfrac{1+\varepsilon}{1-\varepsilon}}\,\ln\frac{2}{\delta}\right] + 1, \tag{5.1}$$

where δ is the prescribed accuracy on the relative error measured in the A-norm. For Case 3, we have to take into account that the first eigenvalue is isolated. A heuristic estimate is then obtained by multiplying κ_2 by $\frac{1+\varepsilon}{1-\varepsilon}$ in the appropriate bound, which gives [6, 26]

$$k_\delta \;=\; \mathrm{int}\left[\frac{1}{2}\sqrt{\kappa_2\tfrac{1+\varepsilon}{1-\varepsilon}}\left(\ln\frac{2}{\delta} + \ln\frac{\lambda_2}{\lambda_1}\right)\right] \;+\; \mathrm{int}\left[\sqrt{\kappa_2\tfrac{1+\varepsilon}{1-\varepsilon}} + 1\right] \;+\; 1. \tag{5.2}$$

We made tests with both truncated and untruncated FCG, using in each case $n = 10^4$, $\delta = 10^{-6}$, a pseudorandom right-hand side, and the zero vector as initial approximation. It turned out that, for both Cases 1 and 2, there was almost no difference between FCG(1) and the untruncated version, so we report in Table 1 the results for FCG(1) only. On the other hand, for Case 3, truncated versions converged much more slowly, so we report the results for the untruncated version only; in other cases the comparison with the heuristic estimate (5.2) is meaningless anyway (truncated and untruncated versions are compared further in section 5.3). Besides, we also report the total number of inner iterations for the choice (b), where $\mathbf{w}_i$ is generated by performing inner CG iterations; the number given for $\varepsilon = 1$ is then the number of CG iterations when A is directly preconditioned by C.

One sees that the heuristic estimates predict relatively well the actual convergence, although they cannot pretend in any way to be *upper bounds* on the number of

iterations. In concrete applications, the optimal ε will clearly depend on the overhead involved by outer iterations, and it is therefore difficult to define a general rule. The sensitivity to variations in the preconditioner is somewhat higher when the condition number is relatively large, but such effects seem very slight, at least on these examples. We therefore do not necessarily confirm that ε should be kept proportional to $\kappa^{-1/2}$ as advised in [20].

5.2. A problem with no separated eigenvalues. We consider the linear systems resulting from the finite element discretization of the linear elasticity equations. The aim is to compute the displacement at each gridpoint. Thus, in three dimensions, there are three unknowns per gridpoint, one for each component. If these unknowns are ordered separately, the system matrix presents the 3×3 block structure

$$A = \begin{pmatrix} A_{xx} & A_{xy} & A_{xz} \\ A_{yx} & A_{yy} & A_{yz} \\ A_{zz} & A_{zy} & A_{zz} \end{pmatrix}.$$

A is then known to be spectrally equivalent to its block diagonal part [5, 10]; thus we select

$$B = \begin{pmatrix} A_{xx} & & \\ & A_{yy} & \\ & & A_{zz} \end{pmatrix}.$$

B is itself easily preconditioned, because each diagonal block is similar to a weakly anisotropic discrete Laplace operator. Using more particularly the finite element scheme described in [10], these blocks are even symmetric M-matrices, allowing an appropriate use of incomplete factorization methods (e.g., [4, 13]). This preconditioner of B can serve as is to precondition A [10, 30], but there are some (potential) advantages in using an inner–outer iterative scheme [5], because A is much denser than B; hence it is worth trying to keep the number of multiplications by A to a minimum.

For the numerical test, we consider more particularly the mining stability problem referred to as "Stope" in [11], which is discretized on a $40 \times 40 \times 40$ regular grid by means of the finite element scheme described in [10]. The numerically computed smallest eigenvalues of $B^{-1}A$ are $0.301, 0.326, 0.369, \ldots$, and the largest ones (in decreasing order) are $1.98, 1.91, 1.79, \ldots$. Thus, $\kappa = 6.58$ and there are no isolated eigenvalues.

We report in Table 2 the number of (outer) iterations necessary to reduce the relative residual error below 10^{-6} whenever using the zero vector as an initial approximation; ε is the tolerance used for the inner iterations, the stopping test being also based on the relative residual error and checked independently for each of the three systems $A_{xx}\,\mathbf{w}_x = \mathbf{r}_x$, $A_{yy}\,\mathbf{w}_y = \mathbf{r}_y$, $A_{zz}\,\mathbf{w}_z = \mathbf{r}_z$ (note for completeness that these diagonal blocks were preconditioned by the dynamic relaxed block ILU algorithm from [25]); we also indicate the values taken by the heuristic estimate (5.1).

One may observe a nice agreement between these numerical results and our previous conclusions: As long as the inner solutions remain reasonably accurate, the standard CG algorithm and FCG($m_{\max}$) behave similarly and deliver a convergence close to that of the method with exact preconditioning. Note, however, that for very inaccurate inner solutions, FCG tends to be more stable than CG, and it seems to still obey the rule that the condition number is just multiplied by $\frac{1+\varepsilon}{1-\varepsilon}$. As this stabilization is already obtained for FCG(1), we recommend its use for problems with

TABLE 2
Results for the problem with no separated eigenvalues.

ε	10^{-6}	10^{-3}	10^{-2}	10^{-1}	$\frac{1}{7}$	$\frac{1}{4}$	$\frac{1}{3}$	$\frac{1}{2}$
CG	17	17	17	18	19	29	34	68
FCG(1)	17	17	17	18	18	19	22	31
FCG(15)	17	17	17	18	18	20	21	31
Estimate (5.1)	19	19	19	21	22	24	26	33

this type of eigenvalue distribution. Practitioners may be satisfied with the standard CG algorithm, but for a fairly small overhead (one more inner product computation per iteration) FCG(1) allows the inner solutions to be still less accurate and therefore cheaper.

5.3. A problem with well-separated extremal eigenvalues. To generate a simple example in which $B^{-1}A$ has isolated eigenvalues, we apply a very basic additive Schwarz preconditioning scheme (e.g., see [12] and the references therein) to a problem from [33], namely, the linear system resulting from the five-point approximation of

$$-\partial_x\, a\, \partial_x\, u - \partial_y\, a\, \partial_y\, u = f$$

in the unit square, with homogeneous Dirichlet boundary conditions on the bottom boundary and homogeneous Neumann boundary conditions elsewhere, and with $a = f = 100$ in the box $\left(\frac{1}{4}, \frac{1}{4}\right) \times \left(\frac{1}{4}, \frac{1}{4}\right)$ and $a = 1$, $f = 0$ elsewhere; we use a uniform grid of mesh size $h = \frac{1}{160}$.

The partitioning is based on a division of the domain (that is, the unit square) in 4×2 subdomains with internal boundaries located at $x = \frac{1}{4}$, $x = \frac{1}{2}$, $x = \frac{3}{4}$, and $y = \frac{1}{2}$. Since we want to simulate numerical difficulties that may be unavoidable in other contexts, we deliberately omit the coarse grid correction and consider minimal overlap; that is, the unknowns for each subdomain are just those corresponding to the gridpoints effectively located in the subdomain or on its boundary.

Letting B be the preconditioner corresponding to exact subdomain solves, we have that the numerically computed smallest eigenvalues of $B^{-1}A$ are $0.207\,10^{-3}$, 0.0486, 0.698, ..., and the largest ones (in decreasing order) are 4.00, 2.08, 2.00, ... Hence, $\kappa = 1.9\,10^4$, which is much too large to obtain a satisfactory convergence without the superlinear convergence effects mentioned at the end of section 4.

We report in Table 3 the number of (outer) iterations necessary to reduce the relative residual error below 10^{-6} whenever we use the zero vector as an initial approximation; ε is again the tolerance for the inner subdomain solves, also verified with respect to the relative residual error; for these solves, we used the standard MILU preconditioning without fill-in [4, 13].

The results for both FCG(1) and the untruncated version (FCG(∞)) just confirm our expectation from the previous discussions. The good news is that satisfactory convergence for fairly moderate accuracy of the inner solutions is already restored with reasonable values of $m_{\max}$. We do not discuss which couple (ε, $m_{\max}$) is most effective because this heavily depends on the context, for instance on the quality of the preconditioner for the inner iterations. Note, however, that we programmed the orthogonalization of $\mathbf{d}_i$ in a classical Gram–Schmidt fashion, so that the communication cost per iteration is essentially independent of $m_{\max}$; the tradeoff between increasing $m_{\max}$ or decreasing ε to achieve a given convergence can thus be analyzed by

TABLE 3
Numbers of iteration for the problem with small isolated eigenvalues; Tr–FCG ($m_{\max}$) refers to Algorithm 2.1 with $m_i = \min(i, m_{\max})$.

ε	10^{-6}	10^{-3}	10^{-2}	10^{-1}	$\frac{1}{7}$	$\frac{1}{4}$	$\frac{1}{3}$	$\frac{1}{2}$
FCG(1)	76	86	115	250	400	771	>999	>999
FCG(5)	80	86	117	158	192	167	201	230
FCG(10)	61	86	88	92	125	156	131	142
FCG(20)	62	62	83	86	90	94	97	129
FCG(30)	59	60	63	67	68	72	94	105
FCG(45)	60	61	63	63	64	66	69	77
FCG(∞)	58	59	60	62	62	64	66	71
Tr–FCG(10)	60	84	85	114	125	124	160	144
Tr–FCG(20)	62	80	83	87	90	94	95	105

comparing the cost of the inner solves with that of local vector operations. Which convergence one should target is difficult to predict, but, as a general rule, it seems reasonable to admit only a moderate deterioration of the convergence from the "ideal" case with exact preconditioning. Indeed, inner–outer iterative processes are primarily motivated by the high cost of outer iterations, for instance, the communication cost in the present example. (Would it not be the case, one probably should skip inner iterations and use one application of the corresponding preconditioner instead.)

For the sake of completeness, we also performed some runs using Algorithm 2.1 with a pure truncation strategy (Tr–FCG($m_{\max}$)). Clearly, the additional work with respect to FCG($m_{\max}$) does not pay off since the convergence is nearly the same in all cases, whereas FCG($2\,m_{\max}$) is much faster despite a similar cost per iteration. We also tested a pure restart approach, but we did not succeed in obtaining a useful convergence. This is not surprising since, even with a fixed preconditioner, pure restarting has a disastrous effect with this kind of eigenvalue distribution.

6. Conclusions. When one is willing to use inexact preconditioning, one should distinguish two types of problems.

For problems with a regular distribution of the eigenvalues, for which the fast convergence is due to a nice condition number, inexact preconditioning may be used quite safely, even with very loose stopping criteria for the inner iterations. It is not necessary to use a method much more costly than the standard CG algorithm, but FCG(1) brings still more stability at the price of a fairly small overhead.

The situation is more tricky when a relatively fast convergence depends on "superlinear" effects in connection with a good separation of the extremal eigenvalues. A convergence close to that of the ideal case with exact preconditioning can still be obtained with FCG($m_{\max}$), but at the price of increasing $m_{\max}$ while decreasing the accuracy requirements for the inner solutions. Hence, one will not be able to use stopping criteria as loosely as in the preceding case, and the FCG process itself will be more costly.

These considerations also tell us that whenever using inner–outer iterative schemes, improving the basic preconditioner always brings some additional benefit. For instance, removing small isolated eigenvalues not only accelerates the convergence with exact preconditioning, but also indirectly allows a decrease in the cost of inner

solutions.

Generally speaking, we feel that coupled inner–outer iterations can be efficient when at least either the global preconditioner or the one used for the inner solutions is a highly effective. Indeed, in the former case, the CG process will be very stable, allowing quite loose stopping criteria for the inner iterations. On the other hand, in the latter case, it is relatively cheap to prevent stability problems by requiring a high accuracy for the inner solutions. These considerations also show that when both preconditioners are of medium quality, then coupled inner–outer iterations are not necessarily advisable.

Acknowledgments. My interest in variable preconditioning originates from private discussions with O. Axelsson and his coworkers M. Neytcheva and B. Polman. I also thank R. Blaheta and R. Kohut who provided the code that generates the matrix for the first test example. Mark Embree provided me with some interesting remarks, pointing out the possible use of Eiermann results. Magolu Monga-Made informed me of reference [20] as soon as it appeared.

REFERENCES

[1] O. Axelsson, *Conjugate gradient type methods for unsymmetric and inconsistent systems of linear equations*, Linear Algebra Appl., 29 (1980), pp. 1–16.

[2] O. Axelsson, *A generalized conjugate gradient, least square method*, Numer. Math., 51 (1987), pp. 209–227.

[3] O. Axelsson, *A restarted version of a generalized preconditioned conjugate gradient method*, Comm. Appl. Numer. Methods, 4 (1988), pp. 521–530.

[4] O. Axelsson, *Iterative Solution Methods*, Cambridge University Press, Cambridge, UK, 1994.

[5] O. Axelsson, *On Iterative Solvers in Structural Mechanics; Separate Displacement Orderings*, Tech. report 9721, Department of Mathematics, Catholic University, Nijmegen, The Netherlands, 1997.

[6] O. Axelsson and G. Lindskog, *On the rate of convergence of the preconditioned conjugate gradient method*, Numer. Math., 48 (1986), pp. 499–523.

[7] O. Axelsson and P. S. Vassilevski, *A black box generalized conjugate gradient solver with inner iterations and variable-step preconditioning*, SIAM J. Matrix Anal. Appl., 12 (1991), pp. 625–644.

[8] O. Axelsson and P. S. Vassilevski, *Variable-step multilevel preconditioning methods.* I. *Self-adjoint and positive definite elliptic problems*, Numer. Linear Algebra Appl., 1 (1994), pp. 75–101.

[9] R. Barrett, M. Berry, T. F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. van der Vorst, *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*, SIAM, Philadelphia, 1994.

[10] R. Blaheta, *Displacement decomposition—incomplete factorization preconditioning techniques for linear elasticity problems*, Numer. Linear Algebra Appl., 1 (1994), pp. 107–126.

[11] R. Blaheta, O. Jakl, and J. Starý, *A parallel cg solver for FE analysis of* 3D *problems in geomechanics*, in Geomechanics 96, Z. Rakowski, ed., Balkema, Rotterdam, 1997, pp. 159–163.

[12] T. Chan and T. Mathew, *Domain decomposition algorithms*, Acta Numer., 3 (1994), pp. 61–143.

[13] T. Chan and H. van der Vorst, *Approximate and incomplete factorizations*, in Parallel Numerical Algorithms, D. E. Keyes, A. Samed, and V. Venkatakrishnan, eds., ICASE/LaRC Interdisciplinary Series in Science and Engineering, Vol. 4, Kluwer Academic, Dordecht, 1997, pp. 167–202.

[14] M. Eiermann, *Fields of values and iterative methods*, Linear Algebra Appl., 180 (1993), pp. 167–197.

[15] H. C. Elman and G. H. Golub, *Inexact and preconditioned Uzawa algorithms for saddle point problems*, SIAM J. Numer. Anal., 31 (1994), pp. 1645–1661.

[16] E. Giladi, G. H. Golub, and J. B. Keller, *Inner and outer iterations for the Chebyshev algorithm*, SIAM J. Numer. Anal., 35 (1998), pp. 300–319.

[17] G. H. Golub and M. Overton, *Convergence of two-stage Richardson iterative procedure for*

solving systems of linear equations, in Numerical Analysis, G. Watson, ed., Lectures Notes in Math. 912, Springer-Verlag, Berlin Heidelberg, New York, 1983, pp. 128–139.

[18] G. H. Golub and M. L. Overton, *The convergence of inexact Chebyshev and Richardson iterative methods for solving linear systems*, Numer. Math., 53 (1988), pp. 571–593.

[19] G. H. Golub and C. F. van Loan, *Matrix Computations*, 3rd ed., The John Hopkins University Press, Baltimore, MD, 1996.

[20] G. H. Golub and Q. Ye, *Inexact preconditioned conjugate gradient method with inner-outer iterations*, SIAM J. Sci. Comput., 21 (1999), pp. 1305–1320.

[21] A. Greenbaum, *Iterative Methods for Solving Linear Systems*, SIAM, Philadelphia, 1977.

[22] A. Greenbaum, *Behavior of slightly perturbed Lanczos and conjugate-gradient recurrences*, Linear Algebra Appl., 113 (1989), pp. 7–63.

[23] A. Greenbaum and Z. Strakos, *Predicting the behavior of finite precision Lanczos and conjugate gradient computations*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 121–137.

[24] M. R. Hestenes and E. Stiefel, *Methods of conjugate gradients for solving linear systems*, J. Res. Nat. Bur. Standards, 49 (1952), pp. 409–436.

[25] M. M. Monga-Made and Y. Notay, *Dynamically relaxed block incomplete factorizations for solving two- and three-dimensional problems*, SIAM J. Sci. Comput., 21 (2000), pp. 2008–2028.

[26] Y. Notay, *On the convergence rate of the conjugate gradients in presence of rounding errors*, Numer. Math., 65 (1993), pp. 301–317.

[27] J. K. Reid, *On the method of conjugate gradients for the solution of large sparse systems of linear equations*, in Large Sparse Sets of Linear Equations, J. Reid, ed., Academic Press, London, 1971, pp. 231–254.

[28] Y. Saad, *A flexible inner-outer preconditioned GMRES algorithm*, SIAM J. Sci. Comput., 14 (1993), pp. 461–469.

[29] Y. Saad, *Iterative Methods for Sparse Linear Systems*, PWS Publishing, New York, 1996.

[30] P. Saint-Georges, G. Warzee, R. Beauwens, and Y. Notay, *High performance PCG solvers for FEM structural analyses*, Internat. J. Numer. Methods Engrng., 39 (1996), pp. 1313–1340.

[31] Z. Strakos, *On the real convergence rate of the conjugate gradient method*, Linear Algebra Appl., 154/156 (1991), pp. 535–549.

[32] A. van der Sluis and H. A. van der Vorst, *The rate of convergence of conjugate gradients*, Numer. Math., 48 (1986), pp. 543–560.

[33] H. A. van der Vorst, *The convergence behaviour of preconditioned CG and CG-S*, in Preconditioned Conjugate Gradient Methods, O. Axelsson and L. Kolotilina, eds., Lectures Notes in Math. 1457, Springer-Verlag, New York, 1990, pp. 126–136.

[34] H. A. van der Vorst and C. Vuik, *GMRESR: A family of nested GMRES methods*, Numer. Linear Algebra Appl., 1 (1994), pp. 369–386.

[35] J. H. Wilkinson, *The Algebraic Eigenvalue Problem*, Clarendon Press, Oxford, 1965.

SIAM J. SCI. COMPUT.
Vol. 22, No. 4, pp. 1461–1488
© 2000 Society for Industrial and Applied Mathematics

A THIRD-ORDER SEMIDISCRETE CENTRAL SCHEME FOR CONSERVATION LAWS AND CONVECTION-DIFFUSION EQUATIONS*

ALEXANDER KURGANOV† AND DORON LEVY‡

Abstract. We present a new third-order, semidiscrete, central method for approximating solutions to multidimensional systems of hyperbolic conservation laws, convection-diffusion equations, and related problems. Our method is a high-order extension of the recently proposed second-order, semidiscrete method in [A. Kurgonov and E. Tadmor, *J. Comput Phys.*, 160 (2000) pp. 241–282].

The method is derived independently of the specific piecewise polynomial reconstruction which is based on the previously computed cell-averages. We demonstrate our results by focusing on the new third-order central weighted essentially nonoscillatory (CWENO) reconstruction presented in [D. Levy, G. Puppo, and G. Russo, *SIAM J. Sci. Comput.*, 21 (1999), pp. 294–322]. The numerical results we present show the desired accuracy, high resolution, and robustness of our method.

Key words. hyperbolic systems, convection-diffusion equations, central difference schemes, high-order accuracy, nonoscillatory schemes, weighted essentially nonoscillatory reconstruction.

AMS subject classifications. Primary 65M10; Secondary 65M05

PII. S1064827599360236

1. Introduction.

Numerical methods for approximating solutions of hyperbolic conservation laws,

$$\frac{\partial}{\partial t}u(x,t) + \frac{\partial}{\partial x}f(u(x,t)) = 0, \tag{1.1}$$

and of the related convection-diffusion equations,

$$\frac{\partial}{\partial t}u(x,t) + \frac{\partial}{\partial x}f(u(x,t)) = \frac{\partial}{\partial x}Q[u(x,t), u_x(x,t)], \tag{1.2}$$

have attracted much attention in recent years (see, e.g., [6, 35] and the references therein). Here, $u(x,t)$ is a conserved quantity, $f(u)$ is a nonlinear convection flux, and $Q(u,u_x)$ is a dissipation flux satisfying the (weak) parabolicity condition, $\frac{\partial}{\partial s}Q(u,s) \geq 0\ \forall u, s$. In the most general case $u = (u_1, \ldots, u_n)$ is an n-vector in a $d+1$-dimensional space, $(x,t) = (x_1, \ldots, x_d, t)$, and f and Q are vector-functions.

In this paper, we focus on the class of *central schemes*, all of which can be viewed as an extension of the well-known Lax–Friedrichs (LxF) scheme [5]. The first-order LxF method enjoys the major advantage of simplicity over the upwind schemes (e.g., the Godunov scheme [7]): no (approximate) Riemann solvers or characteristic decompositions are involved in its construction, and therefore, its realization for complicated

*Received by the editors June 15, 1999; accepted for publication (in revised form) November 15, 1999; published electronically November 2, 2000.

http://www.siam.org/journals/sisc/22-4/36023.html

†Department of Mathematics, University of Michigan, Ann Arbor, MI 48109 (kurganov@math.lsa.umich.edu). The work of this author was supported in part by an NSF Group Infrastructure grant. Part of this work was done while the author was visiting the Lawrence Berkeley National Lab.

‡Department of Mathematics, University of California, Berkeley, CA 94720, and Lawrence Berkeley National Lab, Berkeley, CA 94720 (dlevy@math.berkeley.edu). The work of this author was supported in part by the Applied Mathematical Sciences subprogram of the Office of Science, U.S. Department of Energy, under contract DE–AC03–76–SF00098.

multidimensional systems is rather simple. At the same time, the LxF scheme suffers from excessive numerical dissipation, which causes a poor (smeared) resolution of discontinuities and rarefaction waves.

A second-order, nonoscillatory central scheme was first introduced by Nessyahu and Tadmor in [30]. Since then, the Nessyahu–Tadmor (NT) scheme was further extended to higher orders of accuracy [28] (also see [3, 19]), as well as to the multidimensional systems (1.1), in [1] and [12] (also see [18, 20, 21, 22]).

The main ingredient in the construction of the NT method is a second-order, nonoscillatory, monotonic upstream schemes for conservation laws (MUSCL)-type [17], piecewise linear interpolant (instead of the piecewise constant one employed in the LxF scheme) in combination with the exact solver for the time evolution. This approach allows us to significantly improve the resolution of nonsmooth solutions to hyperbolic conservation laws, (1.1), while retaining the main advantage of the LxF scheme—*simplicity*.

Unfortunately, applying the fully discrete NT scheme (or its higher-order extensions) to the second-order convection-diffusion equations, (1.2), does not provide the desired resolution of discontinuities (see, e.g., [14, 15, 16]). This loss of resolution occurs due to the accumulation of excessive numerical dissipation, which is typical of fully discrete central schemes with small time-steps $\Delta t \sim (\Delta x)^2$ (see [16] for details).

To circumvent this difficulty, a second-order *semidiscrete* central scheme was introduced by Kurganov and Tadmor in [16]. This scheme has smaller dissipation than the NT scheme and, unlike the fully discrete central schemes, it can be efficiently used with time-steps as small as required by the CFL stability restriction due to the diffusion term.

The basic idea in the construction of the second-order semidiscrete scheme was to use more accurate information about the *local* speed of propagation of the discontinuities. One was then able to derive a nonstaggered semidiscrete central method, by first integrating over nonequally spaced control volumes, out of which a new piecewise linear interpolant was reconstructed and finally projected on its cell-averages (without evolving in time). The final step was first introduced in [10], in a somewhat different context of transforming staggered methods into nonstaggered methods.

In this paper we extend the results of [16] by introducing a new *third-order, semidiscrete, central scheme*. Our new scheme is derived in a general form which is independent of the reconstruction step, as long as the reconstructed interpolant is sufficiently accurate and nonoscillatory. In particular, we use the new third-order central weighted essentially nonoscillatory (CWENO) reconstruction proposed in [21]. This reconstruction provides a third-order accurate interpolant which is built from the given cell-averages such that it is nonoscillatory in the essentially nonoscillatory (ENO) sense (see [9, 32]). This interpolant is written as a convex combination of two one-sided linear functions and one centered parabola. In smooth regions this convex combination guarantees the desired third-order accuracy. It automatically switches to a second-order, one-sided, linear reconstruction in the presence of large gradients. Such weighted essentially non-oscillatory (WENO) reconstructions were first introduced in the upwind framework, [11, 27], after which they were extended to the central framework [19, 20, 21, 22]. A different approach to creating a high-order (ENO-type) method which does not require Riemann solvers was suggested in [26].

This paper is organized as follows. We start in section 2 with a brief overview of central schemes for conservation laws. In particular we focus in section 2.1 on the CWENO reconstruction which we use as the building block for our third-order

method below.

We then proceed to construct our new third-order scheme. First, we deal with the fully discrete, one-dimensional (1D) setup in section 3. This new fully discrete scheme is sketched in (3.7). We give only the required details that are necessary to fulfill our final goal, namely, to derive the semidiscrete scheme.

With the fully discrete scheme, (3.7), we are ready to approach the semidiscrete limit in section 4.1. Our new third-order, one-dimensional, semidiscrete scheme is then summarized in (4.5). This scheme is written in a general form which is independent of the reconstruction step and can also be combined with any appropriate ODE solver for carrying out the time evolution. In section 4.2 we then extend our semidiscrete scheme to multidimensional hyperbolic and (degenerate) parabolic problems.

We end by presenting several numerical examples in section 5, in which we approximate solutions to hyperbolic conservation laws as well as to convection-diffusion equations. Our new method is shown to enjoy the expected high accuracy as well as the robustness and the simplicity of the entire family of central schemes.

2. Central schemes for conservation laws. We briefly overview the framework of central schemes for conservation laws. Consider the 1D system (1.1). To approximate its solutions, we introduce a spatial scale, Δx, and integrate over the cell $I(x) := \{\xi \mid |\xi - x| \le \Delta x/2\}$,

$$\bar{u}_t + \frac{1}{\Delta x}\left[f\left(\left(x+\frac{\Delta x}{2},t\right)\right) + f\left(u\left(x-\frac{\Delta x}{2},t\right)\right)\right] = 0. \tag{2.1}$$

Here and below, $\bar{u}$ denotes the average of u over I,

$$\bar{u}(x,t) := \frac{1}{\Delta x}\int_{I(x)} u(\xi,t)d\xi.$$

Introducing a time scale, Δt, integrating in time from t to $t+\Delta t$, and sampling (2.1) at the cells $[x_j, x_{j+1}]$, we obtain

$$\bar{u}^{n+1}_{j+1/2} = \bar{u}^{n}_{j+1/2} - \frac{1}{\Delta x}\int_{\tau=t^n}^{t^{n+1}} \left[f(u(x_{j+1},\tau)) - f(u(x_j,\tau))\right] d\tau, \tag{2.2}$$

where $x_j := j\Delta x$, $t^n := n\Delta t$, $u^n_j := u(x_j,t^n)$, and $\bar{u}^n_j := \bar{u}(x_j,t^n)$. Assuming that at time $t = t^n$ we have computed the cell-averages of the approximate solution, $\{\bar{u}^n_j\}$, we would like to utilize (2.2) to compute the cell-averages at the next time level, $t^{n+1} = t^n + \Delta t$. To that extent, we introduce a piecewise-polynomial reconstruction,

$$u(x,t^n) \approx \sum_j P_j(x)\chi_j(x), \tag{2.3}$$

where $\chi_j(x)$ is the characteristic function of the cell $I_j := I(x_j)$, and $P_j(x)$ is a polynomial which is reconstructed from the computed cell-averages, $\{\bar{u}^n_j\}$. The degree of the polynomial depends on the desired order of accuracy of the method. Having such an approximation to $u(x,t^n)$, (2.3), we can easily compute the right-hand side (RHS) of (2.2). The first term, $\bar{u}^n_{j+1/2}$, equals

$$\bar{u}^n_{j+1/2} = \int_{x_j}^{x_{j+1/2}} P_j(x)dx + \int_{x_{j+1/2}}^{x_{j+1}} P_{j+1}(x)dx.$$

For a sufficiently small time-step, Δt, the solution of (1.1) subject to the initial data (2.3), prescribed at time $t = t^n$, will remain smooth at some neighborhood of the grid points x_j for $t \in [t^n, t^{n+1}]$. Hence, the integrals on the RHS of (2.2) can be approximated using a sufficiently accurate quadrature, which is determined by the overall desired accuracy of the method. The values at the intermediate times which will be required in the quadrature can be predicted either by a Taylor expansion or using a Runge–Kutta method (consult [3, 19, 28, 30]).

For example, a piecewise-constant reconstruction, $P_j(x) = \bar{u}_j^n$, and a first-order quadrature,

$$\int_{t^n}^{t^{n+1}} f(u(t))dt \sim \Delta t f(\bar{u}^n),$$

will result in the staggered-LxF scheme (with $\lambda := \Delta t/\Delta x$ denoting the mesh ratio),

$$\bar{u}_{j+1/2}^{n+1} = \frac{\bar{u}_{j+1}^n + \bar{u}_j^n}{2} - \lambda(f(\bar{u}_{j+1}^n) - f(\bar{u}_j^n)).$$

A piecewise linear reconstruction, $P_j(x) = \bar{u}_j^n + (u_x)_j^n(x - x_j)$, with a second-order quadrature in time (such as the midpoint rule), results in the NT scheme. Applying nonlinear limiters on the discrete slopes, $(u_x)_j^n$, will prevent oscillations (for details, see [30]).

To obtain a third-order central scheme, one should use a third-order, piecewise parabolic reconstruction together with a more accurate quadrature in time, e.g., Simpson's quadrature rule (see [28] for details).

Remarks.

1. *Robustness.* In order to reconstruct a nonoscillatory interpolant, one typically is required to use nonlinear limiters. These limiters decrease the order of accuracy of the method at extrema and by that they play a stabilizing role (e.g., see [17, 28, 30, 35]).

2. *Numerical dissipation and time step.* When using fully discrete central schemes to approximate solutions of convection-diffusion equations, (1.2), the stability restriction enforces small time-steps, $\Delta t \sim (\Delta x)^2$. That is why the numerical dissipation is accumulated and we do not obtain high resolution of discontinuities (see [16] for details).

This problem can be avoided by using semidiscrete schemes instead of the fully-discrete schemes. Such a second-order, central, semidiscrete scheme was introduced in [16]. In this paper we develop a third-order, central, semidiscrete scheme with small numerical dissipation, which can be used efficiently with the small time-steps required due to the second-order operators.

3. *Upwind schemes.* Sampling (2.1) at the cells I_j will result in upwind schemes. Here, one remains with the discontinuities along the interfaces and is bound to solve the Riemann problems there, or at least to approximate their solutions. In the scalar, 1D case this can be easily accomplished, but the Riemann problem has no known solution in the general case of systems and/or several space dimensions.

This is the reason why central schemes can be considered as universal methods for solving hyperbolic conservation laws: Riemann solvers are not involved in their construction and, moreover, since (2.2) can be carried out componentwise, no characteristic decomposition is required.

2.1. CWENO reconstruction. The first 1D, third-order central scheme in [28] implemented the nonoscillatory piecewise parabolic reconstruction proposed by Liu

and Osher in [25]. Since then, a variety of simpler reconstructions has appeared in the literature. Among these, we would like to mention the central-ENO reconstruction in [3] and the central-WENO (CWENO) reconstruction in [19] and [21], which was extended to the two-dimensional (2D) setup in [20] and [22].

Our new third-order semidiscrete method which we develop in section 3 and section 4 below can be integrated with any third-order, nonoscillatory reconstruction. In our numerical simulations presented in section 5, we will use the method recently presented in [21], of which we will now give a brief overview.

In each cell I_j we reconstruct a quadratic polynomial as a convex combination of three polynomials $P_{\rm L}(x), P_{\rm R}(x)$, and $P_{\rm C}(x)$,

$$P_j(x) = w_{\rm L}P_{\rm L}(x) + w_{\rm R}P_{\rm R}(x) + w_{\rm C}P_{\rm C}(x), \tag{2.4}$$

with positive weights $w_i \geq 0 \forall i \in \{{\rm C}, {\rm R}, {\rm L}\}$ and $\sum_i w_i = 1$. The polynomials $P_{\rm L}(x), P_{\rm R}(x)$ correspond to left and right one-sided linear reconstructions, respectively, while $P_{\rm C}(x)$ is a parabola, centered around x_j.

The linear functions, $P_{\rm R}(x)$ and $P_{\rm L}(x)$, are uniquely determined by requiring them to conserve the one-sided cell-averages ($\bar{u}^n_j, \bar{u}^n_{j+1}$ and $\bar{u}^n_j, \bar{u}^n_{j-1}$, respectively) as

$$P_{\rm R}(x) = \bar{u}^n_j + \frac{\bar{u}^n_{j+1} - \bar{u}^n_j}{\Delta x}(x - x_j), \quad P_{\rm L}(x) = \bar{u}^n_j + \frac{\bar{u}^n_j - \bar{u}^n_{j-1}}{\Delta x}(x - x_j). \tag{2.5}$$

The centered parabola, $P_{\rm C}(x)$, is chosen so as to satisfy

$$P_{\rm EXACT}(x) = c_{\rm L}P_{\rm L}(x) + c_{\rm R}P_{\rm R}(x) + (1 - c_{\rm L} - c_{\rm R})P_{\rm C}(x) \tag{2.6}$$

with constants c_i's. Here, $P_{\rm EXACT}(x)$ is the unique parabola that conserves the three-cell-averages, $\bar{u}^n_{j-1}, \bar{u}^n_j$, and $\bar{u}^n_{j+1}$, which is given by

$$P_{\rm EXACT}(x) = u^n_j + u'_j(x - x_j) + \frac{1}{2}u''_j(x - xj)^2. \tag{2.7}$$

The approximations to the point-values of $u(x_j, t^n), u_x(x_j, t^n)$, and $u_{xx}(x_j, t^n)$ are denoted by u^n_j, u'_j, u''_j and are given by

$$u^n_j = \bar{u}^n_j - \frac{1}{24}(\bar{u}^n_{j+1} - 2\bar{u}^n_j + \bar{u}^n_{j-1}),$$
$$u'_j = \frac{\bar{u}^n_{j+1} - \bar{u}^n_{j-1}}{2\Delta x}, \qquad u''_j = \frac{\bar{u}^n_{j-1} - 2\bar{u}^n_j + \bar{u}^n_{j+1}}{\Delta x^2}.$$

In [21] it was shown that every symmetric selection of the constants c_i's in (2.6) will provide the desired third-order accuracy. For example, by taking $c_{\rm L} = c_{\rm R} = 1/4$, (2.5)–(2.7) yield

$$P_{\rm C}(x) = \bar{u}^n_j - \frac{1}{12}(\bar{u}^n_{j+1} - 2\bar{u}^n_j + \bar{u}^n_{j-1})$$
$$+\frac{\bar{u}^n_{j+1} - \bar{u}^n_{j-1}}{2\Delta x}(x - x_j) + \frac{\bar{u}^n_{j+1} - 2\bar{u}^n_j + \bar{u}^n_{j-1}}{\Delta x^2}(x - x_j)^2.$$

In smooth regions, the coefficients w_i of the convex combination in (2.4) are chosen to guarantee the maximum order of accuracy (in this particular case, order three), but in the presence of a discontinuity they are automatically switched to the best

one-sided stencil (which generates the least oscillatory reconstruction). The weights are taken as

$$w_i = \frac{\alpha_i}{\sum\limits_m \alpha_m}, \quad \text{where} \quad \alpha_i = \frac{c_i}{(\epsilon + IS_i)^p}, \qquad i, m \in \{\text{C}, \text{R}, \text{L}\}, \tag{2.8}$$
$$c_{\text{L}} = c_{\text{R}} = 1/4, \qquad c_{\text{C}} = 1/2.$$

The constant ϵ guarantees that the denominator does not vanish and is taken as $\epsilon = 10^{-6}$. The value of p may be chosen to provide the highest accuracy in smooth areas and to ensure the nonoscillatory nature of the solution near the discontinuities (consult [11]; see also [19, 21]). In [11] the value $p = 2$ was empirically selected, and here we use the same p in most of the examples presented below. Finally, the smoothness indicators, IS_i, are defined as

$$IS_i = \sum_{l=1}^{2} \int_{x_{j-1/2}}^{x_{j+1/2}} (\Delta x)^{2l-1} (P_i^{(l)}(x))^2 dx.$$

A direct computation then results in

$$IS_{\text{L}} = (\bar{u}_j^n - \bar{u}_{j-1}^n)^2, \qquad IS_{\text{R}} = (\bar{u}_{j+1}^n - \bar{u}_j^n)^2, \tag{2.9}$$
$$IS_{\text{C}} = \frac{13}{3}(\bar{u}_{j+1}^n - 2\bar{u}_j^n + \bar{u}_{j-1}^n)^2 + \frac{1}{4}(\bar{u}_{j+1}^n - \bar{u}_{j-1}^n)^2.$$

It is easy to see that in the presence of large gradients, this reconstruction switches to one of the second-order one-sided linear reconstructions, P_{R} or P_{L}. For more details we refer to [21].

3. The fully discrete one-dimensional construction. In this section we present the new third-order method in the fully discrete framework. Since we are mainly interested in deriving the semidiscrete scheme, we will concentrate only on the details which are required for that task. The scheme we derive here is a third-order extension of the fully discrete second-order scheme presented in [16].

Following [16], we would like to augment the integration over the Riemann fans by more accurate information about the *local* speed of wave propagation. We start by assuming that in every cell, I_j, we have reconstructed a piecewise polynomial interpolant, $P_j(x, t^n)$, from the previously computed cell-averages, $\{\bar{u}_j^n\}$. Then, an upper bound on the speed of propagation of discontinuities at the cell boundaries, $x_{j+1/2}$, is given by

$$a_{j+1/2}^n = \max_{u \in \mathcal{C}(u_{j+1/2}^-, u_{j+1/2}^+)} \rho\Big(\frac{\partial f}{\partial u}(u)\Big), \tag{3.1}$$

where $\rho(A)$ denotes the spectral radius of a matrix A, i.e., $\rho(A) := \max_i |\lambda_i(A)|$ with $\lambda_i(A)$ being its eigenvalues. We denote by $u_{j+1/2}^+$ and $u_{j+1/2}^-$ the left and right intermediate values of $u(x, t^n)$ at $x_{j+1/2}$, i.e.,

$$u_{j+1/2}^+ := P_{j+1}(x_{j+1/2}, t^n), \quad u_{j+1/2}^- := P_j(x_{j+1/2}, t^n),$$

and by $\mathcal{C}(u_{j+1/2}^-, u_{j+1/2}^+)$ a curve in phase space that connects $u_{j+1/2}^-$ and $u_{j+1/2}^+$ via the Riemann fan.

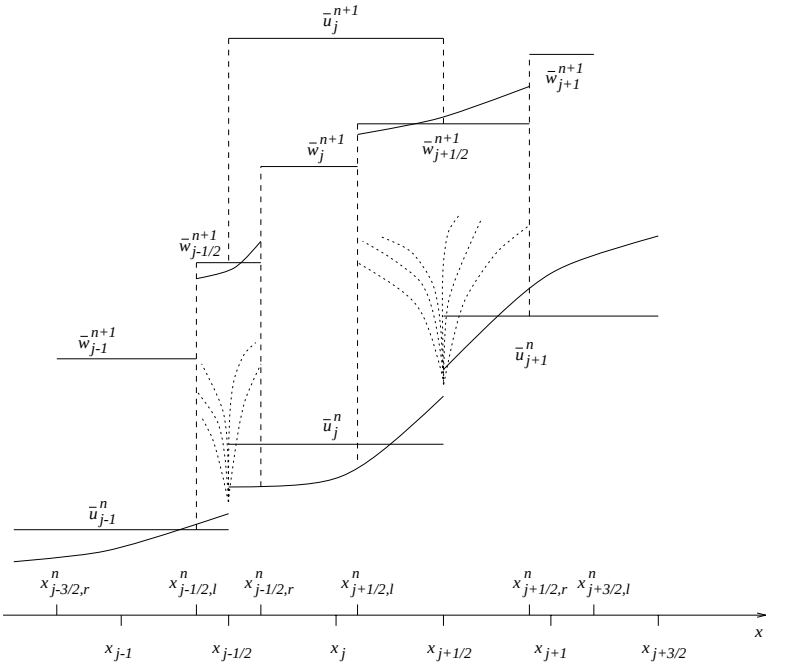


FIG. 3.1. *Modified central differencing.*

Remark. In most practical applications, these local maximal speeds can be easily evaluated. For example, in the genuinely nonlinear or linearly degenerate case one finds that (3.1) reduces to

$$a_{j+1/2}^n := \max\left\{\rho\Big(\frac{\partial f}{\partial u}(u_{j+1/2}^-)\Big), \rho\Big(\frac{\partial f}{\partial u}(u_{j+1/2}^+)\Big)\right\}. \tag{3.2}$$

Given the piecewise polynomial interpolant at time t^n, $\{P_j(x,t^n)\}$, and the local speeds of propagation, $\{a_{j+1/2}^n\}$, we construct the fully discrete, central method in two steps, which are schematically described in Figure 3.1. First, we integrate over the control volumes, $[x_{j-1/2,l}^n, x_{j-1/2,r}^n]\times[t^n,t^{n+1}]$, $[x_{j-1/2,r}^n, x_{j+1/2,l}^n]\times[t^n,t^{n+1}]$, and $[x_{j+1/2,l}^n, x_{j+1/2,r}^n]\times[t^n,t^{n+1}]$, obtaining $\bar{w}_{j-1/2}^{n+1}$, $\bar{w}_j^{n+1}$, and $\bar{w}_{j+1/2}^{n+1}$, respectively. Due to the finite speed of propagation, the points $x_{j+1/2,l}^n$ and $x_{j+1/2,r}^n$,

$$x_{j+1/2,l}^n := x_{j+1/2} - a_{j+1/2}^n\Delta t, \quad x_{j+1/2,r}^n := x_{j+1/2} + a_{j+1/2}^n\Delta t,$$

separate between smooth and nonsmooth regions. That is, the solution of (1.1) subject to the piecewise polynomial initial data prescribed at time $t = t^n$ may be nonsmooth only inside the intervals $[x_{j+1/2,l}^n\,,\, x_{j+1/2,r}^n]$ for $t\in[t^n,t^{n+1})$.

In the second step, we repeat the nonoscillatory reconstruction (this time on a nonuniformly spaced grid) and project the obtained reconstruction on the original, uniform grid, ending up with the cell-averages at the next time level t^{n+1}, $\{\bar{u}_j^{n+1}\}$. This last step does not involve time integration and was introduced in the context of changing staggered methods into nonstaggered methods in [10].

We now turn to the detailed description of this algorithm. Assume that the

piecewise polynomial reconstruction in cell I_j at time t^n is of the form

$$P_j(x,t^n) = A_j + B_j(x-x_j) + \frac{1}{2}C_j(x-x_j)^2. \tag{3.3}$$

Then a direct computation of the integrals over the control volumes, $[x^n_{j+1/2,l}, x^n_{j+1/2,r}]\times[t^n,t^{n+1}]$ and $[x^n_{j-1/2,r}, x^n_{j+1/2,l}]\times[t^n,t^{n+1}]$, yields

$$\begin{aligned}\bar{w}^{n+1}_{j+1/2} &= \frac{A_j+A_{j+1}}{2} + \frac{\Delta x - a^n_{j+1/2}\Delta t}{4}(B_j - B_{j+1})\\ &+ \left(\frac{\Delta x^2}{16} - \frac{a^n_{j+1/2}\Delta t\Delta x}{8} + \frac{(a^n_{j+1/2}\Delta t)^2}{12}\right)(C_j + C_{j+1})\\ &- \frac{1}{2a^n_{j+1/2}\Delta t}\left\{\int_{t^n}^{t^{n+1}}\Big[f(u(x^n_{j+1/2,r},t))dt - f(u(x^n_{j+1/2,l},t))\Big]\,dt\right\},\end{aligned} \tag{3.4}$$

and

$$\begin{aligned}\bar{w}^{n+1}_{j} &= A_j + \frac{\Delta t}{2}(a^n_{j-1/2} - a^n_{j+1/2})B_j\\ &+ \left[\frac{(\Delta x)^2}{24} - \frac{\Delta t\Delta x}{12}(a^n_{j-1/2}+a^n_{j+1/2})\right.\\ &\qquad\left. + \frac{(\Delta t)^2}{6}\Big((a^n_{j-1/2})^2 - a^n_{j-1/2}a^n_{j+1/2} + (a^n_{j+1/2})^2\Big)\right]C_j\\ &- \frac{1}{\Delta x - \Delta t(a^n_{j-1/2}+a^n_{j+1/2})}\left\{\int_{t^n}^{t^{n+1}}\Big[f(u(x^n_{j+1/2,l},t))dt\right.\\ &\qquad\left. - f(u(x^n_{j-1/2,r},t))\Big]\,dt\right\},\end{aligned} \tag{3.5}$$

respectively. To complete these computations, one should approximate the flux integrals on the RHS of (3.4) and (3.5) using, e.g., Simpson's quadrature as described in section 2.

At this stage, the approximate cell-averages, $\{\bar{w}^{n+1}_{j+\frac{1}{2}}, \bar{w}^{n+1}_j\}$, realize the solution at $t=t^{n+1}$ over a nonuniform grid, which is oversampled by twice the number of the original cells at $t=t^n$. To convert these nonuniform averages back into the original grid, we proceed along the lines of [10].

First, from the cell-averages, $\bar{w}^{n+1}_{j+\frac{1}{2}}$, $\bar{w}^{n+1}_j$, given by (3.4)–(3.5), we reconstruct a third-order, piecewise polynomial, nonoscillatory interpolant (e.g., the CWENO interpolant described in section 2.1), which we will denote by $\tilde{w}^{n+1}_{j+1/2}(x)$ and $\tilde{w}^{n+1}_j(x)$, respectively. In fact, we do not need any high-order reconstruction $\tilde{w}^{n+1}_j(x)$ since it will be averaged out (consult Figure 3.1).

We note in passing that even for a nonuniform grid data, the CWENO interpolant can be written explicitly (in the spirit of section 2.1), but these details are irrelevant for the semidiscrete scheme, which will be described in section 4. At that point, all we need is to assume that such a reconstruction exists and that for all j it takes the form

$$\begin{aligned}\tilde{w}^{n+1}_{j+1/2}(x) &= \tilde{A}_{j+1/2} + \tilde{B}_{j+1/2}(x-x_{j+1/2}) + \frac{1}{2}\tilde{C}_{j+1/2}(x-x_{j+1/2})^2,\\ \tilde{w}^{n+1}_{j}(x) &= \bar{w}^{n+1}_j,\end{aligned} \tag{3.6}$$

in the nonsmooth region $(x^n_{j+1/2,l}, x^n_{j+1/2,r})$ and in the smooth region $(x^n_{j-1/2,r}, x^n_{j+1/2,l})$, respectively. Given (3.4), (3.5), and (3.6), we conclude by computing the new cell-averages at time t^{n+1} according to

$$\begin{aligned}\bar{u}_j^{n+1} &= \frac{1}{\Delta x}\left[\int_{x_{j-1/2}}^{x^n_{j-1/2,r}} \tilde{w}^{n+1}_{j-1/2}(x)dx + \int_{x^n_{j-1/2,r}}^{x^n_{j+1/2,l}} \tilde{w}^{n+1}_{j}(x)dx + \int_{x^n_{j+1/2,l}}^{x_{j+1/2}} \tilde{w}^{n+1}_{j+1/2}(x)dx\right]\\ &= \lambda a^n_{j-1/2}\tilde{A}_{j-1/2} + \left[1-\lambda(a^n_{j-1/2}+a^n_{j+1/2})\right]\bar{w}_j^{n+1} + \lambda a^n_{j+1/2}\tilde{A}_{j+1/2}\\ &\quad + \frac{\lambda\Delta t}{2}\left((a^n_{j-1/2})^2\tilde{B}_{j-1/2} - (a^n_{j+1/2})^2\tilde{B}_{j+1/2}\right)\\ &\quad + \frac{\lambda(\Delta t)^2}{6}\left((a^n_{j-1/2})^3\tilde{C}_{j-1/2} + (a^n_{j+1/2})^3\tilde{C}_{j+1/2}\right).\end{aligned} \tag{3.7}$$

Remark. The third-order reconstruction (3.6) is necessary in order to guarantee the overall third-order accuracy, since simple averaging over $[x_{j-\frac{1}{2}}, x_{j+\frac{1}{2}}]$ (without reconstruction) reduces the order of the resulting scheme (see [10]).

4. The semidiscrete scheme. We are now ready to derive our main result, which is the new third-order, semidiscrete, central scheme. First, we describe our ideas in the 1D framework and then we extend them to multidimensional problems.

4.1. 1D problems. We start with the derivation of the third-order semidiscrete scheme for 1D (systems of) hyperbolic conservation laws. Using the fully discrete scheme obtained in section 3, the semidiscrete approximation can be directly written as the limit

$$\frac{d}{dt}\bar{u}_j(t) = \lim_{\Delta t\to 0}\frac{\bar{u}_j^{n+1}-\bar{u}_j^n}{\Delta t}. \tag{4.1}$$

Substituting (3.7) into (4.1) results in

$$\begin{aligned}\frac{d\bar{u}_j}{dt} = \lim_{\Delta t\to 0}\Bigg\{ &\frac{1}{\Delta x}a^n_{j-1/2}\tilde{A}_{j-1/2} - \frac{1}{\Delta x}(a^n_{j-1/2}+a^n_{j+1/2})\bar{w}_j^{n+1}\\ &+\frac{1}{\Delta x}a^n_{j+1/2}\tilde{A}_{j+1/2} + \frac{1}{\Delta t}(\bar{w}_j^{n+1}-\bar{u}_j^n)\Bigg\}.\end{aligned} \tag{4.2}$$

In the limit as $\Delta t\to 0$, all the Riemann fans have zero widths and therefore,

$$\tilde{A}_{j+1/2} = \bar{w}^{n+1}_{j+1/2}, \qquad \tilde{A}_{j-1/2} = \bar{w}^{n+1}_{j-1/2}. \tag{4.3}$$

Using (3.3) we can also obtain

$$\begin{aligned}u(x^n_{j+1/2,r},t) &\to P_{j+1}(x_{j+1/2},t)\\ &= A_{j+1} - \frac{\Delta x}{2}B_{j+1} + \frac{(\Delta x)^2}{8}C_{j+1} =: u^+_{j+1/2}(t),\\ u(x^n_{j+1/2,l},t) &\to P_j(x_{j+1/2},t)\\ &= A_j + \frac{\Delta x}{2}B_j + \frac{(\Delta x)^2}{8}C_j =: u^-_{j+1/2}(t).\end{aligned} \tag{4.4}$$

Finally, plugging (3.4), (3.5), and (4.3) into (4.2) we compute the time limit explicitly, ending up with our new semidiscrete scheme,

$$\frac{d\bar{u}_j}{dt} = -\frac{1}{2\Delta x}\left[f(u^+_{j+1/2}(t)) + f(u^-_{j+1/2}(t)) - f(u^+_{j-1/2}(t)) - f(u^-_{j-1/2}(t))\right]$$

$$+\frac{a_{j+1/2}(t)}{2\Delta x}\left[u^+_{j+1/2}(t)-u^-_{j+1/2}(t)\right]-\frac{a_{j-1/2}(t)}{2\Delta x}\left[u^+_{j-1/2}(t)-u^-_{j-1/2}(t)\right], \tag{4.5}$$

with local speeds $a_{j+1/2}(t)$, e.g.,

$$a_{j+1/2}(t):=\max\left\{\rho\Big(\frac{\partial f}{\partial u}(u^-_{j+1/2}(t))\Big),\rho\Big(\frac{\partial f}{\partial u}(u^+_{j+1/2}(t))\Big)\right\}.$$

Remarks.

1. We would like to emphasize that the scheme (4.5) was derived independently of any specific piecewise-quadratic reconstruction. If one wants, e.g., to use the CWENO reconstruction described in section 2.1, then the values of A_j, B_j, and C_j in (4.4) are

$$\begin{aligned}
A_j&=\bar u^n_j-\frac{w_{\rm C}}{12}(\bar u^n_{j+1}-2\bar u^n_j+\bar u^n_{j-1}),\\
B_j&=\frac{1}{\Delta x}\left[w_{\rm R}(\bar u^n_{j+1}-\bar u^n_j)+w_{\rm C}\frac{\bar u^n_{j+1}-\bar u^n_{j-1}}{2}+w_{\rm L}(\bar u^n_j-\bar u^n_{j-1})\right],\\
C_j&=2w_{\rm C}\frac{\bar u^n_{j-1}-2\bar u^n_j+\bar u^n_{j+1}}{\Delta x^2},
\end{aligned}$$

where $w_{\rm L}, w_{\rm C}$, and $w_{\rm R}$ are defined in (2.8).

2. Our third-order scheme, (4.5), admits the conservative form

$$\frac{d\bar u_j}{dt}=-\frac{H_{j+1/2}(t)-H_{j-1/2}(t)}{\Delta x}, \tag{4.6}$$

with the numerical flux

$$\begin{aligned}
H_{j+1/2}(t):=&\frac{f(u^+_{j+1/2}(t))+f(u^-_{j+1/2}(t))}{2}\\
&-\frac{a_{j+1/2}(t)}{2}\left[u^+_{j+1/2}(t)-u^-_{j+1/2}(t)\right].
\end{aligned} \tag{4.7}$$

This scheme is a natural generalization of the second-order semidiscrete scheme from [16]. Moreover, the second-order scheme has exactly the same form, (4.6)–(4.7); the only difference is in the more accurate computation of the intermediate value $u^+_{j+1/2}(t)$ and $u^-_{j+1/2}(t)$. It is interesting to note that also the fully discrete, staggered, second- and third-order central schemes have the same structure (see [28]).

3. Similar to the case of the second-order scheme [16], the nonoscillatory property of the piecewise parabolic reconstruction, (3.3), will guarantee the nonoscillatory nature of our semidiscrete scheme. But unlike the piecewise linear reconstruction utilized in the second-order method, a piecewise parabolic reconstruction can be only essentially nonoscillatory. This means that, in principle, such a reconstruction may increase the total variation of the computed piecewise constant solution. Our numerical examples, however, demonstrate that the growth of the total variation is always bounded. Such desirable behavior of bounded total variation in the context of central-WENO schemes was already observed in [23].

4. We would like to stress once again the simplicity of our new method, which does not require any (approximate) Riemann solver or any use of the characteristic variables—the reconstruction of piecewise polynomial interpolant, (3.3), is carried

out *componentwise*. In particular, unlike the standard central schemes, but similar to the second-order semidiscrete method in [16], our method is based on one grid (and not on staggering between two grids). This can be a big advantage (compared with the traditional central schemes) when dealing with boundary conditions and complex geometries.

5. Finally, similar to the second-order semidiscrete scheme [16], the third-order scheme, (4.6)–(4.7), boils down in the scalar linear case to an upwind scheme. In the nonlinear, *scalar*, case, this is a finite volume scheme based on a local LxF (Rusanov) monotone flux. Since we derived our method as a central Godunov-type scheme, our result also naturally holds for systems (including multi-dimensional systems as shown below).

Next, let us consider the general convection-diffusion equation, (1.2). Similar to the case of the second-order semidiscrete scheme [16], operator splitting is not needed. We can apply our third-order semidiscrete scheme, (4.6)–(4.7), to the (degenerate) parabolic equation, (1.2), in a straightforward manner. This results in the scheme

$$\frac{d\bar{u}_j}{dt} = -\frac{H_{j+1/2}(t) - H_{j-1/2}(t)}{\Delta x} + Q_j(t). \tag{4.8}$$

Here, $H_{j+1/2}(t)$ is our numerical convection flux, (4.7), and $Q_j(t)$ is a high-order approximation to the diffusion term, $Q(u, u_x)_x$. In the examples below we use the fourth-order central differencing of the form

$$\begin{aligned} Q_j(t) = \frac{1}{12\Delta x}\Big[&- Q(u_{j+2}(t), (u_x)_{j+2,j}) + 8Q(u_{j+1}(t), (u_x)_{j+1,j}) \\ &-8Q(u_{j-1}(t), (u_x)_{j-1,j}) + Q(u_{j-2}(t), (u_x)_{j-2,j})\Big], \end{aligned} \tag{4.9}$$

where

$$\begin{aligned} (u_x)_{j+2,j} &:= \frac{1}{12\Delta x}\Big[25u_{j+2}(t) - 48u_{j+1}(t) + 36u_j - 16u_{j-1}(t) + 3u_{j-2}(t)\Big], \\ (u_x)_{j+1,j} &:= \frac{1}{12\Delta x}\Big[3u_{j+2}(t) + 10u_{j+1}(t) - 18u_j + 6u_{j-1}(t) - u_{j-2}(t)\Big], \\ (u_x)_{j-1,j} &:= \frac{1}{12\Delta x}\Big[u_{j+2}(t) - 6u_{j+1}(t) + 18u_j - 10u_{j-1}(t) - 3u_{j-2}(t)\Big], \\ (u_x)_{j-2,j} &:= \frac{1}{12\Delta x}\Big[-3u_{j+2}(t) + 16u_{j+1}(t) - 36u_j + 48u_{j-1}(t) - 25u_{j-2}(t)\Big], \end{aligned} \tag{4.10}$$

and $\{u_j(t)\}$ are point-values of the reconstructed polynomials, (3.3), i.e., $u_j(t) = P_j(x_j, t)$.

4.2. Multidimensional extensions. Without loss of generality, let us consider the 2D (system of) convection-diffusion equations,

$$u_t + f(u)_x + g(u)_y = Q^x(u, u_x, u_y)_x + Q^y(u, u_x, u_y)_y, \tag{4.11}$$

where the case $Q^x \equiv Q^y \equiv 0$ corresponds to the 2D pure hyperbolic problem.

Suppose that we have computed an approximate solution to (4.11) at some time t and have reconstructed a 2D piecewise polynomial, third-order, ENO interpolant over the uniform spatial grid, $(x_j, y_k) = (j\Delta x, k\Delta y)$.

Following [16], the 2D extension of our third-order semidiscrete scheme, (4.8),(4.7), can be written in the following form:

$$\frac{d\bar{u}_{j,k}}{dt} = -\frac{H^x_{j+1/2,k}(t) - H^x_{j-1/2,k}(t)}{\Delta x} - \frac{H^y_{j,k+1/2}(t) - H^y_{j,k-1/2}(t)}{\Delta y} + Q^x_{j,k}(t) + Q^y_{j,k}(t). \tag{4.12}$$

Here, $H^x_{j+1/2,k}(t)$ and $H^y_{j,k+1/2}(t)$ are x- and y-numerical convection fluxes, respectively (they can be viewed as a generalization of the 1D flux, (4.7)),

$$\begin{aligned} H^x_{j+1/2,k}(t) &:= \frac{f(u^+_{j+1/2,k}(t)) + f(u^-_{j+1/2,k}(t))}{2} \\ &\quad - \frac{a^x_{j+1/2,k}(t)}{2}\left[u^+_{j+1/2,k}(t) - u^-_{j+1/2,k}(t)\right], \\ H^y_{j,k+1/2}(t) &:= \frac{g(u^+_{j,k+1/2}(t)) + g(u^-_{j,k+1/2}(t))}{2} \\ &\quad - \frac{a^y_{j,k+1/2}(t)}{2}\left[u^+_{j,k+1/2}(t) - u^-_{j,k+1/2}(t)\right]. \end{aligned} \tag{4.13}$$

The numerical fluxes, (4.13), are expressed in terms of the intermediate values, $u^{\pm}_{j+1/2,k}(t)$, $u^{\pm}_{j,k+1/2}(t)$, which are obtained from the piecewise polynomial reconstruction. The local speeds, $a^x_{j+1/2,k}(t)$ and $a^y_{j,k+1/2}(t)$, are computed, e.g., by

$$\begin{aligned} a^x_{j+1/2,k}(t) &:= \max\left\{\rho\Big(\frac{\partial f}{\partial u}(u^-_{j+1/2,k}(t))\Big), \rho\Big(\frac{\partial f}{\partial u}(u^+_{j+1/2,k}(t))\Big)\right\}, \\ a^y_{j,k+1/2}(t) &:= \max\left\{\rho\Big(\frac{\partial g}{\partial u}(u^-_{j,k+1/2}(t))\Big), \rho\Big(\frac{\partial g}{\partial u}(u^+_{j,k+1/2}(t))\Big)\right\}. \end{aligned} \tag{4.14}$$

Finally, $Q^x_{j,k}(t)$ and $Q^y_{j,k}(t)$ are high-order, central differencing approximations to the diffusion terms $Q^x(u, u_x, u_y)_x$ and $Q^y(u, u_x, u_y)_y$.

Remarks.

1. We would like to emphasize that the problem of constructing a 2D, third-order, nonoscillatory interpolant is highly nontrivial. Several essentially 2D reconstructions were proposed in [20, 21, 22]. Alternatively, one can use 1D CWENO reconstruction, direction by direction, in order to compute the intermediate values $u^{\pm}_{j+1/2,k}(t)$ and $u^{\pm}_{j,k+1/2}(t)$.

Following is the recipe for the computation of $u^-_{j+1/2,k}$ (the computation of other intermediate values can be carried out in a similar way):

$$u^-_{j+1/2,k} = w_{\text{L}} P^k_{\text{L}}(x_{j+1/2}) + w_{\text{R}} P^k_{\text{R}}(x_{j+1/2}) + w_{\text{C}} P^k_{\text{C}}(x_{j+1/2}), \tag{4.15}$$

where the P's are the polynomials introduced in section 2.1,

$$\begin{aligned} P^k_{\text{R}}(x) &= \bar{u}_{j,k} + \frac{\bar{u}_{j+1,k} - \bar{u}_{j,k}}{\Delta x}(x - x_j), \\ P^k_{\text{L}}(x) &= \bar{u}_{j,k} + \frac{\bar{u}_{j,k} - \bar{u}_{j-1,k}}{\Delta x}(x - x_j), \end{aligned}$$

(4.16)
$$P^k_{\rm C}(x) = \bar u_{j,k} - \frac{1}{12}(\bar u_{j+1,k} - 2\bar u_{j,k} + \bar u_{j-1,k}) - \frac{1}{12}(\bar u_{j,k+1} - 2\bar u_{j,k} + \bar u_{j,k-1}) + \frac{\bar u_{j+1,k} - \bar u_{j-1,k}}{2\Delta x}(x - x_j) + \frac{\bar u_{j+1,k} - 2\bar u_{j,k} + \bar u_{j-1,k}}{\Delta x^2}(x - x_j)^2.$$

The weights, $w_{\rm L}, w_{\rm R}, w_{\rm C}$, which are given by (2.8), are based on the smoothness indicators in (2.9).

Note that the only difference between this reconstruction and the 1D reconstruction, (2.4)–(2.9), is an additional term in $P^k_{\rm C}(x)$, $-\frac{1}{12}(\bar u_{j,k+1} - 2\bar u_{j,k} + \bar u_{j,k-1})$, which corresponds to the second derivative in the y direction and guarantees the third-order accuracy of the computed intermediate values. This "dimension by dimension" approach was implemented in Example 5 below.

2. It is straightforward to extend the 2D scheme, (4.12), to more space dimensions. In particular, the dimension-by-dimension approach is a very simple and promising approach for multidimensional problems.

5. Numerical examples. We conclude the paper with a number of numerical examples. Here, in order to retain the overall high accuracy, the semidiscrete scheme is combined with a high-order, stable ODE solver to complete the spatio-temporal discretization. Numerically, we observed that a variety of explicit methods provides satisfactory results in the context of our semidiscrete scheme.

For the inviscid problems (Examples 1, 2, 3, and 5), we used the third-order total variation diminishing (TVD) Runge–Kutta-type method introduced by Shu and Osher in [33]. However, if we apply this time-integration method or any other standard Runge–Kutta-type method to (degenerate) parabolic problems, the time-step can be very small due to their strict stability restrictions.

To overcome this difficulty, we used (in Examples 4 and 5) the third-order ODE solver (called DUMKA3) by Medovikov [29]. This explicit method has larger stability domains (compared with the standard Runge–Kutta methods), which allow larger time-steps. In practice, DUMKA3 works as fast as implicit methods (see [29] for details).

We abbreviate our third-order semidiscrete scheme by SD3, which will be combined with the third-order TVD Runge–Kutta-type method (RK3) or with DUMKA3.

Example 1: Linear accuracy test. Consider the scalar linear hyperbolic equation

$$u_t + u_x = 0, \qquad x \in [0, 2\pi], \tag{5.1}$$

augmented with the smooth initial data, $u(x,0) = \sin x$, and periodic boundary conditions. This simple problem admits a global classical solution, which was computed at time $T = 1$ with a varying number of grid points, N.

In Table 5.1 we check the accuracy of our third-order semidiscrete scheme, SD3, coupled with the RK3 ODE solver. Clearly, this is a high-order method. The asymptotic convergence rate seems to be better than three, which is similar to the superconvergence observed in [27].

The error is measured in terms of the pointwise values,

$$\|\tilde u - u\|_{L^1} := \Delta x \sum_j |\tilde u_j(T) - u(x_j, T)|, \quad \|\tilde u - u\|_{L^\infty} := \max_j |\tilde u_j(T) - u(x_j, T)|.$$

TABLE 5.1
Accuracy test for the linear advection problem (5.1)*; the errors at* $T = 1$.

N	L^1-error	Rate	L^∞-error	Rate
40	4.492e-02	–	2.822e-02	–
80	1.092e-02	2.04	1.065e-02	1.41
160	2.162e-03	2.34	3.426e-03	1.64
320	1.811e-04	3.58	4.705e-04	2.86
640	9.267e-06	4.29	2.267e-05	4.38
1280	5.409e-07	4.10	1.171e-06	4.27

TABLE 5.2
Accuracy test for the Burgers equation, (5.2)*; the pre-shock errors.*

N	L^1-error	Rate	L^∞-error	Rate
40	2.370e-02	–	2.225e-02	–
80	5.759e-03	2.04	9.053e-03	1.30
160	1.161e-03	2.31	2.921e-03	1.63
320	9.541e-05	3.61	3.926e-04	2.90
640	4.882e-06	4.29	1.778e-05	4.46
1280	3.044e-07	4.00	5.732e-07	4.96

Here, $\tilde{u}$ is an approximate solution, which is realized by its values at the grid points, x_j,

$$\tilde{u}_j(T) = P_j(x_j, T),$$

where the P_j's are the piecewise parabolic interpolants, (3.3), constructed at the final time $t = T$.

Example 2: The Burgers equation. In this example we approximate solutions to the inviscid Burgers equation,

$$u_t + \left(\frac{u^2}{2}\right)_x = 0, \qquad x \in [0, 2\pi], \tag{5.2}$$

augmented with the smooth initial data, $u(x, 0) = 0.5 + \sin x$, and periodic boundary conditions.

The unique entropy solution of (5.2) develops a shock discontinuity at the critical time $T_c = 1$. Table 5.2 shows the L^1- and L^∞-norms of the errors at the preshock time $T = 0.5$, when the solution is still smooth. Once again, our results indicate that the method is a high-order method also when implemented for nonlinear problems.

In Figures 5.1 and 5.2 we present the approximate solutions at the postshock time $T = 2$, when the shock is well developed. The essentially nonoscillatory nature of our scheme can be clearly observed.

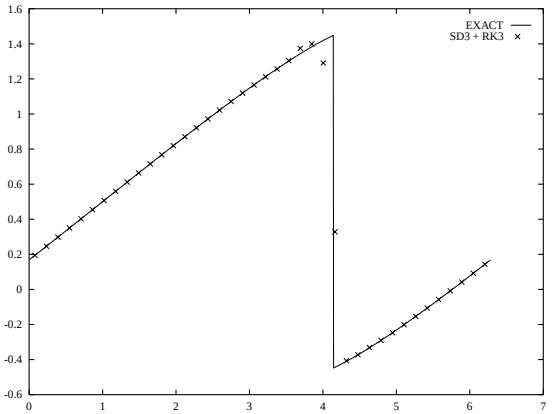


FIG. 5.1. *The Burgers equation,* (5.2)*;* $T = 2$*,* $N = 40$.

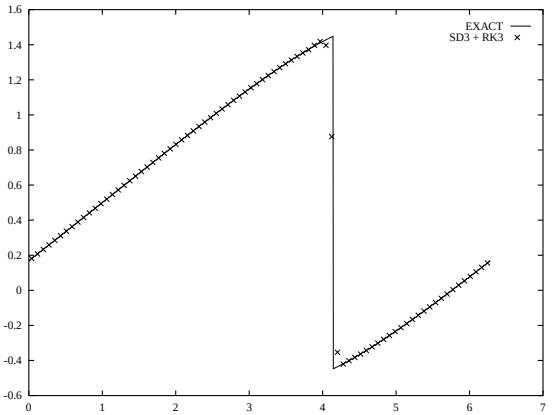


FIG. 5.2. *The Burgers equation,* (5.2)*;* $T = 2$*,* $N = 80$.

Example 3: Euler equations of gas dynamics. Let us consider the 1D Euler system

$$\frac{\partial}{\partial t}\begin{bmatrix} \rho \\ m \\ E \end{bmatrix} + \frac{\partial}{\partial x}\begin{bmatrix} m \\ \rho u^2 + p \\ u(E+p) \end{bmatrix} = 0, \qquad p = (\gamma - 1) \cdot \left(E - \frac{\rho}{2}u^2\right),$$

where ρ, u, $m = \rho u$, p, and E are the density, velocity, momentum, pressure, and total energy, respectively. We solve this system subject to Sod's Riemann initial data, proposed in [34],

$$\vec{u}(x,0) = \begin{cases} \vec{u}_L = (1,0,2.5)^T, & x < 0, \\ \vec{u}_R = (0.125,0,0.25)^T, & x > 0. \end{cases}$$

The approximations to the density, velocity, and pressure obtained by the SD3 scheme with the RK3 time discretization are presented in Figures 5.3–5.8. The coefficient p in the smoothness indicator, (2.8)–(2.9), was taken as 0.6, which seems to be the optimal value in this specific example. WENO-type schemes do require a parameter tuning in order to reduce the amplitude of the oscillations. Formally, this does not affect the order of accuracy of the method (in certain ranges of the parameters).

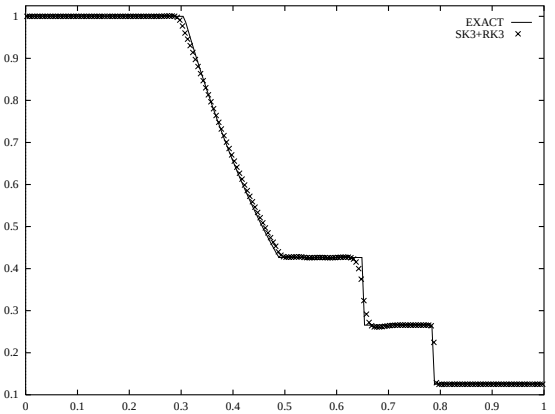


FIG. 5.3. *Sod problem*—density. $N = 200$, $T = 0.1644$.

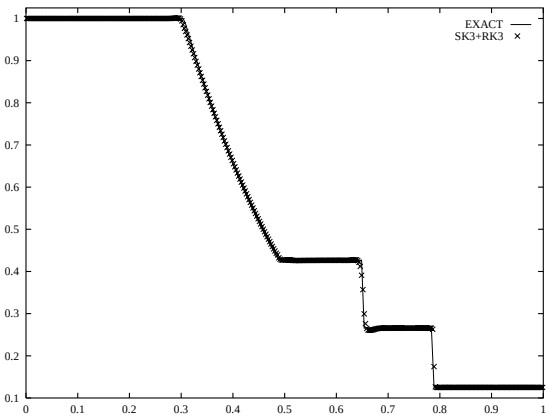


FIG. 5.4. *Sod problem*—density. $N = 400$, $T = 0.1644$.

We would like to stress again that our SD3 scheme does not employ the characteristic decomposition. To improve the resolution of the contact discontinuity, which is always smeared while the solution to the system of Euler equations is computed by the central method, we implemented the artificial compression method (ACM) by Harten [8]. In the context of central schemes, the ACM can be implemented as a corrector step to the componentwise approach (see [30] for details).

Example 4: Convection-diffusion equations—the Buckley–Leverett model. In this example we solve the 1D Buckley–Leverett equation,

$$u_t + f(u)_x = \varepsilon(\nu(u)u_x)_x, \quad \varepsilon\nu(u) \geq 0, \tag{5.3}$$

which can be viewed as a prototype model for the two-phase flow in oil reservoirs. Typically, $\nu(u)$ vanishes at some values of u, and thus (5.3) is a degenerate parabolic equation. Specifically, we take

$$f(u) = \frac{u^2}{u^2 + (1-u)^2}, \quad \nu(u) = 4u(1-u), \quad \varepsilon = 0.01,$$

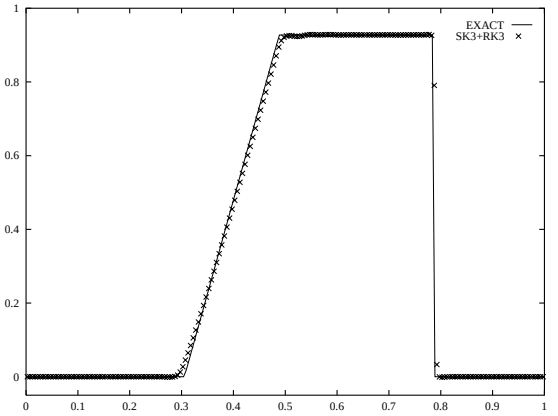


FIG. 5.5. *Sod problem*—velocity. $N = 200$, $T = 0.1644$.

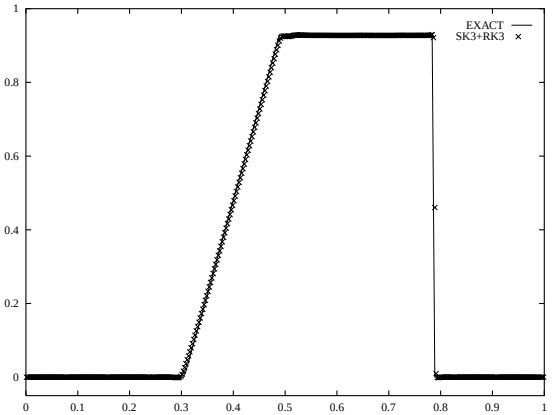


FIG. 5.6. *Sod problem*—velocity. $N = 400$, $T = 0.1644$.

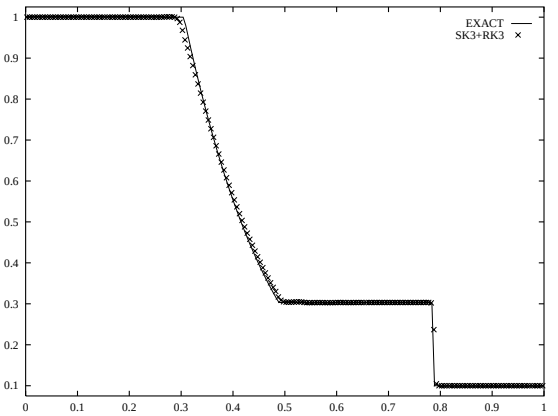


FIG. 5.7. *Sod problem*—pressure. $N = 200$, $T = 0.1644$.

and consider the initial value problem with the Riemann initial data,

$$u(x,0) = \begin{cases} 0, & 0 \le x < 1 - \frac{1}{\sqrt{2}}, \\ 1, & 1 - \frac{1}{\sqrt{2}} \le x \le 1. \end{cases} \tag{5.4}$$

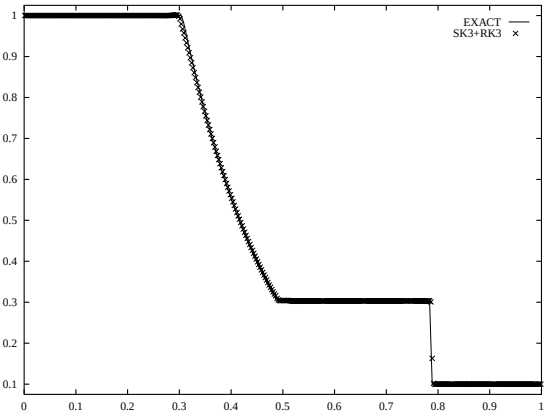


FIG. 5.8. *Sod problem*—pressure. $N = 400$, $T = 0.1644$.

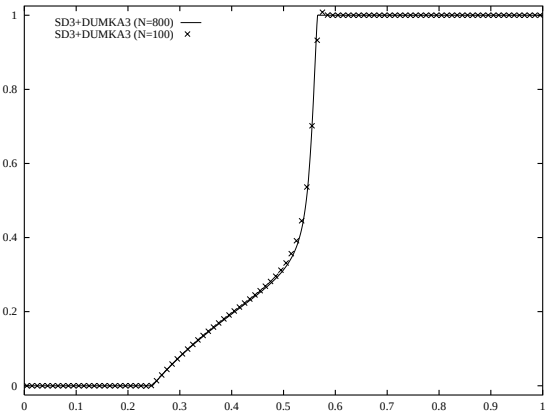


FIG. 5.9. *Buckley–Leverett model,* (5.3)–(5.4). $T = 0.2$.

The numerical solution to this problem, obtained by the SD3 scheme augmented with the DUMKA3 ODE solver, is presented in Figure 5.9.

The model, (5.3), becomes more complicated by adding the effects of gravitation. These effects can be obtained, e.g., by taking

$$f(u) = \frac{u^2}{u^2 + (1-u)^2}(1 - 5(1-u)^2), \tag{5.5}$$

which is nonmonotone on the interval $u \in [0, 1]$.

The numerical solution to this initial value problem is shown in Figure 5.10. Note that the exact solution to problem (5.3)–(5.4) is not available, but our solutions seem to converge to the physically relevant solutions in both cases—with gravitation and without it.

Example 5: Incompressible Euler and Navier–Stokes equations. In this example we consider 2D viscous and inviscid incompressible flow governed by the Navier–Stokes ($\nu > 0$) and Euler ($\nu = 0$) equations,

$$\vec{u}_t + (\vec{u} \cdot \nabla)\vec{u} + \nabla p = \nu \Delta \vec{u}. \tag{5.6}$$

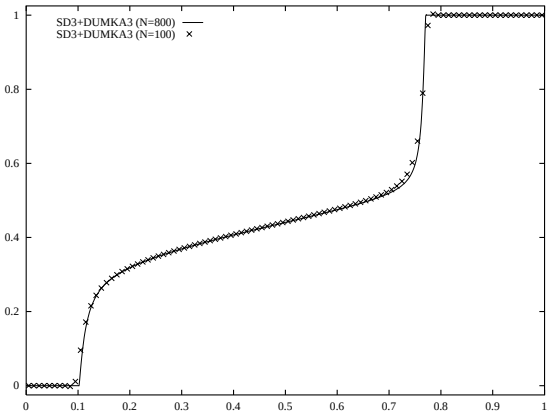


FIG. 5.10. *Buckley–Leverett model,* (5.3)–(5.4)*, including the gravitational effect,* (5.5)*.* $T = 0.2$.

Here, p denotes the pressure, and $\vec{u} = (u, v)$ is the two-component divergence-free velocity field satisfying

$$u_x + v_y = 0. \tag{5.7}$$

In the 2D case (5.6) admits an equivalent scalar formulation in terms of the vorticity,

$$\omega_t + (u\omega)_x + (v\omega)_y = \nu\Delta\omega, \tag{5.8}$$

where $\omega := v_x - u_y$. The incompressibility, (5.7), implies that (5.8) can be written in an equivalent conservative form,

$$\omega_t + f(\omega)_x + g(\omega)_y = \nu\Delta\omega, \tag{5.9}$$

with a *global* convection flux, $(f, g) := (u\omega, v\omega)$. A second-order, fully discrete, staggered, central scheme was used to solve the 2D vorticity equations in [24]. This scheme was proved to satisfy a maximum principle for the vorticity. (For an equivalent scheme in the velocity formulation, see [13].)

When applied to (5.9), our 2D, third-order, semidiscrete scheme, (4.12)–(4.14), takes the form

$$\begin{aligned}\frac{d\bar{\omega}_{j,k}}{dt} = &-\frac{H^x_{j+1/2,k}(t) - H^x_{j-1/2,k}(t)}{\Delta x}\\ &-\frac{H^y_{j,k+1/2}(t) - H^y_{j,k-1/2}(t)}{\Delta y} + \nu Q_{j,k}(t),\end{aligned} \tag{5.10}$$

with the numerical convection fluxes,

$$\begin{aligned}H^x_{j+1/2,k}(t) = &\frac{u_{j+1/2,k}(t)}{2}\left[\omega^+_{j+1/2,k}(t) + \omega^-_{j+1/2,k}(t)\right]\\ &-\frac{a^x_{j+1/2,k}(t)}{2}\left[\omega^+_{j+1/2,k}(t) - \omega^-_{j+1/2,k}(t)\right],\\ H^y_{j,k+1/2}(t) = &\frac{v_{j,k+1/2}(t)}{2}\left[\omega^+_{j,k+1/2}(t) + \omega^-_{j,k+1/2}(t)\right]\\ &-\frac{a^y_{j,k+1/2}(t)}{2}\left[\omega^+_{j,k+1/2}(t) - \omega^-_{j,k+1/2}(t)\right],\end{aligned} \tag{5.11}$$

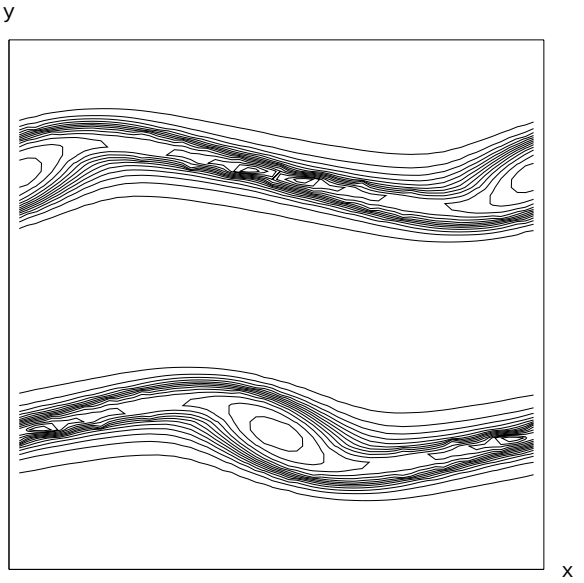


FIG. 5.11. *Incompressible Euler equations; third-order method;* $T = 4$, 64×64 *grid.*

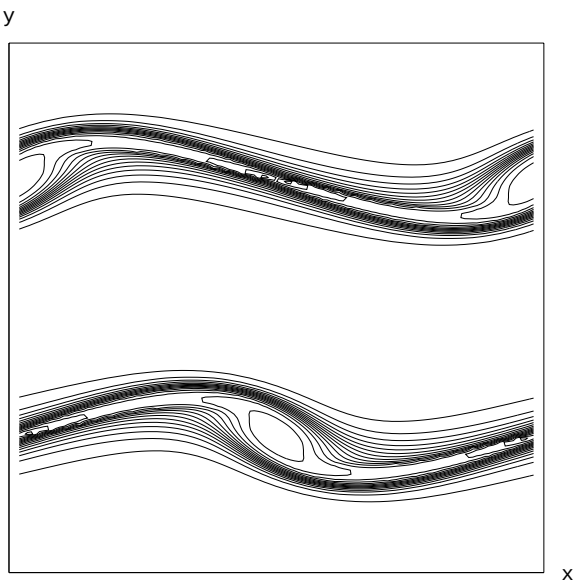


FIG. 5.12. *Incompressible Euler equations; third-order method;* $T = 4$, 128×128 *grid.*

and the local speeds,

$$a^x_{j+1/2,k}(t) := |u_{j+1/2,k}(t)|, \qquad a^y_{j,k+1/2}(t) := |v_{j,k+1/2}(t)|. \tag{5.12}$$

To approximate the linear viscosity, $\Delta\omega$, we used the fourth-order central differencing,

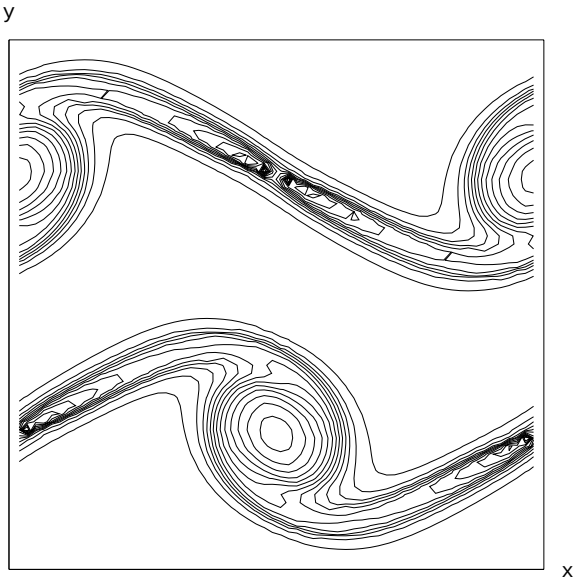


FIG. 5.13. *Incompressible Euler equations; third-order method;* $T = 6$, 64×64 *grid.*

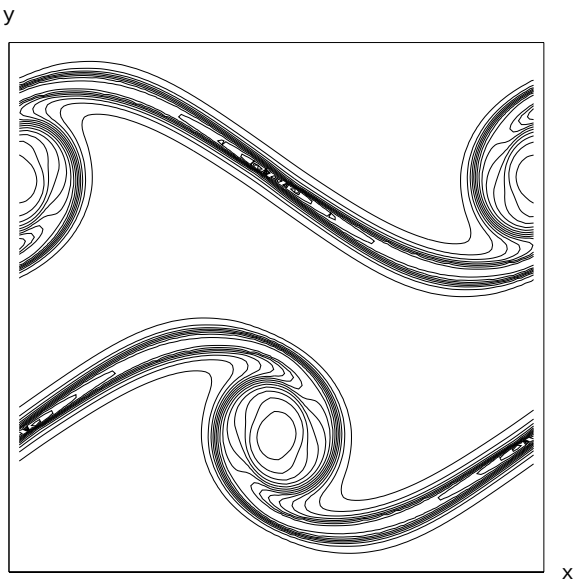


FIG. 5.14. *Incompressible Euler equations; third-order method;* $T = 6$, 128×128 *grid.*

$$Q_{j,k}(t) = \frac{-\omega_{j+2,k}(t) + 16\omega_{j+1,k}(t) - 30\omega_{j,k}(t) + 16\omega_{j-1,k}(t) - \omega_{j-2,k}(t)}{12\Delta x^2} + \frac{-\omega_{j,k+2}(t) + 16\omega_{j,k+1}(t) - 30\omega_{j,k}(t) + 16\omega_{j,k-1}(t) - \omega_{j,k-2}(t)}{12\Delta y^2}. \tag{5.13}$$

To compute the intermediate values of the vorticity, we use the "dimension by dimension" approach described in section 4.2: we reconstruct the corresponding CWENO

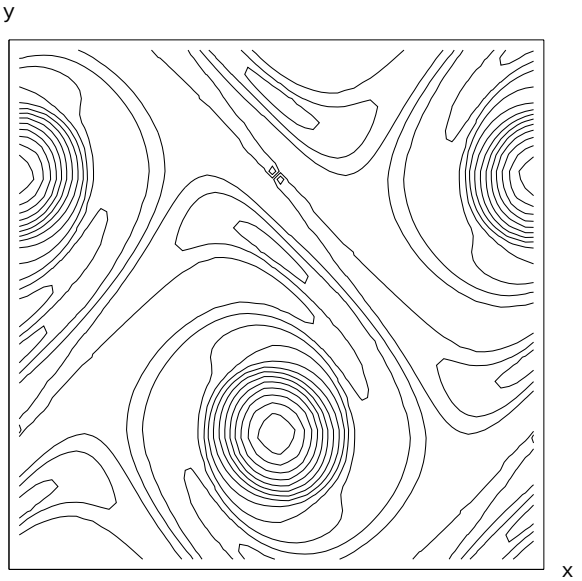


FIG. 5.15. *Incompressible Euler equations; third-order method;* $T = 10$, 64×64 *grid.*

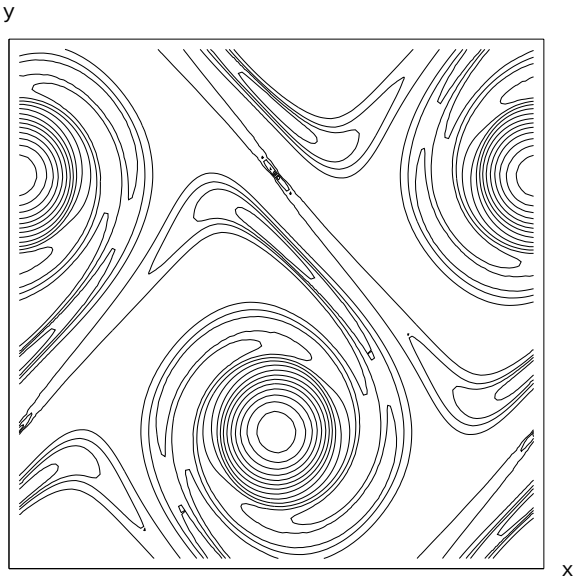


FIG. 5.16. *Incompressible Euler equations; third-order method;* $T = 10$, 128×128 *grid.*

interpolants in the x- and y-directions to obtain the values of $\omega^{\pm}_{j+1/2,k}$ and $\omega^{\pm}_{j,k+1/2}$.

Another important point in the incompressible computations is that in every time-step one has to recover the velocities, $\{u_{j,k}, v_{j,k}\}$, from the known values of the vorticity, $\{\omega_{j,k}\}$. This can be done in many different ways (consult, e.g., [24] and the references therein). Here we have used a stream-function, ψ, such that $\triangle\psi = -\omega$, which is obtained by solving the nine-points Laplacian, $\triangle\psi_{j,k} = -\omega_{j,k}(t)$. This provides the values of the stream-function with fourth-order accuracy. Its gradient,

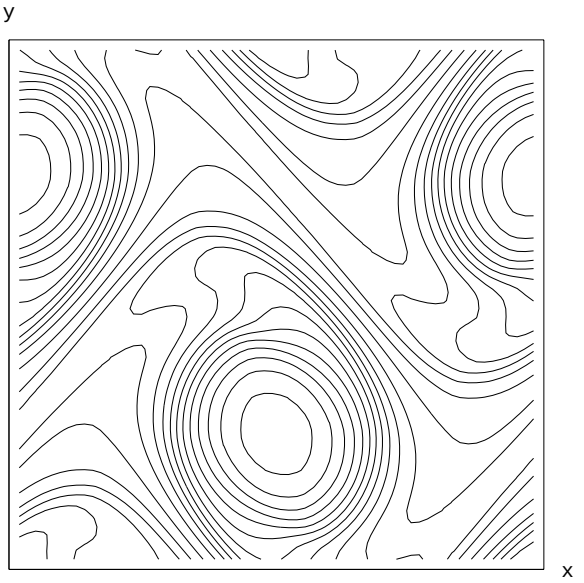


FIG. 5.17. *Incompressible Euler equations; second-order method;* $T = 10$, 64×64 *grid.*

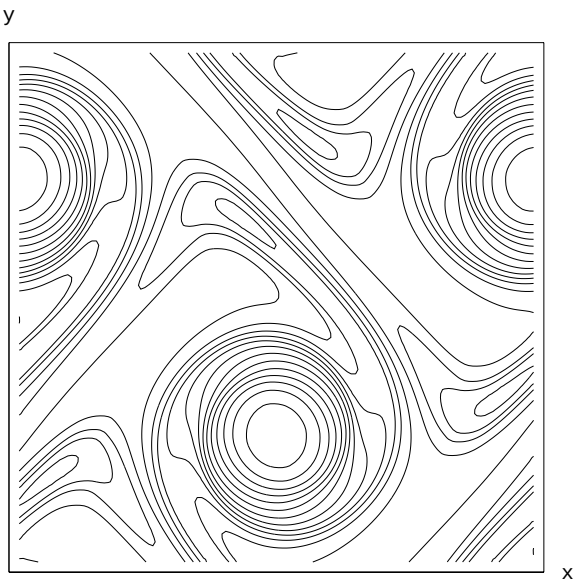


FIG. 5.18. *Incompressible Euler equations; second-order method;* $T = 10$, 128×128 *grid.*

$\nabla\psi$, then recovers the velocity field,

$$u_{j,k}(t) = \frac{-\psi_{j,k+2} + 8\psi_{j,k+1} - 8\psi_{j,k-1} + \psi_{j,k-2}}{12\Delta y},$$
$$v_{j,k}(t) = \frac{\psi_{j+2,k} - 8\psi_{j+1,k} + 8\psi_{j-1,k} - \psi_{j-2,k}}{12\Delta x}. \tag{5.14}$$

Remarks.

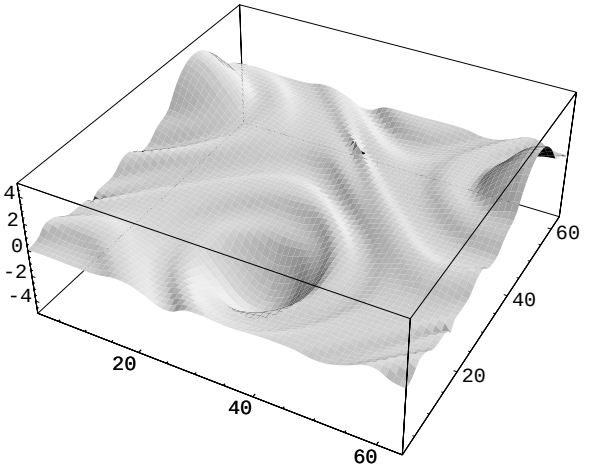


FIG. 5.19. *Incompressible Euler equations; third-order method;* $T = 10$, 64×64 *grid.*

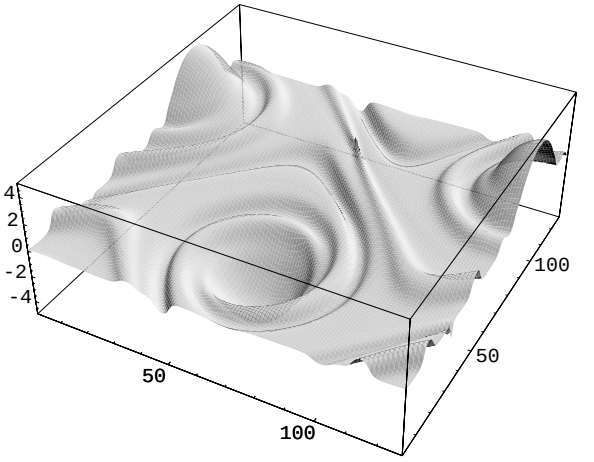


FIG. 5.20. *Incompressible Euler equations; third-order method;* $T = 10$, 128×128 *grid.*

1. Observe that in this way we retain the discrete incompressibility, namely, the discrete velocities computed in (5.14) satisfy

$$\frac{-u_{j+2,k} + 8u_{j+1,k} - 8u_{j-1,k} + u_{j-2,k}}{12\Delta x} + \frac{-v_{j,k+2} + 8v_{j,k+1} - 8v_{j,k-1} + v_{j,k-2}}{12\Delta y} = 0.$$

2. The point-values of the vorticity, which are required for using the nine-points Laplacian, were computed from its cell-averages using the "dimension by dimension" recipe, (4.15)–(4.16).

Finally, the intermediate values of velocities can be computed, e.g., using fourth-order averaging,

$$\begin{aligned} u_{j+1/2,k}(t) &= \frac{-u_{j+2,k}(t) + 9u_{j+1,k}(t) + 9u_{j,k}(t) - u_{j-1,k}(t)}{16}, \\ v_{j,k+1/2}(t) &= \frac{-v_{j,k+2}(t) + 9v_{j,k+1}(t) + 9v_{j,k}(t) - v_{j,k-1}(t)}{16}. \end{aligned} \tag{5.15}$$

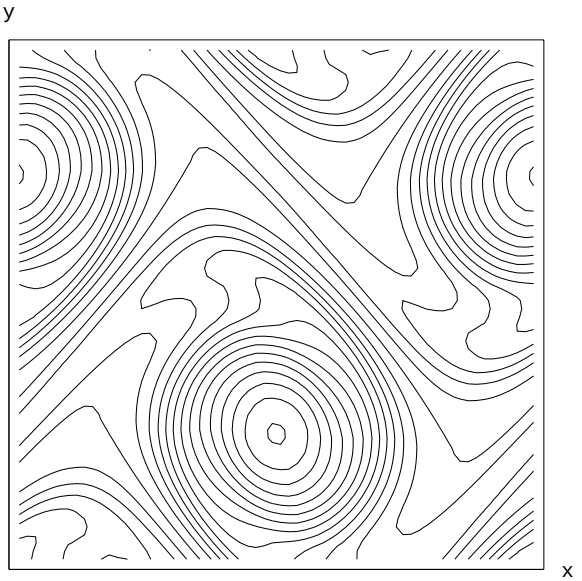


FIG. 5.21. *Incompressible Navier–Stokes equations; third-order method;* $T = 10$, 64×64 *grid.*

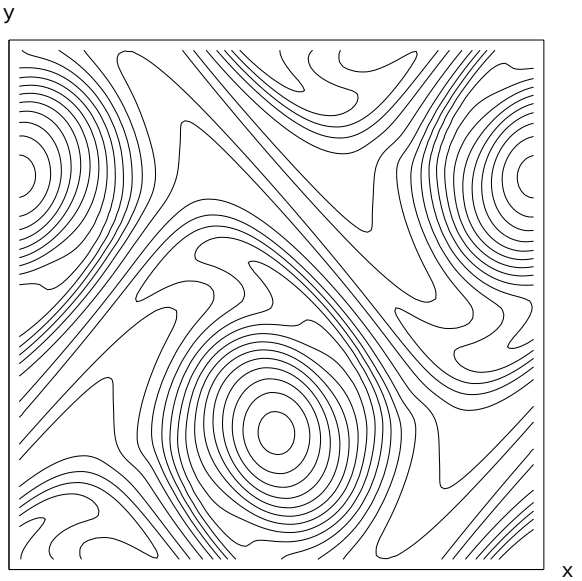


FIG. 5.22. *Incompressible Navier–Stokes equations; third-order method;* $T = 10$, 128×128 *grid.*

We start our numerical experiments by checking the accuracy of our scheme, (5.10)–(5.15), augmented with the DUMKA3 time discretization. We consider the Navier–Stokes equations, (5.6)–(5.7) with $\nu = 0.05$, subject to the smooth periodic initial data (taken from [4]),

$$u(x, y, 0) = -\cos(x)\sin(y), \qquad v(x, y, 0) = \sin(x)\cos(y). \tag{5.16}$$

The exact solution to this problem is simply an exponential decay of the initial data,

TABLE 5.3
Accuracy test for the Navier–Stokes equations. (5.6)–(5.7), (5.16), $\nu = 0.05$. *Errors at* $T = 2$.

$Nx \times Ny$	L^∞-error	Rate	L^1-error	Rate	L^2-error	Rate
32×32	2.429e-02	–	1.791e-01	–	4.559e-02	–
64×64	4.571e-03	2.41	2.814e-02	2.67	7.635e-03	2.58
128×128	8.342e-04	2.45	3.869e-03	2.86	1.146e-03	2.74
256×256	1.208e-04	2.79	4.966e-04	2.96	1.502e-04	2.93

given by

$$u(x,y,t) = -\cos(x)\sin(y)e^{-2\nu t}, \qquad v(x,y,t) = \sin(x)\cos(y)e^{-2\nu t}.$$

The approximate solution with a different number of grid points was computed at time $t = 2$. The errors, measured in terms of vorticity in the L^∞-, L^1-, and L^2-norms are shown in Table 5.3.

Next, the third-order semidiscrete scheme, (5.10)–(5.15), was implemented for the periodic double shear-layer model problem taken from [2]. First, we solve the Euler equations, (5.6)–(5.7) with $\nu = 0$, subject to the $(2\pi, 2\pi)$-periodic initial data,

$$u(x,y,0) = \begin{cases} \tanh(\frac{1}{\rho}(y - \pi/2)), & y \le \pi, \\ \tanh(\frac{1}{\rho}(3\pi/2 - y)), & y > \pi, \end{cases} \qquad v(x,y,0) = \delta \cdot \sin(x). \tag{5.17}$$

Here, the "thick" shear-layer width parameter, ρ, is taken as $\frac{\pi}{15}$ and the perturbation parameter $\delta = 0.05$.

The numerical results at times $T = 4, 6, 10$ with $N = 64 \times 64$ and $N = 128 \times 128$ grid points are presented in Figures 5.11 through 5.16 and 5.19 through 5.20. In order to compare the quality of the results obtained with our new method to previous results, we plot in Figures 5.17 and 5.18 the results obtained for the same double shear-layer problem with the second-order central scheme proposed in [24]. Compared with the second-order method, the new third-order method can resolve the large gradients better. Since we are using only an essentially nonoscillatory reconstruction, some oscillations are created with the third-order method (and not with the "fully" nonoscillatory second-order method).

Finally, we solve the Navier–Stokes equations, (5.6)–(5.7) with $\nu = 0.01$, augmented with the "thick" shear-layer periodic initial data, (5.17).

The numerical results at time $T = 10$ with $N = 64 \times 64$ and $N = 128 \times 128$ grid points are presented in Figures 5.21–5.24.

Acknowledgment. The authors would like to thank Professors S. Karni and R. S. Krasny for helpful comments.

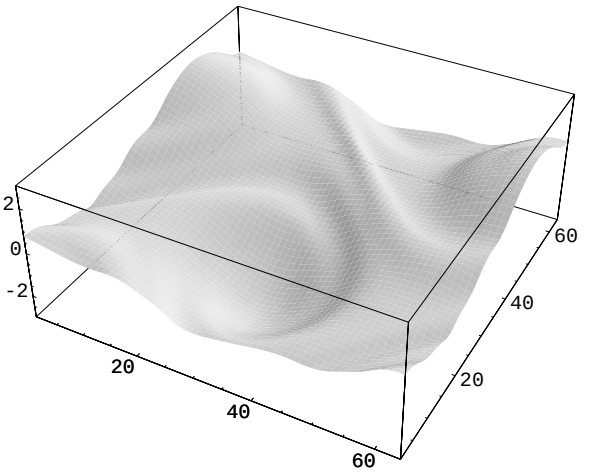


FIG. 5.23. *Incompressible Navier–Stokes equations; third-order method;* $T = 10$, 64×64 *grid.*

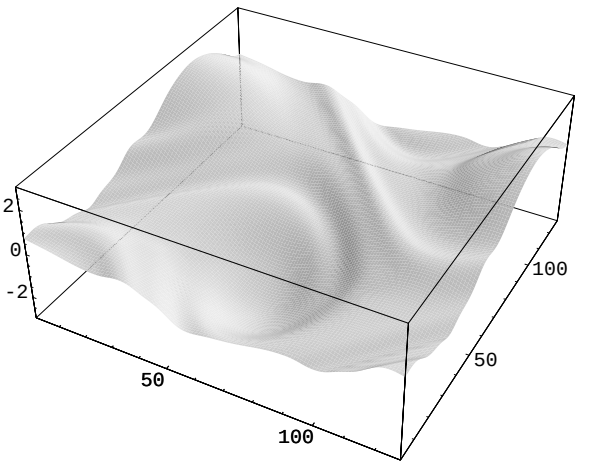


FIG. 5.24. *Incompressible Navier–Stokes equations; third-order method;* $T = 10$, 128×128 *grid.*

REFERENCES

[1] P. ARMINJON AND M. C. VIALLON, *Généralisation du Schéma de Nessyahu-Tadmor pour Une Équation Hyperbolique à Deux Dimensions D'espace*, C.R. Acad. Sci. Paris, Ser. I Math., 320 (1995), pp. 85–88.

[2] J. B. BELL, P. COLELLA, AND H. M. GLAZ, *A second-order projection method for the incompressible Navier–Stokes equations*, J. Comput. Phys., 85 (1989), pp. 257–283.

[3] F. BIANCO, G. PUPPO, AND G. RUSSO, *High-order central schemes for hyperbolic systems of conservation laws*, SIAM J. Sci. Comput., 21 (1999), pp. 294–322.

[4] A. J. CHORIN, *Numerical solution of the Navier–Stokes equations*, Math. Comp., 22 (1968), pp. 745–762.

[5] K. O. FRIEDRICHS AND P. D. LAX, *Systems of conservation equations with a convex extension*, Proc. Nat. Acad. Sci. U.S.A, 68 (1971), pp. 1686–1688.

[6] E. GODLEWSKI AND P. A. RAVIART, *Numerical Approximation of Hyperbolic Systems of Conservation Laws*, Springer, New York, 1996.

[7] S. K. GODUNOV, *A finite difference method for the numerical computation of discontinuous solutions of the equations of fluid dynamics*, Mat. Sb., 47 (1959), pp. 271–290.

[8] A. Harten, *The artificial compression method for computation of shocks and contact discontinuities,* III. *Self-adjusting hybrid schemes*, Math. Comp., 32 (1978), pp. 363–389.

[9] A. Harten, B. Engquist, S. Osher, and S. Chakravarthy, *Uniformly high order accurate essentially non-oscillatory schemes* III, J. Comput. Phys., 71 (1987), pp. 231–303.

[10] G.-S. Jiang, D. Levy, C.-T. Lin, S. Osher, and E. Tadmor, *High-resolution nonoscillatory central schemes with nonstaggered grids for hyperbolic conservation laws*, SIAM J. Numer. Anal., 35 (1998), pp. 2147–2168.

[11] G. S. Jiang and C. W. Shu, *Efficient implementation of weighted ENO schemes*, J. Comput. Phys., 126 (1996), pp. 202–228.

[12] G.-S. Jiang and E. Tadmor, *Nonoscillatory central schemes for multidimensional hyperbolic conservation laws*, SIAM J. Sci. Comput., 19 (1998), pp. 1892–1917.

[13] R. Kupferman and E. Tadmor, *A fast high-resolution second-order central scheme for incompressible flows*, Proc. Nat. Acad. Sci. U.S.A, 94 (1997), pp. 4848-4852.

[14] A. Kurganov, D. Levy, and P. Rosenau, *On Burgers-type equations with nonmonotonic dissipative fluxes*, Comm. Pure Appl. Math., 51 (1998), pp. 443–473.

[15] A. Kurganov and P. Rosenau, *Effects of a saturating dissipation in Burgers-type equations*, Comm. Pure Appl. Math., 50 (1997), pp. 753–771.

[16] A. Kurganov and E. Tadmor, *New high-resolution central schemes for nonlinear conservation laws and convection-diffusion equations*, J. Comput. Phys., 160 (2000), pp. 241–282.

[17] B. van Leer, *Towards the ultimate conservative difference scheme,* V. *A second order sequel to Godunov's method*, J. Comput. Phys., 32 (1979), pp. 101–136.

[18] D. Levy, *A third-order* 2D *central scheme for conservation laws*, in Systém hyperboliques: Nouveaux schémas et nouvelles applications, Vol. 1, Lecture Notes of Écoles CEA-EDF-INRIA, INRIA Rochquencourt, France, 1998, pp. 489–504.

[19] D. Levy, G. Puppo, and G. Russo, *Central WENO schemes for hyperbolic systems of conservation laws*, Math. Model. Numer. Anal., 33 (1999), pp. 547–571.

[20] D. Levy, G. Puppo, and G. Russo, *A third order central WENO scheme for* 2D *conservation laws*, Appl. Numer. Math., 33 (2000), pp. 407–414.

[21] D. Levy, G. Puppo, and G. Russo, *Compact central WENO schemes for multidimensional conservation laws*, SIAM J. Sci. Comput., 22 (2000), pp. 656–672.

[22] D. Levy, G. Puppo, and G. Russo, *Central WENO schemes for multi-dimensional hyperbolic systems of conservation laws*, submitted.

[23] D. Levy, G. Puppo, and G. Russo, *On the behavior of the total variation in CWENO methods for conservation laws*, Appl. Numer. Math., 33 (2000), pp. 415–421.

[24] D. Levy and E. Tadmor, *Nonoscillatory central schemes for the incompressible* 2*-D Euler equations*, Math. Res. Lett., 4 (1997), pp. 1–20.

[25] X.-D. Liu and S. Osher, *Nonoscillatory high order accurate self-similar maximum principle satisfying shock capturing schemes* I, SIAM J. Numer. Anal., 33 (1996), pp. 760–779.

[26] X. D. Liu and S. Osher, *Convex ENO high order multi-dimensional schemes without field by field decomposition or staggered grids*, J. Comput. Phys., 142 (1998) pp. 304–330.

[27] X. D. Liu, S. Osher, and T. Chan, *Weighted essentially nonoscillatory schemes*, J. Comput. Phys., 115 (1994), pp. 200–212.

[28] X. D. Liu and E. Tadmor, *Third order nonoscillatory central scheme for hyperbolic conservation laws*, Numer. Math., 79 (1998), pp. 397–425.

[29] A. A. Medovikov, *High order explicit methods for parabolic equations*, BIT, 38 (1998), pp. 372–390.

[30] H. Nessyahu and E. Tadmor, *Non-oscillatory central differencing for hyperbolic conservation laws*, J. Comput. Phys., 87 (1990), pp. 408–463.

[31] P. L. Roe, *Approximate Riemann solvers, parameter vectors, and difference schemes*, J. Comput. Phys., 43 (1981), pp. 357–372.

[32] C. W. Shu, *Numerical experiments on the accuracy of ENO and modified ENO schemes*, J. Sci. Comput., 5 (1990), pp. 127–149.

[33] C. W. Shu and S. Osher, *Efficient implementation of essentially non-oscillatory shock-capturing schemes*, J. Comput. Phys., 77 (1988), pp. 439–471.

[34] G. Sod, *A survey of several finite difference methods for systems of nonlinear hyperbolic conservation laws*, J. Comput. Phys., 22 (1978), pp. 1–31.

[35] E. Tadmor, *Approximate solutions of nonlinear conservation laws*, in Advanced Numerical Approximation of Nonlinear Hyperbolic Equations, Lecture Notes in Math. 1697, A. Quarteroni, ed., Springer, Berlin, 1998, pp. 1–149.

SIAM J. SCI. COMPUT.
Vol. 22, No. 4, pp. 1489–1510
© 2000 Society for Industrial and Applied Mathematics

A MODIFIED FRACTIONAL STEP METHOD FOR THE ACCURATE APPROXIMATION OF DETONATION WAVES*

CHRISTIANE HELZEL†, RANDALL J. LEVEQUE‡, AND GERALD WARNECKE†

Abstract. The numerical approximation of combustion processes may lead to numerical difficulties, which are caused by different time scales of the transport part and the reactive part of the model equations. Here we consider a modified fractional step method that overcomes this difficulty on standard test problems and allows the use of a mesh width and time step determined by the nonreactive part, without precisely resolving the very small reaction zone. High-resolution Godunov methods are employed and the structure of the Riemann solution is used to determine where burning should occur in each time step. The modification is implemented in the software package CLAWPACK. Numerical results for 1D and 2D detonation waves are shown, including a detonation wave diffracting around a corner.

Key words. combustion, finite volume, fractional step, Godunov methods, stiff source terms

AMS subject classifications. 65M06, 35L65

PII. S1064827599357814

1. Introduction.

We consider hyperbolic systems of conservation laws with source term which arise in the modeling of combustion processes. Here the source term describes a chemical reaction, i.e., a burning process which leads to a production or reduction of physical quantities inside the domain considered. Under appropriate smoothness assumptions the differential form of a system of conservation laws with source term is given as

$$q_t + \sum_{i=1}^{d} f_i(q)_{x_i} = \psi(q), \tag{1.1}$$

where the physical states $q(x,t)$, the fluxes f_i, and the source ψ are vector-valued functions and d is the space dimension.

If the time scale of the ordinary differential equation (ODE) $q_t = \psi(q)$ for the source term is orders of magnitude smaller than the time scale of the homogeneous conservation law $q_t + \sum_{i=1}^{d} f_i(q)_{x_i} = 0$, then the problem is said to be *stiff*. Here we consider a combustion problem, where the chemical reaction, i.e., the burning process, may be much faster than the gas flow.

We restrict our considerations to numerical solutions which are computed by using a fractional step method, in which we alternate between solving the homogeneous conservation law and the ODEs for the reactions. Furthermore, we solve the homogeneous conservation law by using a high-resolution version of the Godunov scheme,

*Received by the editors June 23, 1999; accepted for publication (in revised form) April 21, 2000; published electronically November 8, 2000.

http://www.siam.org/journals/sisc/22-4/35781.html

†Institut für Analysis and Numerik, Otto-von-Guericke Universität Magdeburg, Postfach 4120, 39016 Magdeburg, Germany (christiane.helzel@mathematik.uni-magdeburg.de, gerald.warnecke@mathematik.uni-magdeburg.de). The work of these authors was supported in part by DFG grants Wa 633/7-1, 2 in the priority research programm DANSE and GIF grant I-318-195.06/93.

‡Department of Applied Mathematics and Department of Mathematics, University of Washington, Box 352420, Seattle, WA 98195-2420 (rjl@amath.washington.edu). The work of this author was supported in part by DOE grant DE-FG03-96ER25292 and NSF grants DMS-9505021, DMS-9626645, and DMS-9803442.

which is implemented in CLAWPACK [18]. It is known that the numerical solutions of conservation laws with stiff source terms may be erroneous, for instance, with discontinuities that propagate at the wrong speed and with nonphysical states. These purely numerical problems are caused by the smearing effect of the conservation law solver. A conservative scheme for solving the homogeneous conservation law leads to smeared-out shock profiles while discontinuities are still moving with the correct speed. In a fractional step scheme this smearing effect may lead to an activation of the very fast process described by the source term. This can produce totally nonphysical solutions. Colella, Majda, and Roytburd [11] have observed nonphysical speeds for a simplified combustion problem. LeVeque and Yee [23] have studied incorrect propagation speeds of a contact discontinuity in the numerical solution of an advection equation with a stiff nonlinear source term.

In order to avoid these problems one could use adaptive mesh refinement or front tracking schemes, see, e.g., Bourlioux [9], LeVeque and Shyue [22], Jeltsch and Klingenstein [16]. Using a sufficiently fine mesh it is always possible to avoid nonphysical solutions since the fractional step method is convergent to the relevant solution. Note that a sufficient spatial resolution is as important as a temporal resolution; see [23].

However, resolution of the fine scale is what one really would like to avoid, if one is not interested in the detailed structure of the detonation wave. Therefore some authors, e.g., Pember [25] or Berkenbosch [7] considered the question of whether it is possible to obtain an accurate numerical solution of a stiff hyperbolic system of conservation laws with source term using time and space steps controlled only by the nonstiff part, i.e., without fully resolving the effect of the stiff source term. With such an *underresolved scheme* one would hope to approximate the global solution structure, e.g., the correct propagation speed of a detonation wave.

Note that this approach is not appropriate if one is interested in processes arising on the scale of the reaction zone, i.e., inside the reaction zone, which we do not want to resolve in the stiff case. A correct approximation of pulsating detonation waves or cellular structures can therefore not be obtained by our modified solver. In order to get such results, one has to resolve the reaction zone.

For the combustion problem, the nonphysical numerical solution that is obtained, if the chemical reaction is not resolved on the grid, is a weak detonation wave usually moving with a speed of one mesh cell per time step. Pember [25] postulated that a necessary condition for such a spurious solution to occur in the numerical calculation could be that the temperature of the post detonation state of the approximated spurious weak detonation wave is higher than the ignition temperature. Therefore, this nonphysical numerical solution can be suppressed if the ignition temperature is high enough. In Berkenbosch [7] the dependency of the numerical solution on the ignition temperature is analyzed in much more detail; see also Berkenbosch, Kaasschieter, and Klein [8]. They also show that the critical ignition temperature needed to get an appropriate approximation of the correct strong detonation wave is lower if a high-resolution scheme for the homogeneous conservation law is used in the fractional step scheme. Based on this observation Berkenbosch [7] and Berkenbosch, Kaasschieter, and Klein [8] suggest using a suitable ignition temperature. A similar change of the ignition temperature was also described by Ton; see [28].

Another possibility to approximate shocks as sharp discontinuities (at least in 1D approaches) is supplied by using the Glimm scheme, which was introduced in [14]. This scheme uses the exact values of the Riemann problem at a randomly chosen point inside every mesh cell. The good performance of the Glimm scheme used in

a fractional step approach to solve the reactive Euler equations is demonstrated in Colella, Majda, and Roytburd [11]. A projection step that eliminates intermediate states which are not in equilibrium was introduced by Sjögreen and Engquist [26] and was tested for 1D and 2D problems. The idea of calculating values at randomly chosen points was recently also used by Bao and Jin [2], [3] in a different way. They replaced the ignition temperature by a uniformly distributed random variable inside the interval of the temperature of the completely unburnt state and the completely burnt state.

In Ben-Artzi [4] and Falcovitz and Ben-Artzi [13] the authors considered the approximation of a Chapman–Jouguet (CJ) detonation wave by using an unsplit second order scheme which is based on the solution of generalized Riemann problems. Their scheme was not intended to resolve the stiffness of the problem. Instead it was constructed in a way that should better approximate the coupling between the fluid dynamical and the chemical equations. For the example used in those papers the ignition temperature is above the critical temperature which is needed for a second order fractional step scheme to approximate the correct detonation front. Their numerical results show that the detonation wave is reasonably approximated if an explicit scheme is used to calculate the reduction of unburnt gas, whereas the use of an implicit scheme leads to the unphysical weak detonation wave.

Here we discuss a modification of the fractional step scheme for the approximation of underresolved detonation waves which gives promising numerical resolution for all appropriate ignition temperatures. It turns out that one can get all the information required for this modification from the structure of the Riemann problems occurring in the discretization. We believe that this approach could also give more insight into the numerical problems occurring when solving conservation laws with stiff source terms.

2. The 1D combustion problem.

2.1. The reactive Euler equations. For modeling the combustion process we use the reactive Euler equations as described, for instance, in [12], [15], [20]. We make use of the following basic assumptions. There are only the two chemical states: burnt gas and unburnt gas. The unburnt gas is converted to burnt gas via an irreversible, exothermic chemical reaction described by the function $K(T)$. The reaction rate $K(T)$ depends on the temperature T via an *Arrhenius law* modeled by

$$K(T) = K_0 \exp\left(-E^+/T\right), \tag{2.1}$$

where K_0 is the rate constant and E^+ is the activation energy; see, for instance, Oran and Boris [24]. The reaction rate is negligible for low temperature values and grows exponentially fast if the temperature is high enough. Sometimes this reaction rate is replaced by a discrete *ignition temperature kinetics model* of the form

$$K(T) = \begin{cases} 1/\tau_0 & : \quad T \geq T_{ign}, \\ 0 & : \quad T < T_{ign}, \end{cases} \tag{2.2}$$

where T_{ign} is the ignition temperature and τ_0 is the time scale of the chemical reaction. Note that $1/\tau_0$ roughly corresponds to K_0 in (2.1). In a stiff calculation, where we do not want to resolve the reaction zone, the discrete ignition temperature kinetics model is a reasonable approximation of the Arrhenius law. Therefore, we will first restrict our considerations to this simplified reaction rate. Later we will show that

our modified scheme can also be applied to the more general form of the Arrhenius law (2.1).

The value τ_0 in (2.1) must be seen in relation to the time scale of the convective part of our problem. Further we assume that the burnt and unburnt gases are ideal gases with the same ratio of specific heats γ and temperature $T = p/\rho\mathcal{R}$, where $\mathcal{R}$ is the specific gas constant. Finally we ignore the effects of diffusion. Using all of these assumptions we get the model equations

$$\rho_t + (\rho u)_x = 0 \qquad \text{conservation of mass,} \tag{2.3}$$

$$(\rho u)_t + (\rho u^2 + p)_x = 0 \qquad \text{conservation of momentum,} \tag{2.4}$$

$$E_t + (u(E+p))_x = 0 \qquad \text{conservation of energy,} \tag{2.5}$$

$$(\rho Z)_t + (\rho u Z)_x = -K(T)\rho Z \qquad \text{continuity equation for the unburnt gas,} \tag{2.6}$$

as a combination of the Euler equations with the kinetics model. The variable Z is the mass fraction of unburnt gas, where $Z = 1$ describes the unburnt state and $Z = 0$ the completely burnt state. The other variables are as usual the total density of burnt and unburnt gas ρ, the velocity u, the pressure p, and the total energy E. The total energy is calculated via the equation of state

$$E = \frac{p}{\gamma - 1} + \frac{1}{2}\rho u^2 + q_0 \rho Z, \tag{2.7}$$

where q_0 is the heat release and the term $q_0\rho Z$ is the chemical energy. Note that by using (2.3), (2.6) is equivalent to

$$Z_t + uZ_x = -K(T)Z. \tag{2.8}$$

There are two combustion processes which are associated with the reactive Euler equations, namely, detonations and deflagrations. Here we restrict our considerations to the approximation of detonation processes; see, for instance, Courant and Friedrichs [12], Godlewski and Raviart [15], or Berkenbosch [7] for a characterization of these processes. We will consider the special case of a CJ detonation wave, which is the detonation wave that occurs most frequently in practice.

The solution structure that can be derived from the reactive Euler equations was originally considered by Zel'dovich, von Neumann, and Döring and is therefore called a *ZND structure*. The detonation process is initiated by a shock. Through this shock, pressure, density, and temperature are raised instantaneously. If the temperature of the unburnt gas becomes greater than the ignition temperature, then the combustion is initiated. Through the combustion process, pressure and density are decreased; see Figure 1. For numerical experiments it is useful to start with such an exact ZND structure as initial data. In this case the exact traveling wave solution and especially the exact propagation speed of the detonation wave can be calculated.

We say that the reactive Euler equations for approximating a detonation wave are *stiff* if the reaction zone is small relative to the mesh size $\triangle x$, in which case the ZND structure cannot be resolved on the grid. This is consistent with the characterization of stiffness given in the introduction because the time scale of the chemical reaction τ_0 is proportional to the length of the reaction zone. In this case the time step appropriate for the fluid dynamics is also insufficient to resolve the kinetics.

2.2. Numerical solutions by using the fractional step method. We use a fractional step method to calculate the numerical solution, e.g., we split the calculation

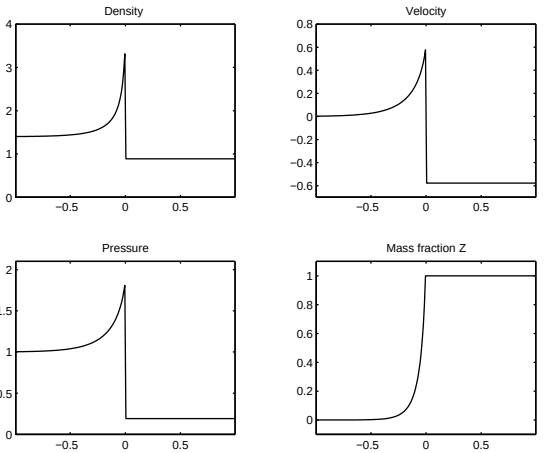


FIG. 1. *ZND structure of a CJ detonation wave.*

of (1.1) into the subproblems

$$q_t + f(q)_x = 0, \tag{2.9}$$

$$q_t = \psi(q) \tag{2.10}$$

and alternate between evolving these numerically. Let $L_{CL}^{\triangle t}$ denote an approximate solution operator of the conservation law (2.9) and $L_{ODE}^{\triangle t}$ be an approximate solution operator of the ODE (2.10), each over a time step of length $\triangle t$. Then the numerical solution at time step t_{n+1} is calculated from the numerical solution at time t_n via the relation

$$Q^{n+1} = L_{ODE}^{\triangle t} L_{CL}^{\triangle t} Q^n.$$

Here, we will restrict our considerations to this "Godunov splitting" scheme, but the results can also be extended to other fractional step methods, e.g., the Strang splitting scheme. See, for instance, LeVeque [20] for more details on fractional step methods. We use a high-resolution version of the Godunov scheme with an exact Riemann solver for solving the system of conservation laws. The solution of the Riemann problem for the nonreactive two-component Euler equations, i.e., the solution of (2.3)–(2.6) with $K(T) = 0$ and piecewise constant initial values, is very similar to the solution of the Riemann problem for the Euler equations. The solution theory of this frequently-used Riemann problem is well known; see, e.g., Smoller [27] or Kröner [17]. This solution consists of constant states separated by three waves, where the 2-wave is always a contact discontinuity. The quantities u and p are Riemann invariants of the 2-wave, thus they are constant across the contact discontinuity. The 1- and the 3-wave are shock or rarefaction waves. Further it is known that the characteristic speed of the contact discontinuity is equal to u. The mass fraction of unburnt gas in the homogeneous part of (2.8) is advected with the speed u, i.e., with the speed of the contact discontinuity. Therefore the mass fraction of unburnt gas in a Riemann problem is equal to Z_l everywhere to the left of the contact discontinuity and equal to Z_r everywhere to the right-hand side of this 2-wave. Different iterative schemes for solving the Riemann problem for the Euler equations were developed, for instance, the Chorin

algorithm [10]. The latter can easily be extended to solving the Riemann problem of the nonreactive two-component Euler equations. It was used in our calculations.

The second step in the fractional step scheme consists in solving the ODE for the source term. The assumption that $K(T)$ is constant is permissible for the ignition temperature kinetics model (2.2). Therefore, the ODE $Z_t = -K(T)Z$ can be solved exactly. We get the solution

$$Z_j^{n+1} = \exp\left(-K(\overline{T}_j^n)\triangle t\right)\overline{Z}_j^n,$$

where $\overline{Z}_j^n$ and $\overline{T}_j^n$ are the values after one time step $\triangle t$ of the conservation law solver. We consider the following test problem of a CJ detonation wave moving with speed one.

Example 1. The initial values consist of totally burnt gas on the left-hand side and totally unburnt gas on the right-hand side. The density, velocity, and pressure of the burnt gas are given by $\rho_b = 1.4$, $u_b = 0$, $p_b = 1$, and $Z_b = 0$. It is then possible to calculate the values for the unburnt state so that the states are connected by a CJ detonation wave moving with speed 1. We find that $\rho_u = 0.887565$, $u_u = -0.577350$, and $p_u = 0.191709$. The mass fraction of unburnt gas is $Z_u = 1$, which means that there is only unburnt gas. Further we can calculate the states within the reaction zone. The other parameters are set to $\gamma = 1.4$, $\mathcal{R} = 1$, and $q_0 = 1$.

For our first set of tests we use the discrete ignition temperature kinetics reaction rate model (2.2). The time scale of the chemical reaction τ_0 as well as the ignition temperature T_{ign} vary in the numerical calculations considered. We consider two values of τ_0 for our experiments: $\tau_0 = 10^{-6}$, in which case the problem is stiff on all grids we consider, and the nonstiff case $\tau_0 = 0.1$. In Figure 1 the ZND structure is shown for the nonstiff case, and these values are also used as initial data. The temperature is increased by the shock. The wave propagates to the right into unburnt gas and the temperature directly behind the fluid dynamical shock is called the *von Neumann temperature*, T_{vN}. The ignition temperature has to be higher than T_u, i.e., higher than the temperature of the unburnt gas, and lower or equal to T_{vN}. In our example the von Neumann temperature is $T_{vN} = 0.545918$ and the temperature of the unburnt gas is $T_u = 0.21598$. Note that the same initial value problem was considered in LeVeque [20] where some preliminary results with this approach were presented.

Now we set $\tau_0 = 10^{-6}$ and $T_{ign} = 0.22$, i.e., the problem is stiff and the temperature is only slightly higher than the temperature of the unburnt gas. In this case the fractional step scheme produces a nonphysical solution, a discontinuity which is propagating with a speed of one mesh cell per time step as mentioned in the introduction; see Figure 2.

This phenomenon has been studied by several authors, see, for instance, [8], [9], [11], [13], and it is well understood that it is caused by the numerical diffusion of the conservation law solver feeding into the fractional step method. For understanding this purely numerical problem we consider one time step using the fractional step method to approximate a detonation wave in a very stiff case, i.e., the reaction zone is much smaller than the width of one mesh cell. For this stiff case the Riemann problem, consisting of totally unburnt gas at the right-hand side and totally burnt gas at the left-hand side, would be a reasonable approximation of the ZND structure. Now we apply one time step of the Godunov scheme to this Riemann problem. The solution structure of the Riemann problem considered is indicated in Figure 3.

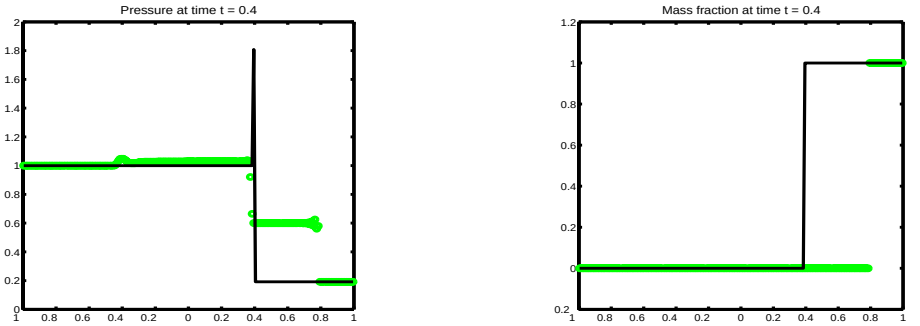


FIG. 2. *Exact* (− − −) *and numerical* (ooo) *solution using the classical fractional step method, with high-resolution Godunov scheme. Parameter values:* $\tau_0 = 10^{-6}$, $T_{ign} = 0.22$, $\triangle x = 0.01$, *and* $\triangle t = 0.005$, *giving a Courant number between* 0.5 *and* 1.

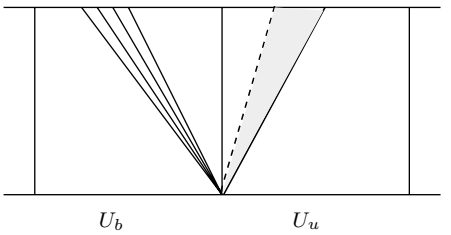


FIG. 3. *Structure of the Riemann solution for the reactive Euler equations. From left to right:* 1*-rarefaction wave,* 2*-contact discontinuity,* 3*-shock; left state: completely burnt gas, right state: unburnt gas.*

The rightmost wave (3-wave) is a shock moving into the unburnt gas. The 2-wave denoted by a dashed line is the contact discontinuity separating burnt gas to the left from initially unburnt gas to the right. However, the shock increases the temperature of the unburnt gas in the region between the 3-wave and 2-wave (the shaded region in Figure 3). The temperature is now greater than T_{ign} and this gas will burn. In the stiff case it burns completely during this time step.

The incorrect propagation speed in the fractional step method arises from the fact that the structure shown in Figure 3 is first averaged over the grid cells to create new piecewise constant states before the reaction terms are applied. As a result the entire grid cell to the right in Figure 3 obtains a single averaged temperature. If this is greater then T_{ign} then *all* the unburnt gas in this cell burns during the reaction step, including the gas to the right of the 3-wave. This causes the interface between burnt and unburnt gas to advance by one full grid cell. See [23] for more details. The closer T_{ign} is to T_u the more likely this is to happen. If T_{ign} is considerably larger than T_u, then in some steps the full cell will burn and in other steps nothing will burn, and the average speed could still come out close to correct. Furthermore, if the ignition temperature is high enough then an unphysical intermediate state (for instance in pressure) lower than the value of the completely burnt state can not be obtained. This has been investigated by Berkenbosch [7]; see also [8].

2.3. Modification of the fractional step method. We have seen that the nonphysical solutions are a consequence of smearing the reaction zone before burning. Using this observation we construct a modification of the fractional step scheme which

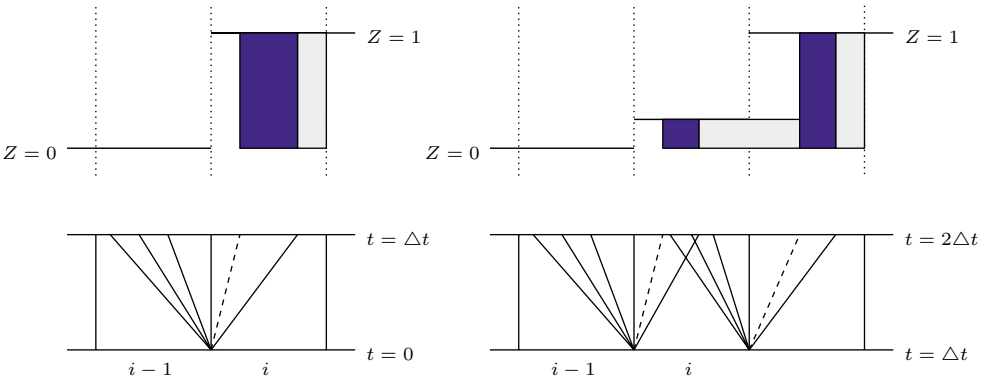


FIG. 4. *Modification of the scheme along the smeared-out reaction front in the stiff case. Left: first time step; right: second time step.*

eliminates those combustion processes which are purely a consequence of the smearing effects. First we consider the stiff case, in which the exact reaction zone is smaller than one mesh cell and the reactions take place very fast.

Again considering the situation which is shown in Figure 3, we should allow a reaction only between the shock and the contact discontinuity in the first time step, because only there do sufficiently high temperature and unburnt gas coexist. The essential idea of our approach is to apply the ODE solver only in the region where burning should occur. In the first time step this is easy, but after one time step we do expect some smearing of the correct shock location and hence a nonphysical "average" temperature in at least one grid cell. Now we want to interpret the cell average in a different way. As shown in Figure 4 the piecewise constant initial values for the second time step contain one mesh cell i, where the mass fraction of unburnt gas Z has a value with $0 < Z < 1$. The average of the temperature in cell i might be higher than the ignition temperature, but more correctly the burnt gas in cell i has a temperature higher than T_{ign} and the unburnt gas has a temperature lower than T_{ign}.

In the stiff case the reaction takes place very fast and during one time step the whole portion of the gas which was heated up by the shock has already burnt. Therefore we interpret the gas flow described by the homogeneous conservation law in the second time step as a transport of the unburnt gas. A reaction can only take place if this unburnt gas is ignited by a shock. This modification is described in Figure 4. The shaded areas indicate the mass fraction of unburnt gas after the transport by the homogeneous conservation law. Only in the dark shaded area a reaction is initiated. We make use of the fact that a rarefaction wave, such as the one generated between cells i and $i + 1$, cannot increase the temperature of the unburnt gas. In consequence there is no reaction to the right of the 3-shock in cell i even though the *average* value of the temperature might be higher than T_{ign}. The solution structure could also consist of a 1-shock and a 3-shock. In this case we allow a reaction between these two waves.

The temperature as well as the pressure can only be increased by a shock. We observe that along the smeared-out reaction front the Riemann problems occurring in each time step for solving the homogeneous conservation law have a 3-shock if the combustion front moves from left to right.

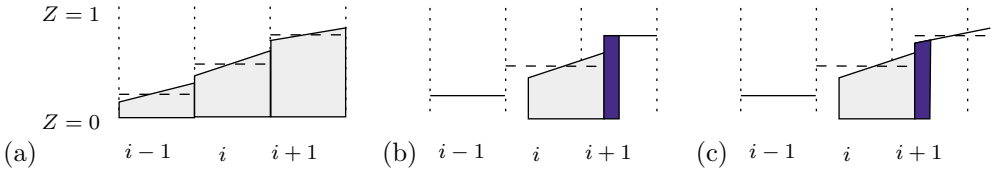


FIG. 5. *Description of the modification using the high-resolution wave propagation algorithm.*

On the other hand, a mesh cell may also lie within the reaction zone. If the temperature in a mesh cell is higher than the ignition temperature and there is still unburnt gas but no shock wave is coming into this mesh cell that will further increase the temperature there, then this cell belongs to the (smeared-out) reaction zone. In this case we allow the burning process in the whole mesh cell; i.e., we use the classical fractional step method.

For our example of a CJ detonation wave moving from the left- to the right-hand side, we can easily determine whether a mesh cell belongs to the smeared-out shock or to the reaction zone and, hence, whether the burning process in the stiff case should take place between shock and contact discontinuity, or in the whole mesh cell by only considering the 3-wave. If the 3-wave is a shock, then we allow the burning process only between the shock and contact discontinuity, and if the 3-wave is a rarefaction wave, then we allow the burning process in the whole mesh cell.

By using a second order finite volume scheme, sharper results can be obtained in which discontinuities are smeared-out over fewer mesh cells than when using first order schemes. Nevertheless, in combination with a fractional step scheme, the same numerical problems occur for both first and second order approximations of the conservation laws if the source term is not treated carefully. Note that the numerical results which are shown in Figure 2 were calculated with a high-resolution Godunov scheme to approximate the convective part. The same nonphysical solution would be obtained using the first order Godunov scheme. For our improved numerical calculations we have combined the high-resolution wave propagation algorithm from CLAWPACK, which is described in LeVeque [21], with the modification of the fractional step scheme described above.

The high-resolution method is based on using piecewise linear reconstruction in place of piecewise constant functions. Slopes are chosen based on nearby solution values, and limiters are applied to avoid spurious oscillations. For all calculations shown in this paper we used the monotonized-centered limiter, proposed by van Leer [29] and given explicitly in [21] in the context of CLAWPACK. Other standard limiters, e.g., minmod or superbee, which are also available in CLAWPACK were also tested and gave comparable results. Figure 5(a) shows the piecewise linear reconstruction of Z in three grid cells. From the structure of the Riemann problem we know that discontinuities in Z are propagated at the speed of the contact discontinuity. In the high-resolution method each piecewise linear structure is propagated at the local fluid speed and then averaged onto the grid; see [21]. With combustion, we again wish to apply burning only to the unburnt gas lying between a shock and contact where the temperature has been raised above the ignition temperature. This is illustrated in Figure 5(b) and (c) for the case where such a shock arises from the Riemann problem between cells i and $i+1$ and moves into cell $i+1$ at a speed greater than the contact speed. Hence, it is the unburnt gas initially in cell $i+1$ which now burns, as indicated by the dark shaded region in each figure. In Figure 5(b) we simply use the cell average

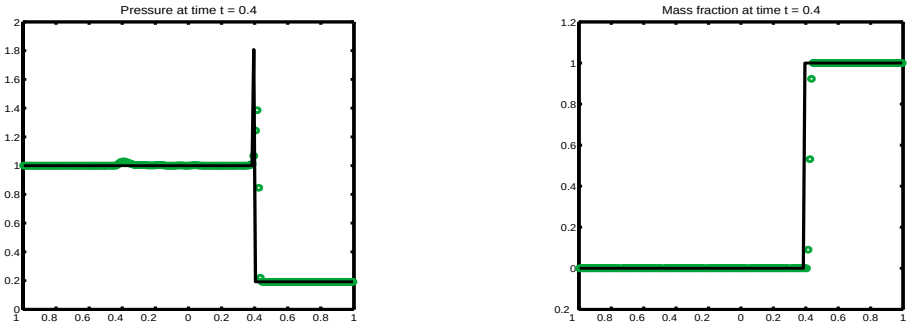


FIG. 6. *Exact and numerical solution using the modification, with high-resolution Godunov scheme. Parameter values:* $\tau_0 = 10^{-6}$ *and* $T_{ign} = 0.22$, $\triangle x = 0.01$, $\triangle t = 0.005$ *giving a Courant number between* 0.5 *and* 1.

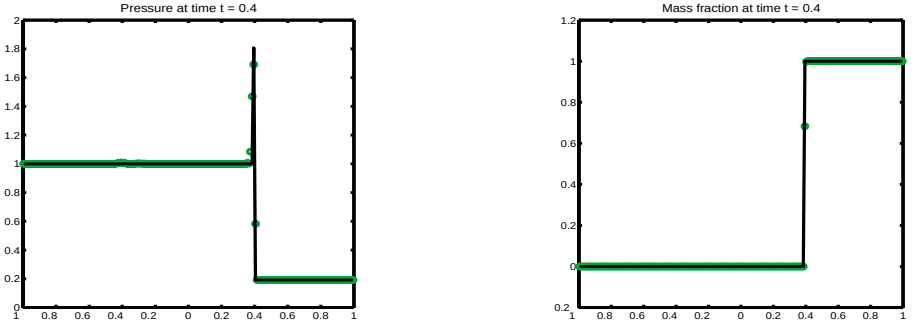


FIG. 7. *Exact and numerical solution using the modification, with high-resolution Godunov scheme. Parameter values:* $\tau_0 = 10^{-6}$ *and* $T_{ign} = 0.54$, $\triangle x = 0.01$, $\triangle t = 0.005$ *giving a Courant number between* 0.5 *and* 1.

Z_{i+1} to determine how much gas burns. This is easier to implement than the more accurate procedure indicated in Figure 5(c), where the piecewise linear structure in cell $i + 1$ is used to determine the amount that burns. Both have been tested in one dimension but we have found very little difference in the observed accuracy. Consequently, we have used the simpler approach of Figure 5(b) in the 2D extension presented below.

Figure 6 shows the numerical solution of the stiff problem with low ignition temperature using the modification which was described above. One should compare with Figure 2 where the same problem was solved by using the classical fractional step method. In Figure 7 the stiff problem was solved with our modification for a case with a higher ignition temperature, near the von Neumann temperature, in order to show that we get a good approximation of the detonation wave for all possible ignition temperatures.

In the nonstiff case the fractional step method gives a good approximation of the detonation wave, since for this case the two time scales fit together; see LeVeque [20] for a numerical calculation. This is consistent with our modification of the stiff case in the following sense. In the nonstiff case most of the mesh cells with $0 < Z < 1$ belong to the reaction zone and therefore a reaction should take

TABLE 1

Numerical propagation speed, calculated taking (2.11)*, using the classical fractional step method* ($S_{fs,1}$ *or* $S_{fs,2}$) *and the modified method* ($S_{mod,1}$ *or* $S_{mod,2}$) *with first-order and high-resolution versions of the Godunov scheme for different ignition temperatures* (T_{ign}) *and different time scales* τ_0*. The numerical propagation speed is given for the time* $t = 0.4$*. For all calculations we used the mesh width* $\triangle x = 0.01$ *and a time step* $\triangle t = 0.005$ *giving a Courant number between* 0.5 *and* 1*.*

	τ_0	T_{ign}	$S^{0.4}_{fs,1}$	$S^{0.4}_{mod,1}$	$S^{0.4}_{fs,2}$	$S^{0.4}_{mod,2}$
	0.1	0.22	1.052262	1.045323	1.052892	1.044059
Nonstiff	0.1	0.30	1.038623	1.032534	1.047261	1.042274
	0.1	0.40	1.029742	1.029659	1.039676	1.039266
	0.1	0.54	0.985936	0.985932	1.009614	1.009614
	10^{-6}	0.22	2.025000	1.069491	2.000000	1.086376
Stiff	10^{-6}	0.30	1.150000	1.059374	1.125029	1.083180
	10^{-6}	0.40	1.078079	1.043682	1.102789	1.052685
	10^{-6}	0.54	1.025647	1.004062	1.056581	1.007961

place in the whole mesh cell. Now, in contrast to the stiff case, the gas that was heated by the shock did not entirely burn in one time step. Note that this was our main motivation for the modification along the smeared-out reaction front in the stiff case. Therefore, in the nonstiff case we do not restrict a reaction to the region between the shock and contact discontinuity only. Only in the mesh cell where the totally unburnt gas is first heated up by a shock do we need to make an exception. There the reaction should be restricted to the area behind the shock because only there we have a sufficiently high temperature. The numerical results which where obtained with this modification are very similar to those obtained by the classical fractional step method, which is already adequate for the nonstiff case, as seen in Table 1. Note that the approximation of the nonstiff case is also consistent with the transition between the stiff and the nonstiff method which is described below.

Table 1 contains different values for the numerical propagation speed $S^{n\triangle t}$ of the mass fraction of unburnt gas calculated by the formula

$$S^{n\triangle t} = \frac{\triangle x}{n\triangle t} \cdot \sum_{i=-\infty}^{\infty} (Z_i^0 - Z_i^n). \tag{2.11}$$

$S^{n\triangle t}$ is the averaged numerical propagation speed of the detonation wave at time $n\triangle t$, which was calculated via an approximation of the equal area rule; see Berkenbosch [7]. Both a nonstiff and a stiff case are considered with different values of T_{ign}. The correct value of the propagation speed is equal to one in all cases. The index of S in Table 1 further specifies the numerical scheme which was used for the calculation as well as the order. All values of the numerical propagation speed are given for the time $n\triangle t = 0.4$, which corresponds to the time step of the numerical approximations shown in Figure 2 as well as Figures 6 and 7.

Table 1 shows that the classical fractional step scheme gives a good approximation of the stiff problem if the ignition temperature T_{ign} is high enough. This was studied by Berkenbosch in [7] and Berkenbosch, Kaasschieter, and Klein in [8]. With the modified fractional step scheme we always get a more accurate approximation of the shock speed and maintain reasonable accuracy even in the stiff case.

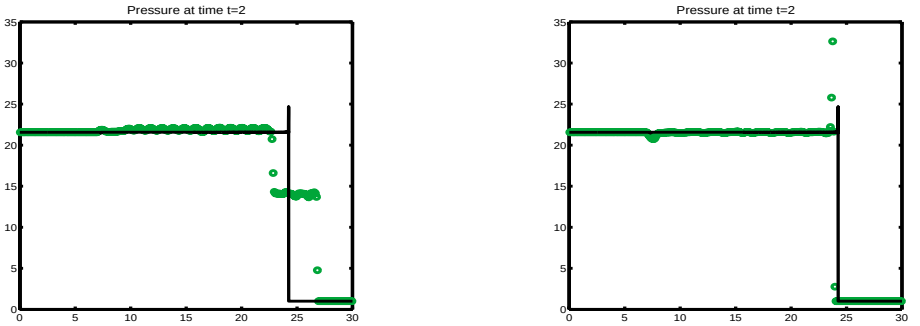


FIG. 8. *Resolved* (− − −) *and underresolved* (ooo) *approximation of the pressure for Example 2. For the resolved calculation we used the mesh width* $\triangle x = 0.0025$*, and for the underresolved calculation we used* $\triangle x = 0.1$*. The classical fractional step scheme gives an unphysical solution (left); the modified fractional step scheme gives an accurate approximation (right).*

In our next example we apply our modified fractional step scheme to the approximation of a stiff detonation wave for which the reaction is modeled by the Arrhenius law (2.1).

Example 2. We consider the reactive Euler equations with the reaction rate law (2.1). The unburnt state is given by $\rho_u = 1$, $u_u = 0$, $p_u = 1$. For the heat release, we used the value $q_0 = 25$ and the activation energy is set to $E^+ = 25$. Furthermore, the ratio of specific heats is set to $\gamma = 1.4$. The unburnt state is connected to the burnt state by a CJ detonation wave. The physical quantities of the completely burnt state are $\rho_b = 1.6812$, $u_b = 2.8867$, $p_b = 21.5672$. The CJ detonation wave has the speed $s_{CJ} = 7.1247$. Initially the discontinuity is located at the point $x = 10$. We use the rate constant $K_0 = 164180$.

Figure 8 shows numerical results of the pressure for underresolved calculations of the detonation wave described in Example 2. The solid line is the reference solution which was calculated on a very fine mesh. Our modified fractional step scheme, which can be applied in the same way as described for the ignition temperature kinetics model, approximates the correct propagation speed of the detonation wave, whereas the usual fractional step scheme again leads to an unphysical weak detonation wave followed by a nonreactive shock.

In order to approximate the transition between the stiff and the nonstiff case on those mesh cells which approximate the smeared-out leading shock of a detonation wave in an appropriate way, we limit the amount of the mass fraction of unburnt gas, which is converted to burnt gas during one time step, from above by $(\triangle Z_i)_{max} = \text{area } Z_i$, where Z_i is the cell average of the mass fraction of unburnt gas and area specifies the part of the mesh cell, where the source term should be applied in a stiff calculation. In the 1D case this part of the mesh cell is given by $\text{area} = \frac{(s-c)\triangle t}{\triangle x}$, where s is the speed of the shock and c is the propagation speed of the contact discontinuity. This implies that along the smeared-out leading shock of a detonation wave, *at most* the whole mass fraction of unburnt gas between shock and contact discontinuity would completely burn. In the nonstiff case the reduction of the mass fraction of unburnt gas due to the ODE for the source term equation applied to the whole cell average will in general be less than $(\triangle Z_i)_{max}$, and our modified fractional step scheme automatically switches to the classical fractional step scheme along the smeared-out leading shock of a detonation wave. The same criteria can be applied to calcula-

tions which use the Arrhenius law (2.1). However, for this reaction rate equation the reaction rate and also the length of the reaction zone depends on the temperature. Therefore, especially in multidimensional examples, the problem can include stiff and nonstiff regions depending on the temperature. We will consider such a problem in section 3.

3. The 2D combustion problem. Now we consider the modification of the scheme for 2D reactive Euler equations, i.e., for the system of equations

$$\rho_t + (\rho u)_x + (\rho v)_y = 0, \tag{3.1}$$

$$(\rho u)_t + \left(\rho u^2 + p\right)_x + (\rho uv)_y = 0, \tag{3.2}$$

$$(\rho v)_t + (\rho uv)_x + \left(\rho v^2 + p\right)_y = 0, \tag{3.3}$$

$$E_t + (u(E+p))_x + (v(E+p))_y = 0, \tag{3.4}$$

$$(\rho Z)_t + (\rho uZ)_x + (\rho vZ)_y = -\rho K(T)Z. \tag{3.5}$$

Here u is the velocity of the gas in x-direction and v is the y-component of the velocity. The equation of state is in the 2D case

$$E = \frac{p}{\gamma - 1} + \frac{1}{2}\rho(u^2 + v^2) + q_0\rho Z.$$

Again we want to use a fractional step scheme for approximating the solution. As in the 1D case we have to apply a modification in order to avoid both nonphysical propagation speeds of the combustion front and wrong intermediate states. For the solution of the homogeneous conservation law we used the high-resolution version of the Godunov scheme implemented in CLAWPACK [18]. The method is based on solving 1D Riemann problems at each cell boundary as well as taking transverse directions into consideration. The part of the flux across a cell boundary which is propagated in the transverse direction is calculated via the tangential Riemann problem based on a Roe linearization; see LeVeque [21]. This improves the stability and accuracy of the scheme.

3.1. Modification of the 2D fractional step method. As we have already mentioned, the 2D version of the Godunov scheme is based on solving 1D Riemann problems in the x- as well as in the y-direction. Therefore, we can use the same modification as in the 1D case; i.e., in the stiff case along the reaction front a reaction will only be possible between the shock and contact discontinuity. This modification is indicated in Figure 9, where we assume the case of a 3-shock.

Now we also want to consider the influence of the transverse propagation. This means that parts of the shaded areas in Figure 9 are propagated into other mesh cells. Here we only want to consider the propagation in the y-direction of the reaction area calculated by a Riemann problem in x-direction. From the transverse Riemann solver we can get a decomposition of each wave into a linear combination of eigenvectors of the Jacobian matrix of the flux functions in the tangential direction. These subwaves are moving upwards or downwards with the speeds μ^i corresponding to the eigenvalues of the Jacobian matrix of the form $\mu^1 = v - c$, $\mu^2 = v$, $\mu^3 = v + c$; see [21].

The most accurate possibility for calculating the transverse propagation of the area where a reaction is supposed to take place would require calculating the temperature for all the subwaves corresponding to the area behind a shock along the reaction zone. If the temperature is higher than the ignition temperature, then the

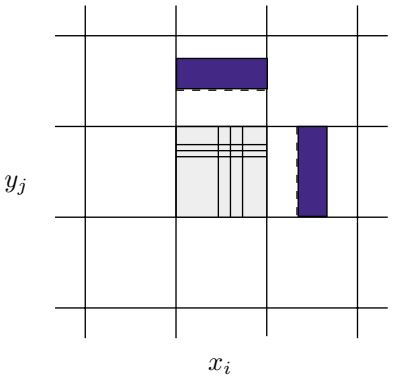


FIG. 9. *Structure of the modification of the normal Riemann problem in x direction between the states $q(x_i, y_j)$ and $q(x_{i+1}, y_j)$ as well as for the Riemann problem in y direction between the states $q(x_i, y_j)$ and $q(x_i, y_{j+1})$ used in the stiff case.*

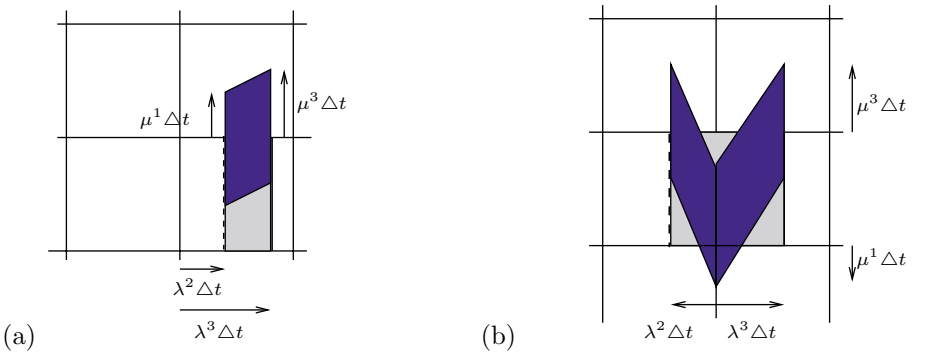


FIG. 10. *Two different possibilities for the transverse modification. The dark shaded regions are the cell portions where a reaction takes place for a Riemann problem in the x-direction with transverse modification.*

area between the shock and contact discontinuity should be propagated in the transverse direction. Such a modification would be quite expensive. For the transverse propagation, it is also not necessary to decouple every wave separately. Instead of decoupling each wave, only the left- and right-going flux differences have to be considered. Therefore, we have used the simplified transverse propagation of cell areas where a reaction takes place as indicated in Figure 10 which only uses the transverse speeds μ^1 and μ^3. If both of these eigenvalues have the same sign, then the area which is propagated in the y direction is equal to the trapezoid with a height equal to the distance between shock and contact discontinuity and the sides $|\mu^1 \triangle t|$ and $|\mu^3 \triangle t|$. This is shown in Figure 10(a) for the case where all eigenvalues are positive. In the case where $\mu^1 < 0$ and $\mu^3 > 0$, we would have a triangular portion corresponding to μ^1, which is moving into the cell below, and another triangular portion moving into the cell above, which corresponds to the eigenvalue μ^3. Figure 10(b) shows yet another possibility: the transverse propagation of the reaction area consisting of a left-going contact discontinuity and a right-going 3-shock assuming also that the eigenvalues μ^1 and μ^3 have different signs.

3.2. Approximation of a radialsymmetric CJ detonation wave. We restrict our considerations to the stiff case where the fractional step scheme may have numerical problems. First we want to note that also in the 2D case the usual fractional step scheme without modification leads to a good approximation if the ignition temperature is high enough. This was observed by Berkenbosch, Kaasschieter, and Klein [8] for 1D and 2D combustion waves.

Example 3. For our numerical computations we consider a radially symmetric CJ detonation wave. The initial values consist of totally burnt gas inside of a circle with radius 0.3 and totally unburnt gas everywhere outside of this circle. Furthermore, the unburnt and burnt states are chosen in a way analogous to the 1D case, i.e., $\rho_b = 1.4$, $\rho_u = 0.887565$, $p_b = 1$, $p_u = 0.191709$, $Z_b = 0$, $Z_u = 1$, $u_b = 0$, $u_u = -0.577350\cos\alpha$, $v_b = 0$, $v_u = -0.577350\sin\alpha$, where α is the angle in polar coordinates. The ignition temperature is set to $T_{ign} = 0.26$.

For this radially symmetric problem it is possible to get more insight into the structure of the solution by comparing the solution calculated with a 2D algorithm and the numerical solution of the 1D system of reactive Euler equations with an additional source term for the radial symmetry. The 1D reactive Euler equations for a radially symmetric problem are

$$\rho_t + (\rho\hat{u})_r = -\frac{1}{r}\rho\hat{u},$$

$$(\rho\hat{u})_t + (\rho\hat{u}^2 + p)_r = -\frac{1}{r}\rho\hat{u}^2,$$

$$E_t + (\hat{u}(E+p))_r = -\frac{1}{r}\hat{u}(E+p),$$

$$Z_t + \hat{u}Z_r = -K(T)Z,$$

where $\hat{u} = u\cos(\alpha) + v\sin(\alpha)$ is the speed in radial direction and $r = \sqrt{x^2 + y^2}$ is the distance from the center. Now we can use the 1D fractional step scheme with the modification for the stiff source term to get an approximation along any radial slice of the 2D combustion wave. The 1D system is solved on a very fine grid ($\triangle x = 0.00025$) so that the numerical solution is assumed to be an accurate approximation of the exact solution. The numerical solutions of the 1D reference problem are plotted as solid lines in Figures 14–16. The scatter plots of the 2D solutions are obtained by plotting the value on each mesh cell as a function of r, i.e. the distance to the center of symmetry.

In the 2D case we have the additional effect that also the gas flow ahead of the combustion front increases the temperature slightly. For simplicity, we are not interested in this effect for the moment. Therefore, we have to choose the ignition temperature higher than the temperature occurring just ahead of the combustion front during the time interval considered.

Figures 11 and 12 as well as the scatter plots Figures 14 and 15 show the numerical calculations at different time steps obtained using the modified fractional step scheme. Here the combustion front moves with the correct speed and, furthermore, the circular geometry is resolved well on the grid.

For contrast, in Figure 13 contour plots of the pressure and mass fraction of unburnt gas are shown, calculated by using the classical fractional step scheme at the

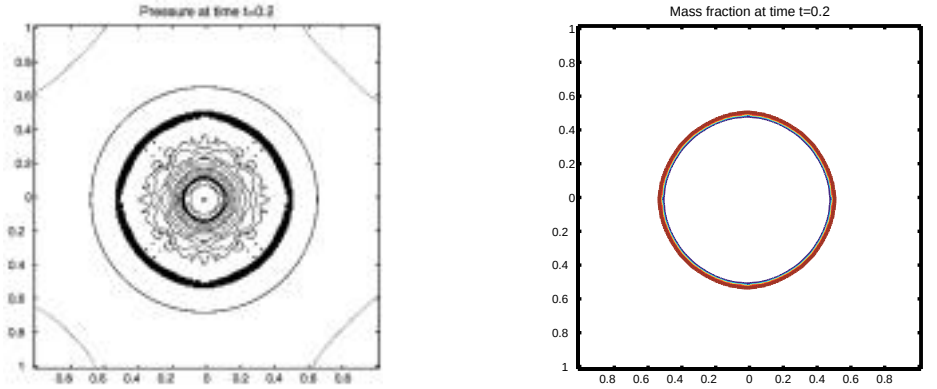


FIG. 11. *Contour plot of pressure and mass fraction of unburnt gas at time* $t = 0.2$ *using the modification with high-resolution Godunov scheme,* $T_{ign} = 0.26$, $\tau_0 = 10^{-6}$, $\triangle x = \triangle y = 0.01$, $CFL \leq 0.75$.

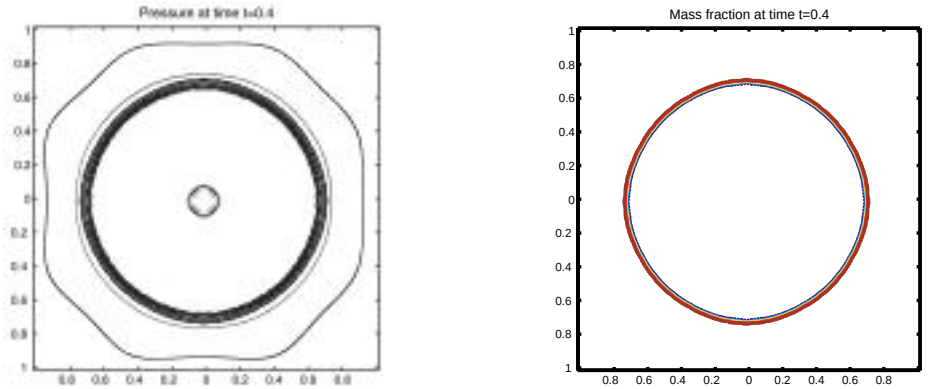


FIG. 12. *Contour plot of pressure and mass fraction of unburnt gas at time* $t = 0.4$ *using the modification with high-resolution Godunov scheme,* $T_{ign} = 0.26$, $\tau_0 = 10^{-6}$, $\triangle x = \triangle y = 0.01$, $CFL \leq 0.75$.

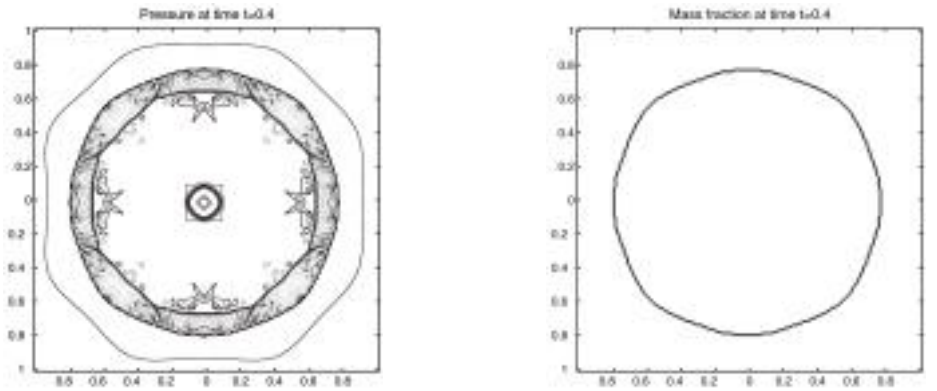


FIG. 13. *Contour plot of pressure and mass fraction of unburnt gas at time* $t = 0.4$ *using the classical fractional step method with high-resolution Godunov scheme,* $T_{ign} = 0.26$, $\tau_0 = 10^{-6}$, $\triangle x = \triangle y = 0.01$, $CFL \leq 0.75$.

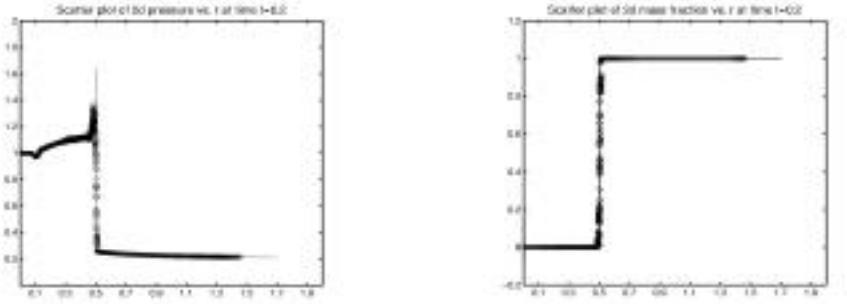


FIG. 14. *Scatter plot of pressure and mass fraction of unburnt gas vs. radius r at time* $t = 0.2$ *using the 2D modification with high-resolution Godunov scheme;* $\triangle x = \triangle y = 0.01$, $\triangle t = 0.005$, $\tau_0 = 10^{-6}$, $T_{ign} = 0.26$.

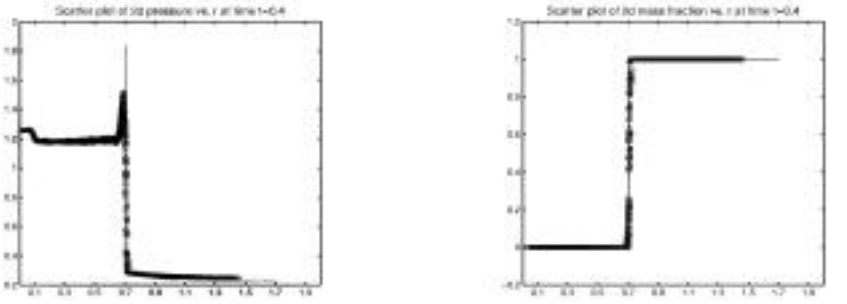


FIG. 15. *Scatter plot of pressure and mass fraction of unburnt gas vs. radius r at time* $t = 0.4$ *using the 2D modification with high-resolution Godunov scheme;* $\triangle x = \triangle y = 0.01$, $\triangle t = 0.005$, $\tau_0 = 10^{-6}$, $T_{ign} = 0.26$.

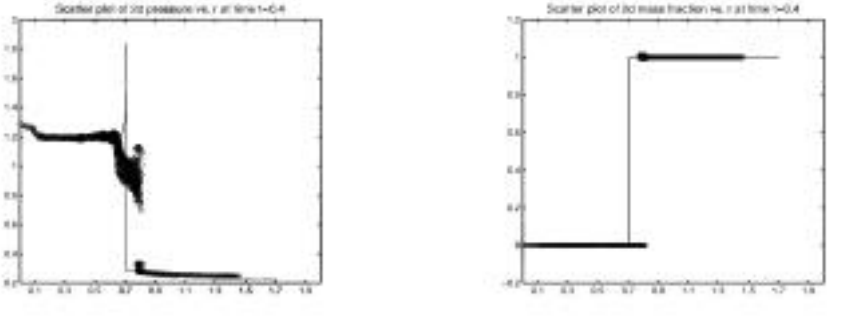
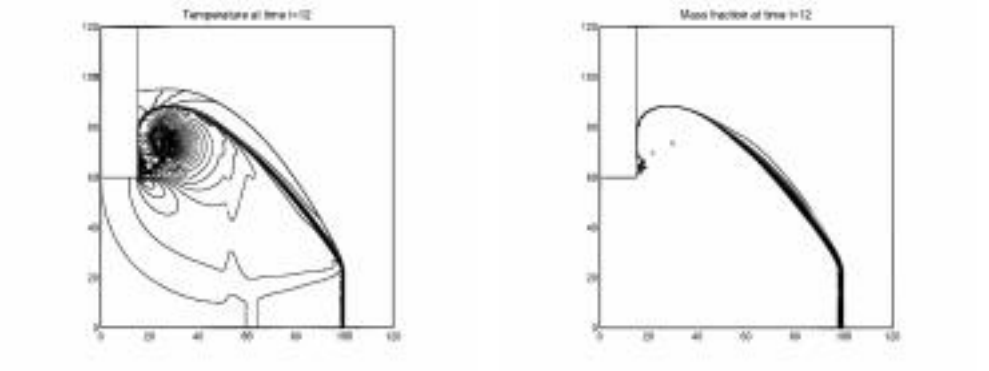


FIG. 16. *Scatter plot of pressure and mass fraction of unburnt gas vs. radius r using the classical fractional step method with high-resolution Godunov scheme;* $\triangle x = \triangle y = 0.01$, $\triangle t = 0.005$, $\tau_0 = 10^{-6}$, $T_{ign} = 0.26$.

later time. The scatter plots in Figure 16 show that the combustion front moves too fast and there is an unphysical intermediate state for the pressure. Moreover, the front does not remain circular.



(a) (b)

FIG. 17. *CJ detonation wave defracting around a corner. Resolved adaptive mesh refinement calculation of temperature and mass fraction of unburnt gas.*

TABLE 2

Error, i.e. difference between radial symmetric $1D$ *reference solution and the solution calculated with the* $2D$ *modified fractional step scheme in normal and tangential direction, in* L_1*-norm and experimental order of convergence (EOC) for the* $2D$ *modification using the high-resolution Godunov scheme;* $\tau_0 = 10^{-6}$, $T_{ign} = 0.26$, $t = 0.4$.

$\triangle x = \triangle y$	$\triangle t$	$\| \Delta\rho \|$	$\| \Delta(\rho u) \|$	$\| \Delta E \|$	$\| \Delta Z \|$
0.04	0.02	0.09481	0.05941	0.20250	0.10190
0.02	0.01	0.06477	0.03847	0.14550	0.04795
EOC		0.55	0.63	0.48	1.09
0.01	0.005	0.04258	0.02927	0.09877	0.02150
EOC		0.61	0.39	0.56	1.16
0.005	0.0025	0.02773	0.01945	0.06395	0.01221
EOC		0.62	0.59	0.63	0.82

Finally, Table 2 shows some numerical order of convergence results. For the numerical approximation of this discontinuous solution the expected order of convergence is between 0.5 and 1; see [19]. With our modified fractional step scheme we could achieve such a convergence order without resolving the reaction zone. Although our numerical solution shows the typical von Neumann peak behind the shock front, we get a relatively large error in the region which approximates the very small reaction zone. This might be a reason why the experimental order of convergence for the variables which contain a peak is lower than the order of convergence for the mass fraction of unburnt gas, which in the stiff case only contains a discontinuity.

The transition between the stiff and the nonstiff case is done in a manner analogous to what was described for the 1D case in section 2.3. Now the maximal rate of reduction (area) of the mass fraction of unburnt gas in those mesh cells which approximate the leading shock wave of the detonation is calculated using all normal and transverse Riemann problems which have an influence on the mesh cell.

3.3. Approximation of a diffracting detonation wave. As a final example, we consider a more interesting and challenging 2D problem, a CJ detonation wave

which is diffracting around a 90 degree corner. We use the reactive Euler equations with the Arrhenius reaction rate law (2.1). For this example the temperature behind the leading shock wave will vary, which causes a change of the reaction rate. We can therefore test our stiff solver as well as the transition between the stiff and the nonstiff approach. Resolved calculations for a similar problem were considered by Xu, Aslam, and Stewart [30] and Aslam and Stewart [1]. The solution of this problem depends strongly on the activation energy and the reaction rate constant. In order to decrease the reaction zone length and make the problem more stiff, we used a larger reaction rate constant than in [30]. We also increased the activation energy in order to obtain the solution structure for our more stiff problem.

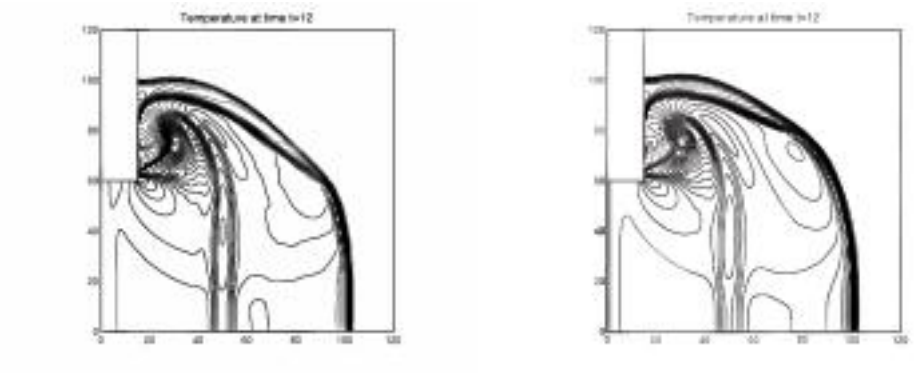
The computational domain is $[0, 120] \times [0, 120]$. There is a solid wall in the upper-left corner of the domain for $x \leq 15$ and $y \geq 60$. Initially unburnt gas is in the region $x > 14$. This unburnt state is given by $\rho_u = 1$, $u_u = 0$, $v_u = 0$, and $p_u = 1$. The ratio of specific heat is $\gamma = 1.4$, the heat release is $q_0 = 25$, the activation energy is $E^+ = 35$, and the reaction rate constant is given as $K_0 = 120$. The unburnt state is again connected by a CJ detonation wave to the burnt state, the half-reaction zone length of this CJ detonation wave is about 0.5.

First we consider a resolved approximation using the adaptive mesh refinement algorithm of AMRCLAW [5], [6]. On the finest discretization level, which is used along the leading shock of the detonation wave, the half-reaction zone length is resolved by 16 mesh points. A 3840×3840 grid would be required to achieve this same resolution on a uniform grid. Contour plots of the temperature as well as of the mass fraction of unburnt gas are given in Figure 17.

As the detonation wave moves around the corner, the leading shock weakens, which decreases the temperature behind the shock. When the shock becomes weak enough, it will no longer raise the temperature above the ignition point. Then we get a nonreactive shock traveling vertically followed by a region in which the temperature is high relative to the initial unburnt state but the reaction rate is still negligible. Some distance behind this shock the gas burns via a deflagration wave. This region can be seen in the plot of the temperature; see Figure 17(a).

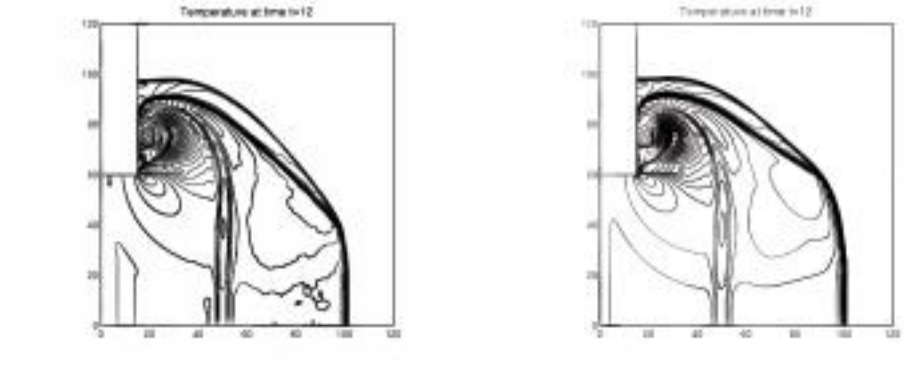
The shock which is traveling in the horizontal direction is still strong enough to ignite a reaction and remains a CJ detonation wave with the constant speed $s_{CJ} = 7.1247$. At some point along the curved shock front there is a transition between the detonation wave traveling horizontally and the shock-deflagration structure traveling vertically. Our goal in this set of experiments is to demonstrate that on underresolved grids the modified method approximates the correct structure better than the classical fractional step method.

Figures 18–20 show underresolved calculations of this problem. When the shock becomes weaker and the temperature behind the shock becomes lower, the reaction rate behind the shock decreases. In the mesh cells which approximate the leading shock front we will therefore also switch from the stiff to the nonstiff solver. Our modified fractional step scheme, used for the plots on the left-hand side, gives a more accurate approximation of the transition of the solution structure from a detonation wave to a nonreactive shock followed by a deflagration wave. The modified approach on the 120×120 grid (Figure 18(a)) gives a representation of this structure that is at least as good as what is seen with the classical method on the 240×240 grid (Figure 19(b)). The modified scheme on the 240×240 grid gives comparable results as the fractional step scheme on the 480×480 grid (compare Figure 19(a)



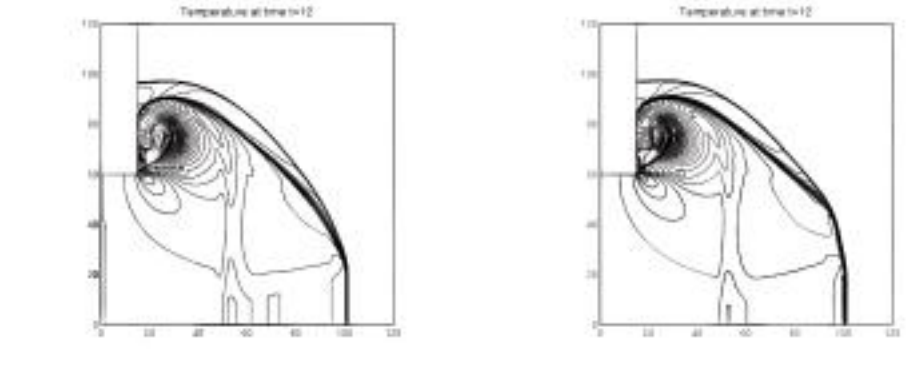
(a) (b)

FIG. 18. *CJ detonation wave diffracting around a corner. Underresolved calculation of the temperature* (120×120 *grid points*). (a) *Using the modified fractional step scheme;* (b) *using the classical fractional step scheme.*



(a) (b)

FIG. 19. *Underresolved calculation of the temperature* (240×240 *grid points*). (a) *Modified fractional step scheme;* (b) *classical fractional step scheme.*



(a) (b)

FIG. 20. *Approximation of the temperature using two grid cells inside the half-reaction zone length* (480×480 *grid points*). (a) *Modified fractional step scheme;* (b) *classical fractional step scheme.*

and Figure 20(b)). Finally the modified scheme on the 480×480 grid produces a very accurate approximation as can be seen by comparing Figure 20(a) with the resolved calculation in Figure 17(a).

For lower or higher values of the activation energy, the point at which the shock structure changes from a reactive to a nonreactive shock moves. If the activation energy is low enough then the shock which is diffracting around the corner may remain strong enough to ignite a reaction and remain a detonation wave everywhere. The reaction rate constant also has an influence on the solution structure. For a larger rate constant the reaction behind the curved shock becomes stronger and the length of the reaction zone becomes smaller. Additional experiments with our method (not shown here) have confirmed that it is robust and yields the correct structure on underresolved grids also in other cases.

4. Conclusions. If stiff source terms are not treated carefully, then the numerical method can produce unphysical solutions even if the scheme is stable. We have considered these numerical difficulties for a combustion problem. If one is not interested in a calculation of the physical processes inside the very small reaction zone but instead wants to determine more global features, e.g., the propagation speed of a detonation wave, then it would be preferable to use a scheme which does not have to resolve the very stiff source term. Here we have shown how the classical fractional step scheme can be modified to give an accurate approximation for the model combustion problem. In section 2 we gave a heuristic motivation of a modification of the fractional step method which was described in that section and extended to 2D problems in section 3.

Our modified fractional step scheme needs information about the structure of the Riemann solution in order to determine the mesh cells over which the leading shock of a detonation wave is smeared. Furthermore, the distance between the shock and contact discontinuity in these mesh cells is required. This information will automatically be provided by an exact Riemann solver as we have described. The distance between the shock and contact discontinuity determines the part of a mesh cell where the source term will be applied in a stiff calculation. For 2D calculations this area depends on all Riemann problems which have an influence to the mesh cell, including the Riemann problems which are solved in order to obtain the fluxes in the transverse direction. However, this further information can be obtained with less effort than the calculation of the change of the cell average due to the solution of the homogeneous problem. Moreover, the modified treatment of the source term is only necessary in those mesh cells which approximate the smeared-out leading shock, e.g., in about three mesh cells for the calculation of the 1D example. Therefore, our modification requires only slightly more effort than the classical fractional step scheme and permits the use of much coarser underresolved grids.

REFERENCES

[1] T.D. Aslam and D.S. Stewart, *Detonation shock dynamics and comparisons with direct numerical simulation*, Combust. Theory Model., 3 (1999), pp. 77–101.

[2] W. Bao and S. Jin, *The Random Projection Method for Stiff Detonation Waves*, Technical Report, School of Mathematics, Georgia Institute of Technology, Atlanta, 1999.

[3] W. Bao and S. Jin, *The Random Projection Method for Hyperbolic Conservation Laws with Stiff Reaction Terms*, Technical Report, School of Mathematics, Georgia Institute of Technology, Atlanta, 1999.

[4] M. Ben-Artzi, *The generalized Riemann problem for reactive flows*, J. Comput. Phys., 81 (1989), pp. 70–101.

[5] M.J. Berger and R.J. LeVeque, amrclaw *software*, http://www.amath.washington.edu/∼rjl/amrclaw/.
[6] M.J. Berger and R.J. LeVeque, *Adaptive mesh refinement using wave-propagation algorithms for hyperbolic systems*, SIAM J. Numer. Anal., 35 (1998), pp. 2298–2316.
[7] A.C. Berkenbosch, *Capturing Detonation Waves for Reactive Euler Equations*, Ph.D. thesis, Technische Universiteit Eindhoven, Eindhoven, The Netherlands, 1995.
[8] A.C. Berkenbosch, E.F. Kaasschieter, and R. Klein, *Detonation capturing for stiff combustion chemistry*, Combust. Theory Model., 2 (1998), pp. 313–348.
[9] A. Bourlioux, *Numerical Study of Unstable Detonations*, Ph.D. thesis, Princeton University, Princeton, NJ, 1991.
[10] A.J. Chorin, *Random choice solution of hyperbolic systems*, J. Comput. Phys., 22 (1976), pp. 517–533.
[11] P. Colella, A. Majda, and V. Roytburd, *Theoretical and numerical structure for reacting shock waves*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 1059–1080.
[12] R. Courant and K.O. Friedrichs, *Supersonic Flow and Shock Waves*, Springer-Verlag, New York, 1985.
[13] J. Falcovitz and M. Ben-Artzi, *Recent developments of the GRP method*, JSME Internat. J. Ser. B, 38 (1995), pp. 497–517.
[14] J. Glimm, *Solutions in the large for nonlinear hyperbolic systems of equations*, Comm. Pure Appl. Math., 18 (1965), pp. 695–715.
[15] E. Godlewski and P.-A. Raviart, *Numerical Approximation of Hyperbolic Systems of Conservation Laws*, Springer-Verlag, New York, 1995.
[16] R. Jeltsch and P. Klingenstein, *Error estimators for the position of discontinuities in hyperbolic conservation laws with source terms which are solved using operator splitting*, Comput. Visual Sci., 1 (1999), pp. 231–249.
[17] D. Kröner, *Numerical Schemes for Conservation Laws*, Wiley-Teubner, Chichester, Stuttgart, 1997.
[18] R.J. LeVeque, clawpack *software*, http://www.amath.washington.edu/∼rjl/clawpack.html.
[19] R.J. LeVeque, *Numerical Methods for Conservation Laws*, Birkhäuser-Verlag, 1990.
[20] R.J. LeVeque, D. Mihalas, E. Dorfi, and E. Müller, *Computational Methods for Astrophysical Fluid Flow*, Saas-Fee Advanced Course 27 Lecture Notes 1997, O. Steiner and A. Gautschy, eds., Springer-Verlag, Berlin, Heidelberg, 1998.
[21] R.J. LeVeque, *Wave propagation algorithms for multidimensional hyperbolic systems*, J. Comput. Phys., 131 (1997), pp. 327–353.
[22] R.J. LeVeque and K.-M. Shyue, *One-dimensional front tracking based on high resolution wave propagation methods*, SIAM J. Sci. Comput., 16 (1995), pp. 348–377.
[23] R.J. LeVeque and H.C. Yee, *A study of numerical methods for hyperbolic conservation laws with stiff source terms*, J. Comput. Phys., 86 (1990), pp. 187–210.
[24] E.S. Oran and J.P. Boris, *Numerical Simulation of Reactive Flow*, Elsevier, New York, 1987.
[25] R.B. Pember, *Numerical methods for hyperbolic conservation laws with stiff relaxation* I. *Spurious solutions*, SIAM J. Appl. Math., 53 (1993), pp. 1293–1330.
[26] B. Sjögreen and B. Engquist, *Numerical approximation of hyperbolic conservation laws with stiff terms*, in Hyperbolic Problems. Theory, Numerical Methods and Applications, Uppsala, Sweden, Vol. II, 1990, Studentlitteratur, Chartwell-Bratt Ltd., Lund, Sweden, Bromley, UK, 1991, pp. 848–860.
[27] J. Smoller, *Shock Waves and Reaction-Diffusion Equations*, Springer-Verlag, New York, 1994.
[28] V.T. Ton, *Improved shock-capturing methods for multicomponent and reacting flows*, J. Comput. Phys., 128 (1996), pp. 237–253.
[29] B. van Leer, *Towards the ultimate conservative difference scheme* IV. *A new approach to numerical convection*, J. Comput. Phys., 23 (1977), pp. 276–299.
[30] S. Xu, T. Aslam, and D.S. Stewart, *High resolution numerical simulation of ideal and non-ideal compressible reacting flows with embedded internal boundaries*, Combust. Theory Model., 1 (1997), pp. 113–142.

SIAM J. SCI. COMPUT.
Vol. 22, No. 5, pp. 1511–1526
© 2000 Society for Industrial and Applied Mathematics

AN EFFICIENT LINEAR SOLVER FOR NONLINEAR PARAMETER IDENTIFICATION PROBLEMS*

YEE LO KEUNG† AND JUN ZOU‡

Abstract. In this paper, we study some efficient numerical methods for parameter identifications in elliptic systems. The proposed numerical methods are conducted iteratively and each iteration involves only solving positive definite linear algebraic systems, although the original inverse problems are ill-posed and highly nonlinear. The positive definite systems can be naturally preconditioned with their corresponding block diagonal matrices. Numerical experiments are presented to illustrate the efficiency of the proposed algorithms.

Key words. parameter identifications, elliptic problems, positive definite

AMS subject classifications. 35R30, 49K

PII. S1064827598346740

1. Introduction.

The purpose of this paper is to investigate some numerical methods for efficiently identifying the unknown coefficient q in the following elliptic problem:

$$-\nabla \cdot (q\nabla u) = f \quad \text{in} \quad \Omega; \quad u = 0 \quad \text{on} \quad \Gamma. \tag{1.1}$$

The identifying process is carried out in a way that the solution u matches its observation data z optimally either in the L^2-norm or in the H^1-norm. Here Ω can be any bounded domain in R^d, $d = 1, 2$, or 3, with piecewise smooth boundary Γ and $f \in H^{-1}(\Omega)$ is given. Practically, we are often asked to recover the parameter $q(x)$ using the observed data z of the solution u. About the data z we are interested in the following two cases:

(a) the measurement of the gradient of u is available,

(b) the measurement of the solution u itself is available.

For parameter identifications, Itô and Kunisch proposed a hybrid method in [11, 12, 13] which combines the output least squares and the equation error formulation within the mathematical framework given by the augmented Lagrangian technique and incorporates a regularization term of the H^2-seminorm of the parameters to be recovered. Chen and Zou [5] and Keung and Zou [14] generalized the method to the case which allows the identifying coefficients to be discontinuous by using the regularization of bounded variations and they provided the rigorous theoretical justifications of the method and its finite element approximation. Independently, Chan and Tai [3, 4, 18] considered also the regularization of bounded variations and did numerous experiments on the performance of the augmented Lagrangian method for identifying highly discontinuous parameters.

The primary approach we are interested in here is based on the following energy-

*Received by editors November 6, 1998; accepted for publication (in revised form) June 18, 2000; published electronically November 17, 2000.

http://www.siam.org/journals/sisc/22-5/34674.html

†Pui Tak Canossian College, 200 Peel Rise, Aberdeen, Hong Kong.

‡Department of Mathematics, The Chinese University of Hong Kong (zou@math.cuhk.edu.hk). Shatin, N.T., Hong Kong. The work of this author was partially supported by Hong Kong RGC grants CUHK 338/96E and CUHK 4004/98P and CUHK Direct Grant.

norm least squares formulation of the aforementioned parameter identifying problem:

$$\text{(P1)} \quad \begin{cases} \text{minimize} & J(q,v) = \frac{1}{2}\int_\Omega q\,|\nabla v - \nabla z|^2 dx + \beta\, N(q) \\ \text{subject to} & (q,v) \in K \times V \quad \text{and} \quad e(q,v) = 0 \end{cases}$$

in case (a) or on the following L^2-norm least squares formulation:

$$\text{(P2)} \quad \begin{cases} \text{minimize} & J(q,v) = \frac{1}{2}\int_\Omega |v - z|^2 dx + \beta\, N(q) \\ \text{subject to} & (q,v) \in K \times V \quad \text{and} \quad e(q,v) = 0 \end{cases}$$

in case (b). Here $V = H_0^1(\Omega)$ and K is a constrained set given by

$$K = \big\{ q \in L^1(\Omega); \quad N(q) < \infty \ \text{ and } \ \alpha_1 \le q(x) \le \alpha_2 \quad \text{almost everywhere (a.e.) in } \Omega \big\}$$

with α_1 and α_2 being two positive constants. $N(q)$ is a regularization term with a weighted coefficient $\beta > 0$. $e(\cdot,\cdot)$ is an operator from $K \times V$ into V defined by the residual equation of (1.1) (in the weak sense):

$$(\nabla e(q,v), \nabla\phi) = (q\nabla v, \nabla\phi) - (f,\phi) \quad \forall\,(q,v) \in K \times V, \quad \phi \in V, \tag{1.2}$$

where $(\cdot,\cdot)$ denotes the duality pairing between $H^{-1}(\Omega)$ and $H_0^1(\Omega)$, which is the extension of the inner product in $L^2(\Omega)$. It is useful to remark that $e(q,v)$ is convex with respect to each variable.

The intention of this paper is to investigate some efficient and easy-to-implement method for the preceding parameter identifications. We know that the regularization of bounded variations, i.e., $N(q) = |q|_{BV(\Omega)}$, is very effective in recovering discontinuous parameters, but it also adds a big difficulty to the numerical resolution process. Namely, one has to solve a nearly singular and indefinite nonlinear minimization system of the form

$$-\beta\nabla \cdot \frac{\nabla q}{\sqrt{|\nabla q|^2 + \varepsilon}} + c(q) = g$$

at each iteration, where ε is the smoothing parameter introduced to smooth the BV-norm term in numerical implementations and $c(q)$ is a linear function of q, which causes the indefiniteness of the system; see [3, 4, 5, 14, 18]. It seems there are very few iterative methods which are known to be globally convergent for solving such a troublesome system. We refer to Chan and Mulet [2], Dobson and Vogel [8], and the references therein for some fixed point methods and their global convergence results when $c(q)$ is not too complicated.

In this paper, instead of the BV-norm regularization we are going to utilize the H^1- or piecewise H^1-norm regularization. We will see that the H^1-norm regularization performs perfectly for identifying smooth coefficients as expected. In fact, our experience indicates that it can also give rise to satisfactory results in most discontinuous coefficient cases. But in the cases where the location of the discontinuities of the identifying coefficients is available, we can achieve much more accurate recoveries even for the coefficients with high discontinuities by using the piecewise H^1-norm regularization. The location of the discontinuities of parameters may be known in some applications or can be identified first by some existing simple algorithms. As we will show later on, with the H^1- or piecewise H^1-norm regularization, each iteration of our algorithms involves only solving linear positive definite algebraic systems, which can be solved by the well-known GMRES iteration with guaranteed convergence [9]. Moreover, the positive definite systems may be preconditioned by their

natural block diagonal matrices, which are very cheap and easy to implement. The numerical experiments will also show that these preconditioners are effective.

The problems (P1) and (P2) will be solved by the augmented Lagrangian method which enables us to relax the residual constraint $e(q,u)=0$ and enhance the convexity of the objective functional. For the purpose we introduce the augmented Lagrangian functional $\mathcal{L}_r : K \times V \times V \to R$ by

$$\mathcal{L}_r(q,v;\mu) = J(q,v) + (\nabla\mu, \nabla e(q,v)) + \frac{r}{2}\|\nabla e(q,v)\|^2_{L^2(\Omega)}, \tag{1.3}$$

where $r \geq 0$ is some given constant.

Following the same arguments as we used in [5] we have the following.

THEOREM 1.1. *$(q^*, v^*) \in K \times V$ is a solution of the minimization problem* (P1) *or* (P2) *if and only if there exists a $\lambda^* \in V$ such that $(q^*, v^*, \lambda^*) \in K \times V \times V$ is a saddle-point of the augmented Lagrangian $\mathcal{L}_r : K \times V \times V \to R$, namely,*

$$\mathcal{L}_r(q^*, v^*; \mu) \leq \mathcal{L}_r(q^*, v^*; \lambda^*) \leq \mathcal{L}_r(q, v; \lambda^*) \quad \forall\, (q,v,\mu) \in K \times V \times V. \tag{1.4}$$

Remark 1.1. We refer to [5] for the proof of similar results as stated in Theorem 1.1 for the TV-regularization, and [12, 13] for other quadratic regularizations.

2. Discretization and augmented Lagrangian algorithms. We now consider the finite element discretization of the augmented Lagrangian $\mathcal{L}_r$ and derive a discrete saddle-point problem.

Let Ω be a polyhedral domain in R^d, $d = 1, 2$, or 3, and $\{\mathcal{T}^h\}_{h>0}$ be a family of regular triangulations (cf. Ciarlet [7]) of the domain Ω with simplicial elements. Denote by V_h the standard piecewise linear finite element space over the triangulation $\mathcal{T}^h$ and

$$\overset{\circ}{V}_h = V_h \cap H_0^1(\Omega), \quad K_h = K \cap V_h.$$

The standard nodal basis functions of $\overset{\circ}{V}_h$ and V_h are denoted as $\{\phi_i\}_{i=1}^{N_0}$ and $\{\phi_i\}_{i=1}^{N}$, respectively.

We define a discrete version of the operator $e(\cdot,\cdot) : K \times V \to V$ as follows:

For any $(q_h, v_h) \in K_h \times \overset{\circ}{V}_h$, $e_h(q_h, v_h) \in \overset{\circ}{V}_h$ is the solution of the system

$$(\nabla e_h(q_h, v_h),\ \nabla\phi) = (q_h \nabla v_h,\ \nabla\phi) - (f, \phi) \quad \forall\, \phi \in \overset{\circ}{V}_h. \tag{2.1}$$

Then we introduce a discrete augmented Lagrangian L_r from $K_h \times \overset{\circ}{V}_h \times \overset{\circ}{V}_h$ to R:

$$L_r(q_h, v_h; \mu_h) = J_h(q_h, v_h) + (\nabla\mu_h,\ \nabla e_h(q_h, v_h)) + \frac{r}{2}\|\nabla e_h(q_h, v_h)\|^2_{L^2(\Omega)} \tag{2.2}$$

with

$$J_h(q_h, v_h) = \frac{1}{2}\int_\Omega q_h |\nabla v_h - \nabla z|^2 dx + \beta \int_\Omega |\nabla q_h|^2\ dx$$

in the case (a) and

$$J_h(q_h, v_h) = \frac{1}{2}\int_\Omega |v_h - z|^2 dx + \beta \int_\Omega |\nabla q_h|^2\ dx$$

in the case (b).

Applying the arguments as we used in [5], we obtain the following.

THEOREM 2.1. *For any $r \geq 0$, there exists at least a saddle-point for the discrete augmented Lagrangian $L_r : K_h \times \overset{\circ}{V}_h \times \overset{\circ}{V}_h \to R$. Moreover, each saddle-point $(q_h^*, v_h^*, \lambda_h^*)$ of L_0 is also a saddle-point of L_r for any $r > 0$.*

We apply the algorithm of the following Uzawa type to find the saddle-points of the discrete augmented Lagrangian L_r defined in (2.2).

UZAWA ALGORITHM. Given $\lambda^0 \in \overset{\circ}{V}_h$. Then for $n \geq 0$, determine the pair $\{q^n, u^n\} \in K_h \times \overset{\circ}{V}_h$ such that

$$L_r(q^n, u^n; \lambda^n) = \min \left\{ L_r(p, v; \lambda^n) \quad \forall\, (p, v) \in K_h \times \overset{\circ}{V}_h \right\} \tag{2.3}$$

and then compute λ^{n+1} by

$$\lambda^{n+1} = \lambda^n + \rho_n \, e_{_h}(q^n, u^n). \tag{2.4}$$

This Uzawa algorithm was proved in [5] to be convergent globally as long as $0 < \rho_n < r$. Earlier proofs of such global convergence of the Uzawa algorithms with different quadratic regularizations can be found in [15] and the references therein. The major cost of the algorithm is solving (2.3), a coupled system with q^n and u^n as unknowns. In our implementations, we will use the following alternative iteration for solving (2.3).

MODIFIED UZAWA ALGORITHM. Given $\lambda^0 \in \overset{\circ}{V}_h$ and $q^0 \in K_h$. Set $n = 1$.

1. Set $k = 1$ and $q^{n,0} = q^{n-1}$.

2. Compute $u^{n,k} \in \overset{\circ}{V}_h$ by solving

$$L_r(q^{n,k-1}, u^{n,k}; \lambda^{n-1}) = \min_{v_h \in V_h^0} L_r(q^{n,k-1}, v_h; \lambda^{n-1}), \tag{2.5}$$

and then compute $q^{n,k} \in V_h$ by solving

$$L_r(q^{n,k}, u^{n,k}; \lambda^{n-1}) = \min_{p_{_h} \in V_h} L_r(p_{_h}, u^{n,k}; \lambda^{n-1}). \tag{2.6}$$

Compute $q^{n,k} = \max\{\alpha_1, \min\{q^{n,k}, \alpha_2\}\}$.
If $\|q^{n,k} - q^{n,k-1}\| \leq$ tolerance, set $u^n = u^{n,k}$ and $q^n = q^{n,k}$, GOTO **3**;
Otherwise set $k = k + 1$, GOTO **2.**

3. Compute λ^n by

$$\lambda^n = \lambda^{n-1} + \frac{3}{4}\, r \, e_{_h}(q^n, u^n). \tag{2.7}$$

Set $n = n + 1$, GOTO **1.**

We will show in the next section that solving the minimization problems (2.5) and (2.6) is equivalent to solving two linear positive definite systems.

2.1. Positive definite systems: Energy-norm case. From the modified Uzawa algorithm we see that the major cost in each iteration of Step 2 is to solve the minimization problems (2.5) and (2.6). We next show that (2.5) and (2.6) are equivalent to two positive definite systems and thus can be solved using the GMRES iteration [16, 17], and more efficiently solved by the preconditioned GMRES

method with their corresponding block diagonal symmetric positive definite matrices as preconditioners.

We first need to derive Gâteaux derivatives of the augmented Lagrangian functional $L_r(q,u;\lambda)$. For the function $e_h(q_h,u_h)$, we can easily see its derivative with respect to q_h, denoted as $e'_h(q_h,u_h)p_h$ at direction $p_h \in V_h$, is a finite element function in $\overset{\circ}{V}_h$ and solves the equation

$$(\nabla e'_h(q_h,u_h)p_h,\nabla\phi) = (p_h\nabla u_h,\nabla\phi) \quad \forall\,\phi\in\overset{\circ}{V}_h, \tag{2.8}$$

while its derivative with respect to u_h, denoted as $e'_h(q_h,u_h)w_h$ at direction $w_h \in\overset{\circ}{V}_h$, is a finite element function in $\overset{\circ}{V}_h$ and solves the equation

$$(\nabla e'_h(q_h,u_h)w_h,\nabla\phi) = (q_h\nabla w_h,\nabla\phi) \quad \forall\,\phi\in\overset{\circ}{V}_h. \tag{2.9}$$

Note from above that for the sake of notation, we distinguish between the directional derivatives of $e_h(q_h,u_h)$ with respect to q_h and u_h only by the direction notation, i.e., depending only on whether we use p_h or w_h. Using these derivatives of $e_h(q_h,u_h)$, we can immediately obtain the Gâteaux derivatives of the augmented Lagrangian functional $L_r(q,u;\lambda)$. Its derivative with respect to u_h at direction w_h is

$$\begin{aligned} &L'_r(q_h,u_h;\lambda_h)w_h\\ &= (q_h\,(\nabla u_h-\nabla z),\nabla w_h) + (\nabla\lambda_h,\nabla e'_h(q_h,u_h)w_h)\\ &\quad + r\,(\nabla e_h(q_h,u_h),\nabla e'_h(q_h,u_h)w_h)\\ &= (q_h\,(\nabla u_h-\nabla z),\nabla w_h) + (q_h\,\nabla\lambda_h,\nabla w_h) + r\,(q_h\,\nabla e_h(q_h,u_h),\nabla w_h), \end{aligned} \tag{2.10}$$

while its derivative with respect to q_h at direction p_h is

$$\begin{aligned} &L'_r(q_h,u_h;\lambda_h)p_h\\ &= \frac{1}{2}\int_\Omega p_h\,|\nabla u_h-\nabla z|^2\,dx + \beta\,(\nabla q_h,\nabla p_h) + (\nabla\lambda_h,\nabla e'_h(q_h,u_h)p_h)\\ &\qquad + r\,(\nabla e_h(q_h,u_h),\nabla e'_h(q_h,u_h)p_h)\\ &= \frac{1}{2}\int_\Omega p_h\,|\nabla u_h-\nabla z|^2\,dx + \beta\,(\nabla q_h,\nabla p_h) + (p_h\,\nabla u_h,\nabla\lambda_h)\\ &\qquad + r\,(p_h\,\nabla u_h,\nabla e_h(q_h,u_h)). \end{aligned} \tag{2.11}$$

Now combining the formula (2.10) and the definition of $e_h(\cdot,\cdot)$, we can find the solution $u^{n,k}$ in (2.5), together with $e_h(q^{n,k-1},u^{n,k})$ as follows:

Find $(u^{n,k},e_h(q^{n,k-1},u^{n,k})) \equiv (u_h,e_h)\in\overset{\circ}{V}_h\times\overset{\circ}{V}_h$ such that

$$(q^{n,k-1}\,\nabla u_h,\nabla w_h) + r\,(q^{n,k-1}\,\nabla e_h,\nabla w_h) = (q^{n,k-1}\,\nabla(z-\lambda^{n-1}),\nabla w_h), \tag{2.12}$$

$$(\nabla e_h,\nabla\phi_h) - (q^{n,k-1}\,\nabla u_h,\nabla\phi_h) = -(f,\phi_h) \tag{2.13}$$

$\forall w_h\in\overset{\circ}{V}_h$ and $\phi_h\in\overset{\circ}{V}_h$. Similarly, the solution $q^{n,k}$ in (2.6) can be solved together with $e_h(q^{n,k},u^{n,k})$:

Find $(q^{n,k},e_h(q^{n,k},u^{n,k})) \equiv (q_h,e_h)\in V_h\times\overset{\circ}{V}_h$ such that

$$\beta\,(\nabla q_h,\nabla p_h) + r\,(\nabla e_h,p_h\nabla u^{n,k}) = -g_k(p_h), \tag{2.14}$$

$$(\nabla e_h,\nabla\phi_h) - (q_h\,\nabla u^{n,k},\nabla\phi_h) = -(f,\phi_h) \tag{2.15}$$

$\forall p_h \in V_h$ and $\phi_h \in \overset{\circ}{V}_h$. Here $g_k(p_h)$ is given by

$$g_k(p_h) = \frac{1}{2}\int_\Omega p_h\,|\nabla(u^{n,k}-z)|^2 dx + (p_h\,\nabla u^{n,k}, \nabla\lambda^{n-1}).$$

Next we show that the linear systems (2.12)–(2.13) and (2.14)–(2.15) are both positive definite. To see this, we introduce

$$\mathbf{A} = (a_{ij}), \quad \mathbf{B} = (b_{ij}), \quad \mathbf{\Lambda} = (\Lambda_j), \quad \mathbf{F} = (f_j)$$

with

$$a_{ij} = (q^{n,k-1}\nabla\phi_i, \nabla\phi_j), \quad b_{ij} = (\nabla\phi_i, \nabla\phi_j)$$

and

$$\Lambda_j = (q^{n,k-1}\nabla(z-\lambda^{n-1}), \nabla\phi_j), \quad f_j = (f, \phi_j)$$

for $i, j = 1, 2, \ldots, N_0$. Then (2.12)–(2.13) can be written as follows:

$$\mathbf{A}\mathbf{u} + r\,\mathbf{A}\mathbf{e} = \mathbf{\Lambda}, \quad \mathbf{B}\mathbf{e} - \mathbf{A}\mathbf{u} = -\mathbf{F},$$

or equivalently

$$(\mathbf{B} + r\mathbf{A})\,\mathbf{e} = \mathbf{\Lambda} - \mathbf{F}, \quad \mathbf{A}\,\mathbf{u} = \mathbf{B}\,\mathbf{e} + \mathbf{F}, \tag{2.16}$$

where $\mathbf{u}$ and $\mathbf{e}$ are the coefficient vectors of u_h and e_h in terms of the basis $\{\phi_i\}_{i=1}^{N_0}$, respectively. These two equations are both symmetric positive definite and can be solved by the conjugate gradient methods, or more efficiently by PCG method with the standard domain decomposition type preconditioners [1].

In an analogous manner, we introduce the following notation for the system (2.14)–(2.15):

$$\mathbf{Q} = (q_{ij}), \quad \mathbf{N} = (n_{ij}), \quad \mathbf{G} = (g_k(\phi_j))$$

with

$$q_{ij} = (\nabla\varphi_i, \nabla\varphi_j), \quad n_{ij} = (\varphi_i\nabla u^{n,k}, \nabla\phi_j)$$

for $i, j = 1, 2, \ldots, N$. Then (2.14)–(2.15) can be written as follows:

$$\beta\,\mathbf{Q}\,\mathbf{q} + r\,\mathbf{N}\,\mathbf{e} = -\mathbf{G}, \quad \mathbf{B}\,\mathbf{e} - \mathbf{N}^\top\,\mathbf{q} = -\mathbf{F}.$$

By eliminating $\mathbf{e}$, we can easily show that the system has a unique solution pair $(\mathbf{q}, \mathbf{e})$. For implementation, we write the system as

$$\begin{pmatrix} r\,\mathbf{B} & -r\,\mathbf{N}^\top \\ r\,\mathbf{N} & \beta\,\mathbf{Q} \end{pmatrix} \begin{pmatrix} \mathbf{e} \\ \mathbf{q} \end{pmatrix} = \begin{pmatrix} -r\,\mathbf{F} \\ -\mathbf{G} \end{pmatrix}. \tag{2.17}$$

If we let $\mathbf{A}_2$ be the coefficient matrix of the above system, and $\mathbf{D}_2$ be the block diagonal matrix of $\mathbf{A}_2$, then we can easily verify that

$$(\mathbf{e}^\top, \mathbf{q}^\top)\mathbf{A}_2 \begin{pmatrix} \mathbf{e} \\ \mathbf{q} \end{pmatrix} = r\,\mathbf{e}^\top\mathbf{B}\,\mathbf{e} + \beta\,\mathbf{q}^\top\mathbf{Q}\,\mathbf{q} = (\mathbf{e}^\top, \mathbf{q}^\top)\,\mathbf{D}_2 \begin{pmatrix} \mathbf{e} \\ \mathbf{q} \end{pmatrix} > 0$$

for any nonzero pair $(\mathbf{q}, \mathbf{e})$. Therefore $\mathbf{A}_2$ is positive definite, and so (2.17) can be solved using the GMRES iterative method, or more efficiently solved using the preconditioned GMRES method. A natural choice of the preconditioner matrix for $\mathbf{A}_2$ is $\mathbf{D}_2^{-1}$, with

$$\mathbf{D}_2 = \begin{pmatrix} r\,\mathbf{B} & \\ & \beta\,\mathbf{Q} \end{pmatrix}, \tag{2.18}$$

which is independent of the number of iterations and so is very cheap and easy to implement.

2.2. Positive definite systems: L^2-norm case. In the L^2-norm case, we can similarly obtain the Gâteaux derivative of the augmented Lagrangian functional with respect to u_h and q_h as in section 2.1. The derivative with respect to u_h at direction w_h is

$$L_r'(q_h, u_h; \lambda_h)w_h = (u_h - z, w_h) + (q_h\,\nabla\lambda_h, \nabla w_h) + r\,(q_h\,\nabla e_h(q_h, u_h), \nabla w_h), \tag{2.19}$$

while its derivative with respect to q_h at direction p_h is

$$L_r'(q_h, u_h; \lambda_h)p_h = \beta\,(\nabla q_h, \nabla p_h) + (p_h\,\nabla u_h, \nabla\lambda_h) + r\,(p_h\,\nabla u_h, \nabla e_h(q_h, u_h)). \tag{2.20}$$

Now combining the formula (2.19) and the definition of $e_h(\cdot,\cdot)$, we can find the solution $u^{n,k}$ in (2.5), together with $e_h(q^{n,k-1}, u^{n,k})$, as follows:

Find $(u^{n,k}, e_h(q^{n,k-1}, u^{n,k})) \equiv (u_h, e_h) \in \overset{\circ}{V}_h \times \overset{\circ}{V}_h$ such that

$$(u_h, w_h) + r\,(q^{n,k-1}\,\nabla e_h, \nabla w_h) = (z, w_h) - (q^{n,k-1}\,\nabla\lambda^{n-1}, \nabla w_h), \tag{2.21}$$

$$(\nabla e_h, \nabla\phi_h) - (q^{n,k-1}\,\nabla u_h, \nabla\phi_h) = -(f, \phi_h) \tag{2.22}$$

$\forall w_h \in \overset{\circ}{V}_h$ and $\phi_h \in \overset{\circ}{V}_h$.

We next show that (2.21)–(2.22) is a positive definite system. To see this, we introduce

$$\mathbf{A} = (a_{ij}), \quad \mathbf{B} = (b_{ij}), \quad \mathbf{M} = (m_{ij}), \quad \mathbf{H} = (h_j), \quad \mathbf{F} = (f_j)$$

with

$$a_{ij} = (q^{n,k-1}\nabla\phi_i, \nabla\phi_j), \quad b_{ij} = (\nabla\phi_i, \nabla\phi_j), \quad m_{ij} = (\phi_i, \phi_j)$$

and

$$h_j = (z, \phi_j) - (q^{n,k-1}\nabla\lambda^{n-1}, \nabla\phi_j), \quad f_j = (f, \phi_j),$$

for $i, j = 1, 2, \ldots, N_0$. Then (2.21)–(2.22) can be written as follows:

$$\mathbf{M}\mathbf{u} + r\,\mathbf{A}\mathbf{e} = \mathbf{H}, \quad \mathbf{B}\mathbf{e} - \mathbf{A}\mathbf{u} = -\mathbf{F},$$

or equivalently,

$$\begin{pmatrix} r\,\mathbf{B} & -r\,\mathbf{A} \\ r\,\mathbf{A} & \mathbf{M} \end{pmatrix} \begin{pmatrix} \mathbf{e} \\ \mathbf{u} \end{pmatrix} = \begin{pmatrix} -r\,\mathbf{F} \\ \mathbf{H} \end{pmatrix}.$$

Let $\mathbf{A}_3$ be the coefficient matrix of the above system and $\mathbf{D}_3$ be the block diagonal matrix of $\mathbf{A}_3$; then we can see that $\mathbf{A}_3$ is positive definite as

$$(\mathbf{e}^\top, \mathbf{u}^\top)\mathbf{A}_3 \begin{pmatrix} \mathbf{e} \\ \mathbf{u} \end{pmatrix} = r\,\mathbf{e}^\top \mathbf{B}\,\mathbf{e} + \mathbf{u}^\top \mathbf{M}\,\mathbf{u} = (\mathbf{e}^\top, \mathbf{u}^\top)\mathbf{D}_3 \begin{pmatrix} \mathbf{e} \\ \mathbf{u} \end{pmatrix} > 0$$

for any nonzero pair $(\mathbf{q}, \mathbf{e})$. Thus the system can be solved by the GMRES method. Another way to solve the system is to first eliminate $\mathbf{u}$ and solve the symmetric positive definite system $(\mathbf{B} + r\mathbf{A}\mathbf{M}^{-1}\mathbf{A})\mathbf{e} = -\mathbf{F} + \mathbf{A}\mathbf{M}^{-1}\mathbf{H}$ for $\mathbf{e}$ and then substitute $\mathbf{e}$ back to the system. Note that if we use the lumped mass scheme to approximate the mass matrix $\mathbf{M}$, then $\mathbf{M}$ is a diagonal matrix. This simplification keeps the same accuracy of approximation as we calculate $\mathbf{M}$ exactly (cf. Hoffmann–Zou [10]).

On the other hand, combining the formula (2.20) and the definition of $e_{_h}(\cdot,\cdot)$, we know that the solution $q^{n,k}$ in (2.6) satisfies the same equation as (2.17) but with $g_k(\phi_j)$ replaced by

$$\int_\Omega \varphi_i\, \nabla u^{n,k} \cdot \nabla \lambda^{n-1}\, dx\ .$$

3. Regularization with piecewise H^1 seminorms. In some applications, the location of the discontinuities of the identifying parameters may be available, either achieved technically or obtained by some simple numerical methods. In this case, we propose to use the piecewise H^1-seminorm regularization to obtain a more accurate recovery for the highly discontinuous parameters.

For simplicity, let us assume that the coefficient $q(x)$ has jumps only across the interface Γ between two subdomains Ω_1 and Ω_2 of Ω, namely,

$$q(x) = \begin{cases} q_1(x), & x \in \Omega_1, \\ q_2(x), & x \in \Omega_2, \end{cases}$$

with $\bar{\Omega} = \bar{\Omega}_1 \cup \bar{\Omega}_2$. The corresponding residual equation (1.2) becomes

$$(\nabla e(q,v), \nabla\phi) = (q\nabla v, \nabla\phi) - \left\{ \int_\Gamma \left[q \frac{\partial u}{\partial \mathbf{n}} \right] \phi + (f, \phi) \right\} \quad \forall\, (q,v) \in K \times V, \quad \phi \in V.$$

Here $\left[q\frac{\partial u}{\partial \mathbf{n}}\right]$ is the jump of the flux $q\frac{\partial u}{\partial \mathbf{n}}$ across the interface Γ, which is given and in fact vanishes in many real applications. We assume that Γ is a piecewise line segment ($d = 2$) or a polygon ($d = 3$). In other cases the finite element discretization needs to be taken care of very technically (cf. Chen–Zou [6]). Let $\mathcal{T}^h$ be a triangulation of Ω with all the finite elements aligning with the interface Γ and $\tilde{V}_h$ be the standard piecewise linear finite element space, respectively, in Ω_1 and Ω_2, namely, with piecewise linear functions which are continuous both in Ω_1 and Ω_2 but probably having jumps across Γ. We define the discrete constrained set to be $\tilde{K}_h = K \cap \tilde{V}_h$.

Now we can reformulate the augmented Lagrangian functional $\mathcal{L}_r$ in (1.3) by replacing the regularization term $N(q)$ there by

$$\tilde{N}(q) = \beta \int_{\Omega_1} |\nabla q_1|^2\, dx + \beta \int_{\Omega_2} |\nabla q_2|^2\, dx.$$

And in this case the discrete constrained set is defined to be $\tilde{K}_h = K \cap \tilde{V}_h$, and the discrete augmented Lagrangian functional L_r in (2.2) is now defined on $\tilde{K}_h \times \overset{\circ}{V}_h \times \overset{\circ}{V}_h$

and has the same form as (2.2) but with the regularization terms in $J_h(q_h, v_h)$ replaced by $\tilde{N}(q_h)$.

All the derivations in sections 2.1–2.2 can be carried over with only minor notational changes. The resulting linear algebraic systems from (2.5) and (2.6) are still all positive definite as we showed in sections 2.1–2.2.

4. Numerical experiments. We now show some numerical experiments on the proposed method for parameter identifications. We apply the modified Uzawa algorithm for the identification of the coefficients in the following test problems:

$$-\frac{d}{dx}\Big(q(x)\frac{d}{dx}u(x)\Big) = f(x), \quad x \in (0,1), \tag{4.1}$$

$$u(0) = 0, \quad u(1) = 0 \tag{4.2}$$

and

$$-\nabla \cdot (q(x,y)\nabla u) + c(x,y)u = f(x,y), \quad (x,y) \in \Omega, \tag{4.3}$$

$$u(x,y) = 0, \quad (x,y) \in \partial\Omega, \tag{4.4}$$

where $\Omega = (0,1) \times (0,1)$. In our implementations, the interval $(0,1)$ is divided into N uniformly distributed subintervals of length $h = 1/N$, the domain $(0,1) \times (0,1)$ is triangulated uniformly into triangular elements with horizontal and vertical edges of length $h = 1/N$.

Most parameters required in the algorithm are attached in each figure. The error $\mathcal{E}$ shown is the relative L^2-norm error between the exact parameter $q(x)$ to be identified and the numerically identified parameter q_h. The numerically recovered q_h shown throughout this section are the result obtained at the 5th iteration, i.e., $n = 5$, in the modified Uzawa algorithm. The augmented Lagrangian coefficient r and the initial guess of the Lagrangian multiplier λ^0 are always set to be 1 and 0, and the finite element mesh size h to be $1/80$ unless otherwise specified. The lower and upper bounds α_1 and α_2 in the constrained set K are taken to be 0.5 and 20.0, respectively. The tolerance is taken to be 10^{-6}, i.e., $\|q^{n,k} - q^{n,k-1}\|_\infty \le 10^{-6}$.

To generate the noisy observation data, we will always take the following form:

$$\nabla z^\delta(x) = \nabla z + \delta\,\mathrm{rand}(x)$$

in case (a) or

$$z^\delta(x) = z + \delta\,\mathrm{rand}(x)$$

in case (b), where $\mathrm{rand}(x)$ is a uniformly distributed random (vector-valued in (a)) function in $[-1,1]$, δ is the noise level parameter. Then in our implementations, we replace all previously appeared ∇z and z by ∇z^δ and z^δ, respectively.

4.1. Energy-norm case. We first show some results for the energy-norm case.

Example 1. We take the observed data z as

$$z(x) = u(q)(x) = \sin(2\pi x)$$

with the coefficient $q(x)$ as

$$q(x) = 3 + 2x^2 - 2\sin(2\pi x).$$

Figures 1 and 2 show the exact solution $q(x)$ (the solid line in Figure 1 but dashed line in Figure 2) and the numerically identified solution $q_h(x)$ (the "o" line in Figure

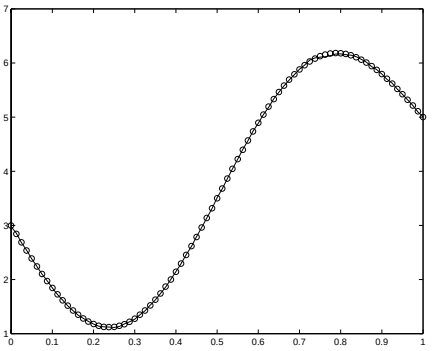


FIG. 1: $q_h^0 = 5.0, \beta = 10^{-5}, \delta = 1\%, \mathcal{E}$=0.0025.

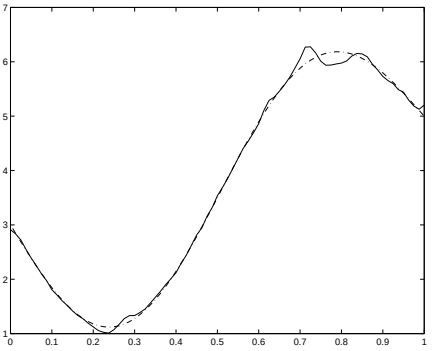


FIG. 2: $q_h^0 = 5.0, \beta = 10^{-5}, \delta = 10\%, \mathcal{E}$=0.021.

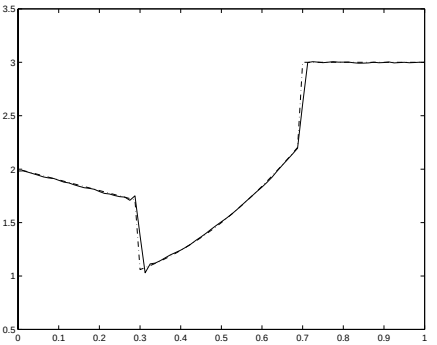


FIG. 3: $q_h^0 = 5.0, \beta = 5 \times 10^{-5}, \delta = 1\%, \mathcal{E}$=0.026.

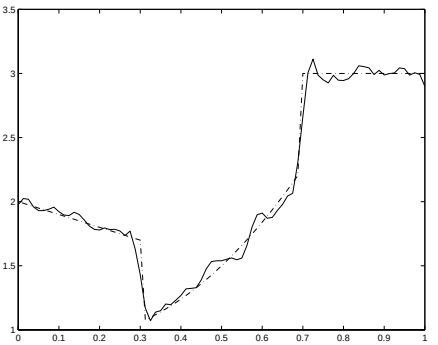


FIG. 4: $q_h^0 = 5.0, \beta = 10^{-5}, \delta = 10\%, \mathcal{E}$=0.031.

1 but solid line in Figure 2) with the noise level $\delta = 1\%$ and $\delta = 10\%$, respectively. Note that the initial guess $q_h^0 = 5.0$ is not a good initial guess at all, but the numerical method converges very stably and fast. Figures 1 and 2 are the results obtained at the 5th iteration ($n = 5$). As the numerical algorithm is a globally convergent algorithm, one can take the initial q_h^0 much worse than $q_h^0 = 5.0$, say, $q_h^0 = 50.0$, and still obtain the same accurate recovery. This is true for all examples shown in this section.

Example 2. We take the observed data z as

$$z(x) = u(q)(x) = \sin(\pi x)$$

with the discontinuous coefficient $q(x)$ as

$$q(x) = \begin{cases} 2 - x, & x \in [0,\ 0.3], \\ 1 - x + 4x^2, & x \in (0.3,\ 0.7), \\ 3, & x \in [0.7,\ 1]. \end{cases}$$

Figures 3 and 4 show the exact solution $q(x)$ (the dashed line) and the numerically identified solution $q_h(x)$ (the solid line) with the noise level $\delta = 1\%$ and 10%, respectively.

Example 3. We take the observed data z as

$$z(x, y) = u(q)(x, y) = \sin(\pi x)\sin(\pi y)$$

and the coefficient $q(x, y)$ as

$$q(x, y) = 1 + 6x^2 y(1 - y).$$

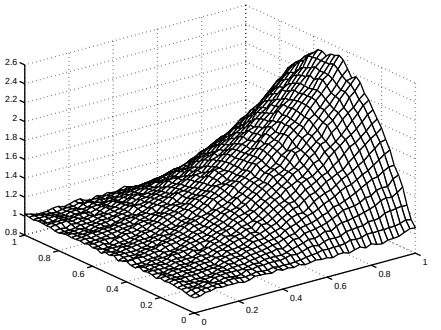


FIG. 5: $q_h^0 = 5.0, \beta = 10^{-5}, \delta = 1\%, \mathcal{E}$=0.025.

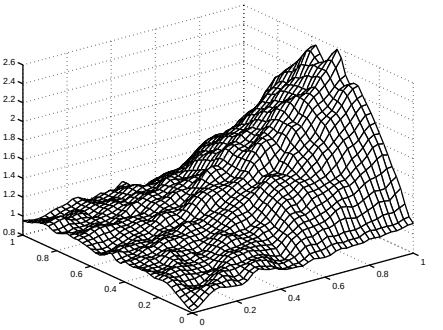


FIG. 6: $q_h^0 = 5.0, \beta = 10^{-4}, \delta = 10\%, \mathcal{E}$=0.079.

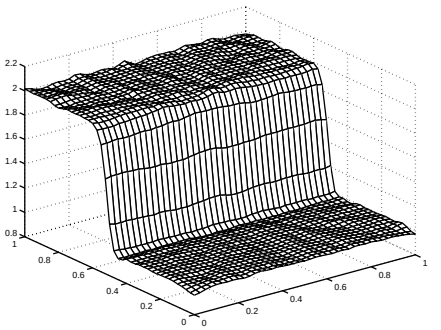


FIG. 7: $q_h^0 = 5.0, \beta = 10^{-5}, \delta = 1\%, \mathcal{E}$=0.049.

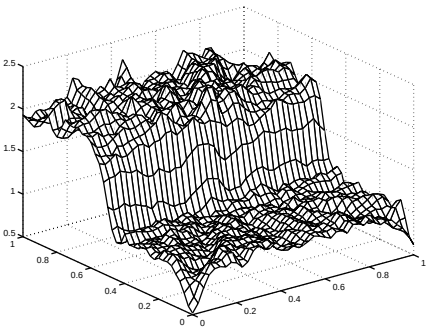


FIG. 8: $q_h^0 = 5.0, \beta = 10^{-5}, \delta = 10\%, \mathcal{E}$=0.069.

Figures 5–6 show the numerically identified solution $q_h(x, y)$ with $h = 1/40$ and the noise level $\delta = 1\%$ and $\delta = 10\%$, respectively.

Example 4. We take the observed data z as

$$z(x) = u(q)(x, y) = \sin(\pi x) \sin(\pi y)$$

with a discontinuous coefficient $q(x, y)$ as

$$\begin{cases} q(x, y) = 1, & y \in [0,\ 0.5], \\ q(x, y) = 2, & y \in (0.5, 1]. \end{cases}$$

Figures 7–8 show the numerically identified solution $q_h(x, y)$ with $h = 1/40$, and the noise level $\delta = 1\%$ and $\delta = 10\%$, respectively.

4.2. L^2-norm case. In this subsection, we show some numerical experiments on the proposed method with the L^2-norm formulation for the identification of the coefficient $q(x)$ in the test problems (4.1)–(4.2) and (4.3)–(4.4). The L^2-norm formulation assumes that the observation data is available via the pointwise function values, not the gradient values as in the energy-norm case. Note that the behavior of the gradient of a function is more essential for knowing the changes of a function than the function values, so in general we cannot expect the same nice performance in the current case as in the energy-norm case.

Example 5. The observed data z and the coefficient $q(x)$ are the same as in Example 1. Figure 9 shows the exact solution $q(x)$ (the dashed line) and the numerically identified solution $q_h(x)$ (the solid line) with the noise level $\delta = 1\%$.

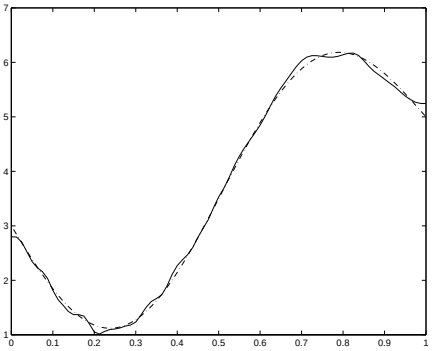


FIG. 9: $q_h^0 = 3.0, \beta = 10^{-7}, \delta = 1\%, \mathcal{E}$=0.018.

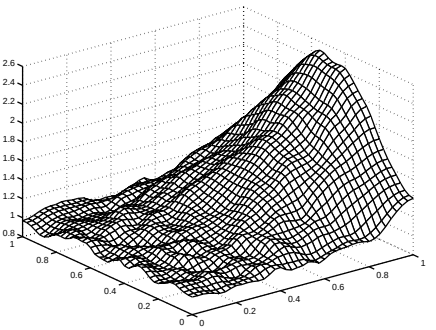


FIG. 10: $q_h^0 = 5.0, \beta = 10^{-6}, \delta = 1\%, \mathcal{E}$=0.031.

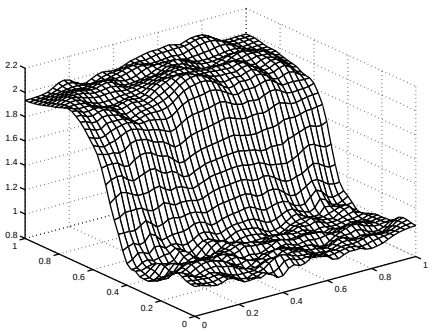


FIG. 11: $q_h^0 = 5.0, \beta = 10^{-6}, \delta = 1\%, \mathcal{E}$=0.091.

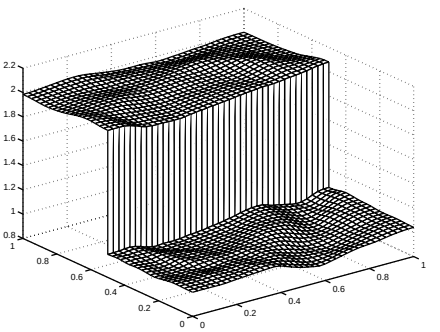


FIG. 12: $q_h^0 = 5.0, \gamma = 10^{-4}, \delta = 10\%, \mathcal{E}$=0.018.

Example 6. The observed data z and the discontinuous coefficient $q(x)$ are taken to be the same as in Example 3. Figure 10 shows the numerical solution $q_h(x, y)$ with $h = 1/40$ and the noise level $\delta = 1\%$.

Example 7. The observed data z and the discontinuous coefficient $q(x)$ are taken to be the same as in Example 4. Figure 11 shows the numerically identified solution $q_h(x, y)$ with $h = 1/40$ and the noise level $\delta = 1\%$; the recovery seems acceptable but not very satisfactory.

If the location of the discontinuity of the parameter $q(x)$ is available, then we can achieve much better identification by using the piecewise H^1-seminorm regularization (see section 3). Figure 12 is the numerically identified results using piecewise H^1-seminorm regularization with the noise level $\delta = 10\%$ and $h = 1/40$. Compared with

Figure 11, where the noise level is only $\delta = 1\%$, we can see that the numerical results using piecewise H^1-seminorm regularization are much more satisfactory.

4.3. GMRES iteration. As stated in the modified Uzawa algorithm, we have to eventually solve two minimization problems (2.5) and (2.6) at each iteration of step 2. But we have shown in sections 2.1–2.2 that these two problems are both equivalent to solving two positive definite systems. In this section we take the energy-norm formulation as an example and present some detailed numerical performance of the Uzawa algorithm using the GMRES method to solve the nonsymmetric but positive definite system (2.17). From the GMRES theory we know this system can be solved by GMRES with guaranteed convergence. However, when discretization mesh size is very fine, (2.17) is large and still expensive to solve, especially in two and three dimensions. So good preconditioners are needed. Fortunately we have a cheap and natural preconditioner available, i.e., M^{-1}. Here M is the block diagonal coefficient matrix

$$M = \begin{pmatrix} rB & 0 \\ 0 & \beta Q \end{pmatrix} \tag{4.5}$$

of (2.17). We are going to demonstrate some numerical experiments and compare the results using GMRES with or without preconditioning.

In our implementation, the stopping criterion of all GMRES or preconditioned GMRES iteration for a system $Ax = b$ is taken to be 10^{-6}, i.e.,

$$\frac{\|b - Ax\|}{\|b\|} < 10^{-6}.$$

The system $Mx = c$ involved in the preconditioned GMRES is always solved by the backward and forward substitutions using the LU factorization of M, i.e., $M = LU$. Note that the two diagonal block matrices B and Q of the preconditioner M are both symmetric and positive definite, in fact they are the stiffness matrices arising from the finite element discretization of the simple Laplacian operator, so the preconditioner M is unchanged in the whole outer (Uzawa iteration) and inner (GMRES) iterations. Thus the LU factorization of M needs to be done only once for all, namely, it can be done before we start the Uzawa algorithm.

We remark that the matrix Q used in the preconditioner M is a singular matrix, so the upper triangular matrix U of the LU factorization may have zeros on its diagonal. In this case, we add 10^{-10} (for $d = 1$) and 10^{-6} (for $d = 2$) to those diagonal entries with magnitudes less than 10^{-10}.

Another way to avoid the singularity of the matrix Q is to replace the seminorm $\int_\Omega |\nabla q|^2 \, dx$ by the full-norm $\int_\Omega (|\nabla q|^2 + q^2) \, dx$ as the regularization term. In this case,

$$Q = (q_{ij}), \quad q_{ij} = (\nabla \varphi_i, \nabla \varphi_j) + (\varphi_i, \varphi_j)$$

and the preconditioner $\tilde{M}$ formed as in (4.5) is always nonsingular.

In the following examples, we will demonstrate the effectiveness of the preconditioner M and the effect of seminorm and full-norm regularizations, by showing the number of GMRES iterations performed.

In each table below,

(a) is the total number of GMRES solvers used when the number of the outer-loop iteration of the modified Uzawa algorithm is $n = 5$;

TABLE 4.1
Averaged number of iterations in each GMRES solver using H^1-seminorm regularization.

h	Without preconditioning		With preconditioning	
	(a)	(b)	(a)	(b)
1/80	65	132	66	132
1/160	65	215	67	160
1/320	65	377	67	166

TABLE 4.2
Averaged number of iterations in each GMRES solver using H^1-full-norm regularization.

h	Without preconditioning		With preconditioning	
	(a)	(b)	(a)	(b)
1/80	66	132	67	122
1/160	66	215	66	146
1/320	69	376	67	156

TABLE 4.3
Averaged number of iterations in each GMRES solver using H^1-seminorm regularization.

h	Without preconditioning		With preconditioning	
	(a)	(b)	(a)	(b)
1/80	57	123	57	115
1/160	57	203	57	129
1/320	57	363	57	135

(b) represents the average number of iterations within one GMRES solver, i.e., the total number of iterations accumulated in all GMRES iterations divided by the number given in (a).

Example 8. The observed data z and the exact coefficient $q(x)$ are taken to be the same as in Example 1. Results by using H^1-seminorm and full-norm regularizations with or without preconditioning are shown in Tables 4.1 and 4.2.

For all the examples given in this section, the graphs of the numerically identified solution $q_h(x)$ are similar to the ones we have shown in section 4.1, so we do not present them here again. From Tables 4.1–4.2, we see that the numbers of GMRES solvers performed (column (a)) are approximately the same with different mesh sizes, which means the convergence of the algorithm is almost *independent of* finite element mesh sizes; this seems to be a surprising observation. When considering the average number of iterations within one GMRES solver (column (b)), the one with preconditioning seems to have much fewer iterations, and this averaged number of iterations seems to be asymptotically constant. This indicates the preconditioners we used are almost optimal, namely, the numbers of the iterations of the preconditioned GMRES required to reach the stopping criteria are independent of the finite element mesh sizes. So the advantage of the preconditioning is more obvious when the problem sizes are larger. We can see such conclusion more evidently when we compare the results in Tables 4.1–4.2 and Tables 4.3–4.4 for the one-dimensional problems with the results in Tables 4.5–4.6 and Tables 4.7–4.8 for the two-dimensional problems.

The above observations are true for all other examples shown below.

Example 9. We take the observed data z and the true parameter $q(x)$ (discontinuous) to be the same as in Example 2. Results using H^1-seminorm and full-norm regularizations with or without preconditioning are shown in Tables 4.3 and 4.4.

Example 10. We take the observed data z and the true coefficient $q(x, y)$ to be

TABLE 4.4
Averaged number of iterations in each GMRES solver using H^1-full-norm regularization.

h	Without preconditioning		With preconditioning	
	(a)	(b)	(a)	(b)
1/80	57	123	57	112
1/160	57	204	57	127
1/320	57	363	57	131

TABLE 4.5
Averaged number of iterations in each GMRES solver using H^1-seminorm regularization.

h	Without preconditioning		With preconditioning	
	(a)	(b)	(a)	(b)
1/10	42	143	41	125
1/20	45	446	45	227
1/30	45	830	45	269
1/40	45	1122	44	276

TABLE 4.6
Averaged number of iterations in each GMRES solver using H^1-full-norm regularization.

h	Without preconditioning		With preconditioning	
	(a)	(b)	(a)	(b)
1/10	41	143	41	110
1/20	45	446	45	207
1/30	45	830	44	236
1/40	45	1122	44	247

TABLE 4.7
Averaged number of iterations in each GMRES solver using H^1-seminorm regularization.

h	Without preconditioning		With preconditioning	
	(a)	(b)	(a)	(b)
1/10	42	145	42	130
1/20	49	449	49	245
1/30	50	840	50	284
1/40	50	1149	50	295

TABLE 4.8
Averaged number of iterations in each GMRES solver using H^1-full-norm regularization.

h	Without preconditioning		With preconditioning	
	(a)	(b)	(a)	(b)
1/10	43	144	42	114
1/20	49	449	49	224
1/30	50	840	50	266
1/40	50	1148	50	282

the same as in Example 3. Results using H^1-seminorm and full-norm regularizations with or without preconditioning are shown in Tables 4.5 and 4.6.

Example 11. The observed data z and the true parameter $q(x, y)$ (discontinuous) are taken to be the same as in Example 4. Results using H^1-seminorm and full-norm regularizations with or without preconditioning are shown in Tables 4.7–4.8.

Acknowledgments. The authors wish to thank an anonymous referee and the editor for many constructive comments that improved this paper.

REFERENCES

[1] T. F. Chan and T. P. Mathew, *Domain decomposition algorithms*, Acta Numer., Cambridge University Press, Cambridge, UK, 1994, pp. 61–143.

[2] T. F. Chan and P. Mulet, *On the convergence of the lagged diffusivity fixed point method in total variation image restoration*, SIAM J. Numer. Anal., 36 (1999), pp. 354–367.

[3] T. F. Chan and X.-C. Tai, *Augmented Lagrangian and Total Variational Methods for Recovering Discontinuous Coefficients from Elliptic Equations*, Technical Report CAM 97-2, Dept. of Math., University of California at Los Angeles, 1997.

[4] T. F. Chan and X.-C. Tai, *Identification of Discontinuous Coefficients from Elliptic Problems Using Total Variation Regularization*, Technical Report CAM 97-35, Dept. of Math., University of California at Los Angeles, 1997.

[5] Z. Chen and J. Zou, *An augmented Lagrangian method for identifying discontinuous parameters in elliptic systems*, SIAM J. Control Optim., 37 (1999), pp. 892–910.

[6] Z. Chen and J. Zou, *Finite element methods and their convergence for elliptic and parabolic interface problems*, Numer. Math., 79 (1998), pp. 175–202.

[7] P. G. Ciarlet, *Basic error estimates for elliptic problems*, in Handbook of Numerical Analysis, P. G. Ciarlet and J.-L. Lions, eds., Vol. II, North Holland, Amsterdam, 1991, pp. 17–352.

[8] D. C. Dobson and C. R. Vogel, *Convergence of an iterative method for total variation denoising*, SIAM J. Numer. Anal., 34 (1997), pp. 1779–1791.

[9] S. C. Eisenstat, H. C. Elman, and M. H. Schultz, *Variational iterative methods for nonsymmetric systems of linear equations*, SIAM J. Numer. Anal., 20 (1983), pp. 345–357.

[10] K.-H. Hoffmann and J. Zou, *Parallel solution of variational inequality problems with nonlinear source terms*, IMA. J. Numer. Anal., 16 (1996), pp. 31–45.

[11] K. Itô, M. Kroller, and K. Kunisch, *A numerical study of an augmented Lagrangian method for the estimation of parameters in elliptic systems*, SIAM J. Sci. Comput., 12 (1991), pp. 884–910.

[12] K. Itô and K. Kunisch, *The augmented Lagrangian method for parameter estimation in elliptic systems*, SIAM J. Control Optim., 28 (1990), pp. 113–136.

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

[14] Y. Keung and J. Zou, *Numerical identifications of parameters in parabolic systems*, Inverse Problems, 14 (1998), pp. 83–100.

[15] K. Kunisch and X.-C. Tai, *Sequential and parallel splitting methods for bilinear control problems in Hilbert spaces*, SIAM J. Numer. Anal., 34 (1997), pp. 91–118.

[16] Y. Saad, *Iterative Methods for Sparse Linear Systems*, PWS Publishing Company, Boston, 1996.

[17] Y. Saad and M. H. Schultz, *GMRES: A generalized minimum residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Comput., 7 (1986), pp. 856–869.

[18] X.-C. Tai, J. Frøyen, M. Espedal, and T. Chan, *Overlapping domain decomposition and multigrid methods for inverse problems*, Contemp. Math, 218 (1998), pp. 523–529.

SIAM J. SCI. COMPUT.
Vol. 22, No. 5, pp. 1527–1532
© 2000 Society for Industrial and Applied Mathematics

A STRUCTURAL DIAGNOSIS OF SOME IC ORDERINGS*

ROBERT BRIDSON† AND WEI-PAI TANG‡

Abstract. We present a novel analysis of the potential effectiveness of a matrix ordering for the incomplete Cholesky factorization (IC) in terms of just the sparsity structure. By looking at the structure of the approximate inverse implicitly created by IC we can help to explain the success of reverse Cuthill–McKee orderings, the problems IC(0) has under red-black orderings that disappear when extra fill is included, and where fill must be added to make fill-reducing orderings such as minimum degree effective.

Key words. ordering, incomplete factorization, incomplete Cholesky, preconditioner, sparse matrix, graph theory

AMS subject classifications. 05C78, 65F10, 65F50

PII. S1064827599353841

1. Introduction.

The incomplete LU factorization (ILU) class of preconditioners has proven to be very competitive. However, like any algorithm that involves factoring a matrix into triangular parts (exactly or only approximately), the ordering of the rows and columns can have a crucial effect. The issue of how best to choose that ordering is an important area of research continuing up to today [1, 3, 4, 5, 6, 8, 9].

A natural first choice for an ordering might be the best fill-reducing ordering available—maybe a minimum degree variant or some kind of nested dissection. This follows the intuition that if there are less fill entries to drop, then the incomplete factorization will be more accurate.

However, it often turns out that envelope orderings like reverse Cuthill–McKee (RCM) orderings do better despite allowing more fill, at least for level-of-fill based incomplete factorizations and for fairly symmetric problems (e.g., see [1] for discussion of highly nonsymmetric matrices). Another puzzling observation is that red-black orderings usually give poor performance for ILU(0) but competitive performance for higher levels of fill.[1] These and many other orderings were considered in depth in [8], but the reasons for the varying performance are still not clear. Other papers have since further explored the effect of ordering on ILU, usually with a view toward explaining (and minimizing) the typical tradeoff between parallelism and convergence speed, but have generally stuck to model PDE problems on uniform rectangular grids [6, 9].

The following analysis provides some justification for these phenomena that works with the sparsity structure of the matrix, without considering the numerical values. The new results may be used as a guide for the construction of high-quality orderings,

*Received by the editors April 9, 1999; accepted for publication (in revised form) July 24, 2000; published electronically November 17, 2000. This work was supported by the Natural Sciences and Engineering Council of Canada, and Communications and Information Technology Ontario (CITO), funded by the Province of Ontario.

http://www.siam.org/journals/sisc/22-5/35384.html

†SCCM Program, Gates 2B, Stanford University, Stanford, CA 94305 (rbridson@stanford.edu).

‡Department of Computer Science, University of Waterloo, Waterloo, ON, N2L 3G1, Canada (wptang@elora.uwaterloo.ca).

[1]Gene Golub has recently suggested that ILU shouldn't be applied directly to a red-black ordered matrix, but rather to the Schur complement formed after diagonal scaling and pre-elimination of the red nodes; our analysis could then be applied to the structure and ordering of the Schur complement.

particularly in the common case where matrix entries may change from solution to solution but the sparsity structure and ordering stay the same.

We restrict our attention to symmetric positive definite matrices A with an incomplete Cholesky factorization $\bar{L}\bar{L}^T$ approximating the exact factorization LL^T. Our analysis is purely structural, however, so the results are equally applicable to matrices with symmetric structure only and whose ILU factors have identical structure after transposition, as is the case with ILU(k). Extensions to the fully unsymmetric case are possible, but we have yet to find satisfying, intuitive results there.

2. Inverse structure. The goal of an incomplete factorization is that the application of the preconditioner is close to the application of the true inverse of A. In other words, we want $(\bar{L}\bar{L}^T)^{-1}$ close to A^{-1}, viewing IC as a kind of implicit approximate inverse. This study focuses on just what the sparsity structure of the matrix can tell us; for orderings that consider numerical values, see, for example, [3, 4, 5, 6, 9].

One obvious desirable property for the incomplete factorization is that $(\bar{L}\bar{L}^T)^{-1}$ at least should have the same nonzero structure as A^{-1} (e.g., fully dense for irreducible matrices, such as those arising from elliptic PDEs). If $(\bar{L}\bar{L}^T)^{-1}$ is constrained to have zeros where A^{-1} has nonnegligible entries, the approximation cannot be good—the required coupling between nodes is absent.

Before proceeding, we will introduce some graph theory notation (see [10, 11] for more details). The (directed) graph of an $n \times n$ matrix B is a graph with vertices $1, \ldots, n$, and an arc $i \to j$ if and only if $B_{ij} \neq 0$. We will just write B instead of "the graph of B" where it is clear. A dipath is an ordered set $\langle i_1, i_2, \ldots, i_p \rangle$ such that $i_1 \to i_2$, $i_2 \to i_3$, $\ldots$, and $i_{p-1} \to i_p$, often written as $i_1 \to i_2 \to \cdots \to i_p$ or simply $i_1 \rightsquigarrow i_p$. The transitive closure of a graph $\mathcal{G}$ is a graph $\mathcal{G}^*$ on the same vertices but with an arc $i \to j$ in $\mathcal{G}^*$ whenever $i \rightsquigarrow j$ in $\mathcal{G}$.

In [11] the nonzero structure of the inverse of a matrix, assuming no fortuitous cancellation, is characterized in terms of its graph: the structure of B^{-1} is the transitive closure of B. In other words, $(B^{-1})_{ij} \neq 0$ if and only if there is a dipath $i \rightsquigarrow j$ in B.

We can immediately determine the structure of A^{-1} then. Assuming A is connected, and making use of its symmetric structure, there is a dipath between any two nodes. Hence A^{-1} is completely dense (first shown in [7]). We are thus interested in having $(\bar{L}\bar{L}^T)^{-1}_{ij} \neq 0$ for all i, j. Observe that

$$(\bar{L}\bar{L}^T)^{-1}_{ij} = \sum_{k=1}^{n} (\bar{L}^{-1})_{ki} (\bar{L}^{-1})_{kj}.$$

Thus $(\bar{L}\bar{L}^T)^{-1}_{ij} \neq 0$ if and only if there is some k such that $k \rightsquigarrow i$ and $k \rightsquigarrow j$ in $\bar{L}$ (assuming no fortuitous cancellation). Notice that since $\bar{L}$ is lower triangular, $u \to v$ in $\bar{L}$ implies that $u \geq v$, and so similarly $u \rightsquigarrow v$ implies $u \geq v$ (and in fact the nodes on the dipath are monotonically decreasing). Therefore it is necessary that $n \rightsquigarrow i$ monotonically for all i since no $k > n$—and this is clearly sufficient too, since then $n \rightsquigarrow i$ and $n \rightsquigarrow j$ for all i, j, which implies $(\bar{L}\bar{L}^T)^{-1}_{ij} \neq 0$ for all i, j.

Thus $(\bar{L}\bar{L}^T)^{-1}$ is fully dense if and only if node $n-1$ is adjacent to n (in $\bar{L}$), node $n-2$ is adjacent to the set $\{n-1, n\}$, node $n-3$ is adjacent to $\{n-2, n-1, n\}$, etc. Imagine selecting the ordering (the labelling of the nodes) in reverse order: choosing which node will be n, then which will be $n-1$, etc. Recall that a graph traversal is a step-by-step selection of the graph's nodes such that at each step, the next node selected is adjacent to some previously selected node. Then what we have follows.

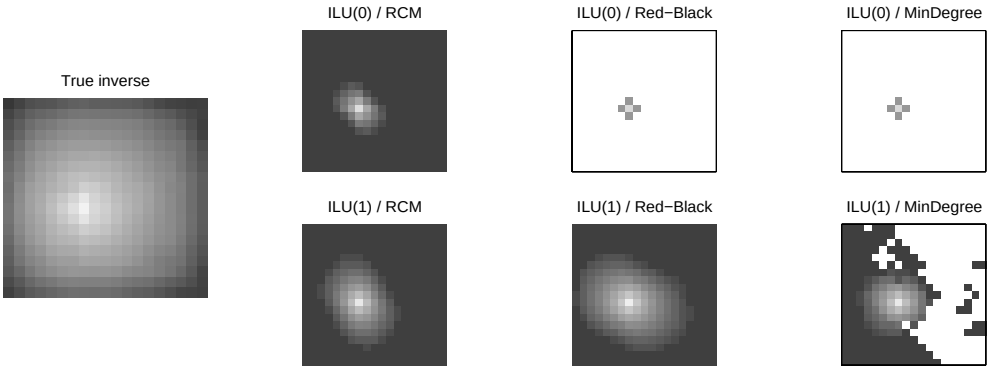


FIG. 2.1. *True inverse and implicit approximate inverses for IC*(0) *and IC*(1) *of the* 5*-point Laplacian under different orderings. Lighter grey indicates larger entries, except that pure white indicates exact zeros.*

THEOREM 2.1. *The implicit approximate inverse* $(\bar{L}\bar{L}^T)^{-1}$ *is fully dense if and only if the ordering is a "reversed graph traversal" (RGT) of* $\bar{L}$*, labelling the root* n*, the next node visited* $n-1$*, then the next one* $n-2$*, and so on.*

For IC(0), where the structure of $\bar{L}$ is the same as the structure of the lower triangle of A, we then are looking for a reversed graph traversal of A. A prime example of this is RCM ordering, where a pseudoperipheral node is chosen as the root, a special breadth-first traversal is made from the root, and the ordering is taken by reversing the traversal so that the root is numbered n.

Parenthetically, this helps to explain why ILU can often give faster convergence than an (explicit) approximate inverse with the same number of nonzeros. A nonfactored sparse approximate inverse is by definition far from dense—for an elliptic PDE this means that the preconditioner can't resolve low-frequency components of the error and is bound to have slow convergence. Factored sparse approximate inverses may have a dense product, but under more restrictive conditions than ILU: the last row/column of the lower/upper triangular factors, respectively, must be dense, which is often not the case.

Another result concerns red-black ordering on a 5-point grid. For IC(0) this is far from being an RGT—every black node is numbered higher than all its neighbors (which are red), and so there are no dipaths leading to a black node. However, for IC(1) each black node is connected in $\bar{L}$ to the eight nearest black nodes, and assuming the usual row-ordering of the black nodes, this does give an RGT. This is one reason why red-black orderings are competitive only at higher fill levels.

Unless special measures are taken, minimum degree, nested dissection, and similar orderings are typically not RGTs for low fill ILU. Figure 2.1 shows an example from a 5-point Laplacian on a 31×31 square grid. Each plot is of a column of the implicit approximate inverse (or the true inverse) appropriately matched to the PDE's domain—i.e., a 31^2 column vector reshaped into a grid function on the 31×31 mesh. These are in essence discretized views of "slices" of the Green's function or its IC approximations. The shading of the squares shows how big the nonzero entries are. It's clear that even if the numerical factorization were optimal, red-black ordered IC(0) and the displayed low fill minimum degree ordered preconditioner must have serious difficulties, whereas the RCM ordered factorization and red-black ordered ILU(1) at least have the structural potential to be very effective.

1	7	4	49	22	28	25
3	8	6	48	24	29	27
2	9	5	47	23	30	26
19	20	21	46	42	41	40
10	18	13	45	31	39	34
12	17	15	44	33	38	36
11	16	14	43	32	37	35

FIG. 2.2. *An example ordering on a* 7×7 *grid that is both a nested dissection and an RGT.*

However, orderings like minimum degree and nested dissection sometimes can give RGTs for low fill ILU. In particular, on square grids it is easy to determine a nested dissection that is an RGT as well—Figure 2.2 gives an example for a 7×7 grid. The key point is that each separator is itself ordered as an RGT with its root adjacent to the highest numbered node from the highest level separator possible. In this way there always exists a path in the underlying graph between any node and node n that does not pass through a lower numbered node than the starting node—in fact, a monotonic path that goes along the separators.

3. Inverse factor structure. For an even better simulation of the application of $A^{-1} = L^{-T}L^{-1}$, one could demand that the structure of the inverses of the incomplete factors match those of the true factors. Our intuition is that this will allow increased global coupling; it can be seen as an intermediate step between the full coupling of the exact factorization and the weakest global coupling discussed in the previous section.

Requiring that $\bar{L}^{-1}$ is as full as L^{-1} is a stronger condition than requiring just that $\bar{L}^{-T}\bar{L}^{-1}$ is as full as A^{-1}, and so this narrows down which graph traversals can be used. In particular, we will observe that this condition is not satisfied by RGT nested dissection orderings with low fill-in, providing some explanation of why they are typically still not as effective as other orderings.

Recall that a depth-first search is a graph traversal where at each step the next node to be visited is chosen to be adjacent to the most recently visited node possible. The subgraph induced by the edges linking each successive node (the child) to the most recently visited adjacent node (its parent) is the associated depth-first search tree.

Define a deepening search to be a topological (or post-order) traversal of a depth-first search tree, i.e., a traversal of a depth-first search tree beginning at the root and visiting each parent node before any of its children. Alternatively put, at each step of a deepening search consider the connected components of unvisited nodes, and select a node from any of those components that is adjacent to the most recently visited node possible for that component. (A depth-first search is a particular type of deepening search.) This leads to the following result.

THEOREM 3.1. *Assume that there is no numerical cancellation and that the structure of the lower triangle of* A *is a subset of the structure of* $\bar{L}$*, which is in turn a subset of the structure of the exact lower triangular factor* L*. Then the structure of* $\bar{L}^{-1}$ *matches the structure of* L^{-1} *if and only if* A *has been ordered according to a reversed deepening search (RDS) of* $\bar{L}$*. In particular, any RDS of* A *fulfills this condition.*

Proof. Recall that the structure of the inverse B^{-1} of some matrix B is given by the transitive closure of the graph of B, which is the same as the transitive closure of the transitive reduction of B [11]. Since the structure of $\bar{L}$ is a subset of the structure

of L, the structure of $\bar{L}^{-1}$ is a subset of the structure of L^{-1}, and is equal to it if and only if $\bar{L}$ contains the transitive reduction of L.

Further recall that the transitive reduction of L is the elimination tree of A [12]. Thus a necessary and sufficient condition for the result is that $\bar{L}$ contain the elimination tree of A.

Note that augmenting the structure of A with any subset of edges from L (i.e., adding "nonzeros" that are actually numerically zero) does not effect the structures of its triangular factors, and hence also does not effect its elimination tree. Thus we can simplify our argument by considering the structure of A augmented with the edges from $\bar{L}$ and $\bar{L}^T$, denoted by $\bar{A}$, instead of A. The necessary and sufficient condition is then reduced to $\bar{A}$ containing its own elimination tree.

Let $E(i)$ denote the set of nodes reachable from i through paths in $\bar{A}$ using only nodes $\{1, \ldots, i-1\}$, or in other words, the nodes reachable from i after all higher numbered nodes have been removed from the graph of $\bar{A}$. Partition each $E(i)$ according to the connected components it induces in the graph of $\bar{A}$, giving subsets $E_1(i), \ldots, E_{f(i)}(i)$.[2] Thus each $E_k(i)$ is a set of nodes numbered less than i that induces a connected subgraph of $\bar{A}$, with at least one node adjacent to i.

As characterized in [2], for example, the nonzeros in row i of L^{-1} correspond to the set $E(i)$. Observe that the elimination tree, the transitive reduction of L^{-1}, has an edge between i and the highest numbered node in each $E_k(i)$, by the following reasoning. Certainly these edges must be contained in the transitive reduction, since there is no equivalent path to them in L^{-1}: L^{-1} is triangular, so there can't be a path from i to the highest numbered node going through lower numbered nodes. These edges also suffice to give the full transitive closure, since it is clear that for each such edge (i, j) with $j = \max(E_k(i))$, we have $E(j) = E_k(i) \setminus \{j\}$, so inductively a path using just these exists from i to any node in $E(i)$.

Therefore, the necessary and sufficient condition becomes that, for each i, there is an edge in $\bar{A}$ from i to each of $\max(E_1(i)), \ldots, \max(E_{f(i)}(i))$. Imagine visiting the nodes in reverse order, starting at n and ending at 1. The condition means that within a connected component of unvisited nodes, the next node to visit must be a neighbor of the most recently visited node adjacent to the component. In other words, the condition is simply that the ordering must be an RDS of $\bar{L}$ (and the associated depth-first search tree is the elimination tree, consisting of the edges between each i and $\max(E_1(i)), \ldots, \max(E_{f(i)}(i))$.

Finally, observe that, barring numerical cancellation, adding more fill to $\bar{L}$ cannot reduce the structure of its inverse. Thus if the ordering is an RDS of the original A, i.e., of an IC(0) factor, then any higher fill incomplete factorizations $\bar{L}\bar{L}^T$ must satisfy the condition. □

An example of an RDS ordering that has already been investigated for ILU (and shown to be fairly effective) is the spiral ordering of [8]. This illustrates a special case of RDS orderings, where the associated depth-first search tree is actually a Hamiltonian path (i.e., it doesn't branch at all) and so the subdiagonal of A is all nonzero, which guarantees that L^{-1} will be a fully dense lower-triangular matrix.

Although RCM orderings are always RGT orderings, they typically are not RDS orderings, at least for low fill IC. However, it should be noted that by rearranging each level set appropriately, they may always be converted into RDS orderings—perhaps at the expense of increasing the envelope, but that is generally inconsequential in

[2] These may be called eliminated elements at step i of a complete factorization, and would correspond to the quotient nodes adjacent to i in the quotient graph [10].

modern sparse matrix packages for iterative methods.

On the other hand, although we observed in the previous section that separator-based fill-reducing orderings such as nested dissection can be made into RGT orderings by reordering separators, it is in general impossible to similarly convert them to RDS orderings. Lower-level separators generally are adjacent to higher-level ones at the wrong places, namely, the middle nodes rather than an end node. This may help to explain why these orderings, even if they are RGTs, generally perform poorly with low fill.[3]

4. Conclusion. The two theorems in this paper provide some understanding of already observed phenomena—the robust performance of RCM orderings, etc. However, we look forward to a constructive use in designing more effective orderings (and fill patterns) for IC. It certainly seems likely that any ordering considered today can be made into an RGT either by adding extra fill or reordering separators, for example, and many can be further made into an RDS. Alhough this is no guarantee of a good quality preconditioner, this at least removes structural impediments and paves the way for an effective incomplete numerical factorization.

Acknowledgments. We would like to thank the anonymous referees for their helpful suggestions in refocusing and clarifying the paper and for some instructive references.

REFERENCES

[1] M. Benzi, D. B. Szyld, and A. van Duin, *Orderings for incomplete factorization preconditioning of nonsymmetric problems*, SIAM J. Sci. Comput., 20 (1999), pp. 1652–1670.
[2] R. Bridson and W.-P. Tang, *Ordering, anisotropy and factored sparse approximate inverses*, SIAM J. Sci. Comput., 21 (1999), pp. 867–882.
[3] S. Clift, H. Simon, and W.-P. Tang, *Spectral Ordering Techniques for Incomplete LU Preconditioners for CG Methods*, manuscript.
[4] S. Clift and W.-P. Tang, *Weighted graph based ordering techniques for preconditioned conjugate gradient methods*, BIT, 35 (1995), pp. 30–47.
[5] E. D'Azevedo, P. Forsyth, and W.-P. Tang, *Towards a cost-effective ILU preconditioner with high level fill*, BIT, 32 (1992), pp. 442–463.
[6] S. Doi, *On parallelism and convergence of incomplete LU factorizations*, Appl. Numer. Math., 7 (1991), pp. 417–436.
[7] I. Duff, A. Erisman, C. Gear, and J. Reid, *Sparsity structure and Gaussian elimination*, SIGNUM Newsletter, 23 (1988), pp. 2–8.
[8] I. Duff and G. Meurant, *The effect of ordering on preconditioned conjugate gradients*, BIT, 29 (1989), pp. 635–637.
[9] V. Eijkhout, *Analysis of parallel incomplete point factorizations*, Linear Algebra Appl., 154/156 (1991), pp. 723–740.
[10] A. George and J. Liu, *Computer Solution of Large Sparse Positive Definite Systems*, Prentice–Hall, Englewood Cliffs, NJ, 1981.
[11] J. Gilbert, *Predicting structure in sparse matrix computations*, SIAM J. Matrix. Anal. Appl., 15 (1994), pp. 62–79.
[12] J. Liu, *The role of elimination trees in sparse factorization*, SIAM J. Matrix Anal. Appl., 11 (1990), pp. 134–172.

[3] We note that for unsymmetric problems, however, the story can be quite different [1].

SIAM J. SCI. COMPUT.
Vol. 22, No. 5, pp. 1533–1548
© 2000 Society for Industrial and Applied Mathematics

ASSESSMENT OF A HIGH RESOLUTION CENTERED SCHEME FOR THE SOLUTION OF HYDRODYNAMICAL SEMICONDUCTOR EQUATIONS*

A. MARCELLO ANILE†, NIKOLAOS NIKIFORAKIS‡, AND ROSA M. PIDATELLA†

Abstract. Hydrodynamical models are suitable to describe carrier transport in submicron semiconductor devices. These models have the form of nonlinear systems of hyperbolic conservation laws with source terms, coupled with Poisson's equation. In this article we examine the suitability of a high resolution centered numerical scheme for the solution of the hyperbolic part of these extended models, in one space dimension. Because of the lack of physically significant exact analytical solutions, the method is assessed against a benchmark for the system of compressible, unsteady Euler equations with source terms, which has an exact solution; the latter is shown to be nearly identical to the numerical one. The method is then used to solve the extended hydrodynamical model (EM) based on the maximum entropy closure recently introduced by Anile, Romano, and Russo, simulating a ballistic diode $n^+ - n - n^+$, which models a metal oxide semiconductor field effect transistor (MOSFET) channel. Results are presented for the reduced- and full-equation EM formulation at steady state, for an initially discontinuous electron density at the junctions. Transient results show the evolution of highly nonlinear waves emanating from the neighborhood of the junctions.

Key words. semiconductors, high resolution schemes, slope limiter centered

AMS subject classifications. 65C20, 65M06

PII. S1064827599361588

1. Introduction.

Enhanced functional integration in modern electron devices requires an accurate modeling of energy transport in semiconductors in order to describe high-field phenomena such as hot-electron propagation, impact ionization, and heat generation in the bulk material. Furthermore, when using compound semiconductors for high frequency applications, usually one deals with multivalley band structures and in these cases the transfer of carriers from one valley to the other must also be modeled. The standard drift-diffusion models cannot cope with high-field phenomena because they do not comprise energy as a dynamical variable. Also they do not incorporate dynamical transfer of carriers from one valley to the other and this renders them ill-suited for simulating time dependent high frequency phenomena. Therefore, generalizations of the drift-diffusion equations have been sought which would incorporate energy as a dynamical variable and which also could treat time dependent high frequency phenomena. Because of their mathematical similarity to the equations of compressible fluid flow, these models are called hydrodynamical models. Semiconductor hydrodynamical models are obtained from the infinite hierarchy of moment equations of the semiclassical Boltzmann transport equation (BTE) by a suitable truncation procedure. This requires making suitable assumptions on (i) choosing the appropriate moments, (ii) closing the hierarchy of moment equations by finding appropriate expressions for the $N + 1$ order moment in terms of the previous

*Received by the editors September 23, 1999; accepted for publication (in revised form) June 15, 2000; published electronically November 17, 2000. This work was partially supported by a CNR grant (96.03855.CT01) and by the European TMR Programme entitled "Asymptotic Methods in Kinetic Theory," contract ERBFMRXCT970157.

http://www.siam.org/journals/sisc/22-5/36158.html

†Dipartimento di Matematica, Universitá di Catania, viale A. Doria 6-95125 Catania, Italy (anile@dipmat.unict.it, rosa@dipmat.unict.it).

‡Department of Applied Mathematics and Theoretical Physics, University of Cambridge, Silver Street, Cambridge CB3 9EW, UK (n.nikiforakis@damtp.cam.ac.uk).

ones, and (iii) modeling the production terms on the right-hand side of the moment equations which arise from the moments of the collision terms in the BTE.

Various closure assumptions have been made for the semiconductor transport moment systems, leading to various classes of hydrodynamical models, e.g., [12, 9, 19, 18]. However, these various closure assumptions are, at best, only phenomenological and lack a consistent physical and mathematical justification.

Lately a closure assumption based on the maximum entropy principle of extended thermodynamics [28, 24] or, equivalently, the method of exponential closures [21] has been applied, in the parabolic band approximation, to the semiconductor moment equations, leading to a semiconductor hydrodynamical model free from phenomenological assumptions and enjoying important mathematical properties such as hyperbolicity [5, 4, 6, 8].

With this closure the distribution function used to calculate the higher order moments is assumed to be the one which maximizes the entropy under the constraints of the given set of moments. The resulting constitutive equations for various moments have been compared with the results obtained by Monte Carlo (MC) simulations in [7, 29] and are very encouraging in support of the maximum entropy ansatz. In these models the production terms are modeled by means of a fitting of the MC data for both homogeneous and inhomogeneous doped semiconductors.

1.1. Previous work on steady-state model integrations. Apart from the usual balance equations for carrier density, momentum, and energy, these extended models (EMs) comprise evolution equations for the heat flux and shear stress. The resulting system is hyperbolic in a suitable domain of the space of variables.

In the stationary case, by neglecting the viscous stresses and linearizing the heat flux equation for small temperature gradients (Maxwellian iteration) one obtains an extension of the Fourier law which includes also a convective term, which we call the Anile and Pennisi (AP) model [5]. When the convective term in the constitutive equation for the heat flow is (incorrectly) neglected, one obtains the model of Blotekjaer [12] and Baccarani and Wordeman [9] (BBW), which is often used in industrial simulation studies. Gardner, Jerome, and Rose [16] and Gardner [17] numerically integrated the BBW model for a benchmark case study (1-dimensional (1D) quasiballistic $n^+ - n - n^+$ diode). They discretized the system of equations by using either central differences (if the flow is everywhere subsonic) or the second upwind method (for transonic flow). The discretized system is then linearized by using Newton's method with a damping factor. In this way Gardner [17] was able to show evidence for an electron shock wave in the diode. A method similar to Gardner's was used by Anile, Maccora, and Pidatella [3] in order to solve the AP model with viscosity included. Gardner's results were also recovered within their approach. A similar approach has been used by Benvenuti, Coughran, and Pinto [10] in order to solve coupled thermal hydrodynamical models, using a Galerkin-type formulation.

Another approach which has been used in order to find the steady state for the BBW or AP model (in 1D and 2D) is that of using an artificial time method and mixed finite element space discretization. The ensuing time dependent numerical solutions are evolved until steady state is reached [27].

1.2. Previous work on time dependent calculations. The above approaches have dealt with the steady-state system of equations. For several practical applications one has to account for dynamic processes like self-heating of the device, coupling with mechanical effects, etc. This requires that at least the time derivatives in the particle and energy balance equations be reinstated. For recent advanced applications, such

as microwave power generation and optoelectronics, the remaining time derivatives in the momentum and energy-flux equations have to be taken into account.

Numerical solutions of various unsteady semiconductor hydrodynamical models have been presented in various articles. The hyperbolic part of the simpler BBW model coincides with the Euler hydrodynamical equations; shock capturing schemes have been adopted for its solution [15], achieving the stationary solutions as the limit of the time dependent one.

Lately Gruzinskis et al. [18] have applied finite difference methods to solve their hydrodynamical formulation which has a phenomenological closure and obtained wave-like solutions.

Blokhin and Iordanovich [11] have presented numerical solutions for the AP model using a flux vector splitting method, marching the algorithm to steady state.

Similar results have been presented in [31], again for the AP equations, using the Nessyahu–Tadmor (NT) scheme [32] for the convective step, which has the advantage over upwind-based schemes that it does not require the knowledge of the characteristic speeds of the system, which are not known analytically in this case. In their paper Romano and Russo [31] recovered the results obtained by Fatemi, Jerome, and Osher [15] in the case of the BBW model. Further work using the NT scheme was carried out by Anile, Romano, and Russo [7] in order to obtain the steady state for the 1D quasiballistic diode by marching in time the unsteady algorithm.

Since there are no known nontrivial analytical solutions of the EMs against which to compare the numerical results, confidence on the numerical results is enhanced by utilizing other unrelated numerical schemes or by solving mathematically similar problems with known solutions from other disciplines. This motivated recent work on kinetic schemes [2] (which gave results identical to those obtained with the NT scheme, although at a higher computational cost) as well as the work presented in this article.

2. Rationale for the current numerical approach. Our intention is to ultimately perform accurate multidimensional calculations of the full time dependent EMs equations to simulate the behavior of realistic devices (bipolar junction transistor (BJT), MOSFET, resonant diodes, etc.) in both transient and steady-state regimes. In this context accuracy means being able to capture small scale wave features (e.g., related to Gunn-type oscillations), as well as the bulk behavior. In this article we have considered the case of a 1D quasiballistic $n^+ - n - n^+$ diode, which is used as a benchmark problem for models of submicron electron devices, since the salient features of its behavior are understood. The modeling of a realistic device of this kind because of the discontinuous doping profile introduces strong gradients in the initial electron density at the junctions. The subsequent evolution of the system gives rise to nonlinear waves, before reaching a steady state. This implies that we must use methods which do not suffer from excess numerical diffusion or spurious oscillations in the vicinity of steep gradients. Also, although there are "source terms," the conservation properties of the hyperbolic left-hand side must be maintained. These requirements point us to the *high resolution* family of methods (see, for example, the textbooks by Hirsch [22], LeVeque [26], or Toro [33]) and in particular to those who lend themselves to be combined readily with computational techniques like adaptive mesh refinement (AMR) and modern computer architectures (such as massively parallel computers (MPP)). A typical method of this family offers a conservative discretization, low numerical diffusion, and no spurious oscillations near steep gradients.

High resolution *upwind* methods are most suitable for the numerical solution of systems of hyperbolic conservation laws because they introduce characteristic informa-

tion (regarding the local directionality of the flow) in the evaluation of the numerical fluxes. A great number of upwind high resolution schemes use the solution of the Riemann problem to evaluate the intercell fluxes. However, when the solution of the Riemann problem is not known, it is possible to construct *centered* schemes which do not compromise the qualities of the high resolution family, albeit at a small loss of "sharpness" of the solution.

The solution is updated by evaluating a finite volume formula derived by considering the integral form of the conservation laws. In particular, for a 1D system of the form

$$\mathbf{U}_t + \mathbf{F}(\mathbf{U})_x = 0, \tag{2.1}$$

where $\mathbf{U}$ and $\mathbf{F}(\mathbf{U})$ are the vectors of the conserved variables and the fluxes, respectively, and the equivalent integral formulation

$$\oint [\mathbf{U}dx + \mathbf{F}(\mathbf{U})dt] = 0, \tag{2.2}$$

the resulting update formula is (see [34])

$$\mathbf{U}_i^{n+1} = \mathbf{U}_i^n + \frac{\Delta t}{\Delta x}[\mathbf{F}_{i-1/2} - \mathbf{F}_{i+1/2}], \tag{2.3}$$

where $\mathbf{U}_i^{n+1}$ and $\mathbf{U}_i^n$ are the solutions at the next and current time-levels, $n+1$ and n, respectively. $\mathbf{F}_{i-1/2}$, $\mathbf{F}_{i+1/2}$ are the numerical fluxes[1] at the interfaces of the computational cells of the discretized space; see Figure 2.1. For a given cell width, Δx, and timestep, Δt, the values of the numerical fluxes need to be evaluated in formula (2.3) in order to compute the conserved variables at the next time-level.

To this end centered schemes can be constructed using a nonlinear combination of a good second (or higher) order scheme with a first order monotone scheme.

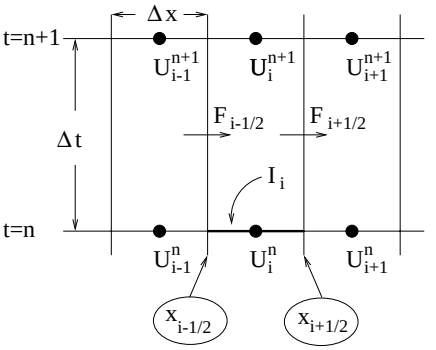


FIG. 2.1. *Geometrical representation of the finite volume update formula.*

2.1. The slope limiter centered scheme. The centered method we use for the calculations presented in this article, namely the slope limiter centered (SLIC) scheme [33], uses a version of the second order MUSCL–Hancock [36] (where MUSCL stands for monotone upstream-centered scheme for conservation laws) and the first-order centered (FORCE) scheme [33]. Since the scheme has been presented elsewhere, we summarize below only its main points, and in particular the ones necessary for our discussion.

[1]The numerical fluxes are approximations to the physical fluxes.

The MUSCL–Hancock approach achieves a second order extension of Godunov's first order scheme by reconstructing the data as piecewise linear functions in every cell. The left and right cell boundary extrapolated values for cell i

$$\mathbf{U}_i^L = \mathbf{U}_i^n - \frac{1}{2}\Delta_i, \qquad \mathbf{U}_i^R = \mathbf{U}_i^n + \frac{1}{2}\Delta_i$$

(where Δ_i is a slope vector), are evolved in time by half a timestep as linear combinations of the conserved variables and the fluxes:

$$\mathbf{U}_i^{Lnew} = \mathbf{U}_i^L + \frac{1}{2}\frac{\Delta t}{\Delta x}[\mathbf{F}(\mathbf{U}_i^L) - \mathbf{F}(\mathbf{U}_i^R)],$$
$$\mathbf{U}_i^{Rnew} = \mathbf{U}_i^R + \frac{1}{2}\frac{\Delta t}{\Delta x}[\mathbf{F}(\mathbf{U}_i^L) - \mathbf{F}(\mathbf{U}_i^R)].$$

However, the intercell fluxes are evaluated using the FORCE scheme, instead of using the solution of the Riemann problem. FORCE is a result of a combination of the Lax–Friedrichs and the Richtmyer schemes (see [33] and references therein). In particular, the flux at the interface of two states U_L, U_R is

$$F_{i+1/2}^{force} \equiv F_{i+1/2}^{force}(U_L, U_R) = 0.5\left[F_{i+1/2}^{LF}(U_L, U_R) + F_{i+1/2}^{Ri}(U_L, U_R)\right]. \tag{2.4}$$

In the above expressions, the first order Lax–Friedrichs flux, F^{LF}, is given by

$$F_{i+1/2}^{LF} \equiv F_{i+1/2}^{LF}(U_L, U_R) = 0.5\,[F(U_L) + F(U_R)] + 0.5\frac{\Delta x}{\Delta t}[U_L + U_R].$$

The second order Richtmyer scheme (which computes a numerical flux by first defining an intemediate state) is

$$U_{i+1/2}^{Ri} \equiv U_{i+1/2}^{Ri}(U_L, U_R) = 0.5(U_L + U_R) + 0.5\frac{\Delta t}{\Delta x}[F(U_L) - F(U_R)],$$

setting $F^{Ri} = F(U_{i+1/2}^{Ri})$.

The resulting scheme is second order accurate in space and time, so to avoid spurious oscillations in the vicinity of steep gradients, the slopes Δ_i are replaced by limited slopes $\bar{\Delta}_i$, using slope limiter functions. For a detailed exposition of the scheme, the slope limiter functions, and validation problems, see the textbook by Toro [33].

An advantage of the scheme is that every flux component is "limited" independently, using the appropriate conserved variable. This is of paramount importance because normally a single function is used to limit all components; since there is no formal theory on the selection of these functions, this can compromise the performance of the scheme, especially if one flux is continuous at a point in space and time, while another is discontinuous at the same point. Even if a suitable global function is empirically found, this will have to be changed if the equations are altered.

From the above discussion it is evident that once a skeleton algorithm for the scheme is coded, any system of hyperbolic conservation laws can be solved simply by typing the vectors of the conserved variables and the fluxes. For our purposes this is very important because the effects of altering the terms in the conservation laws on the physics of the problem can be readily investigated. For example, in this article we solve the full as well as the reduced EM semiconductor equations; see the following sections.

Also, because the scheme uses a conventional finite volume update formula on a conventional computational mesh, it can be implemented in existing computer program for 2D extensions with or without source terms, as explained in the next section.

2.2. Fractional steps. The equations for our problem are of the form

$$\frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{F}(\mathbf{U})}{\partial x} = \mathbf{S}(\mathbf{U}), \tag{2.5}$$

coupled with the Poisson equation

$$\frac{\partial^2 \Phi}{\partial x^2} = S_1. \tag{2.6}$$

We adopt the method of fractional steps to evaluate (2.5) and (2.6), where the homogeneous hyperbolic part

$$\frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{F}}{\partial \mathbf{U}} = 0$$

is solved using SLIC (with initial and boundary conditions as specified for the complete system), and then we evaluate the remaining ordinary differential equation

$$\frac{\partial \mathbf{U}}{\partial t} = S(\mathbf{U})$$

and the Poisson equation (2.6) using conventional techniques (e.g., Runge–Kutta and tridiagonal matrix solver). Calling R the relaxation step operator, C the convection step operator, and P the Poisson operator, and assuming that the numerical schemes used are at least second order accurate, a Strang splitting [30] which maintains this level of accuracy is

$$\mathbf{U}^{n+1} = R(\Delta t/2)P(\Delta t/2)C(\Delta t)P(\Delta t/2)R(\Delta t/2)\mathbf{U}^n.$$

The accuracy of this splitting and the performance of SLIC for arbitrary source-term driven flow is evaluated using a combustion problem which has an exact solution.

3. A validation case study. The scheme has been validated in open literature only for homogeneous systems of equations [33]. Before attempting to implement the semiconductor equations, we validate the scheme against exact solutions of the compressible unsteady Euler equations with source terms using a simplified combustion problem described in some detail by Clarke [13]. It considers flow generated by the action of source terms and is rich in wave structures similar to the ones anticipated in the unsteady evolution of electron flow in semiconductor devices. The governing equations are of the form

$$\mathbf{U}_t + \mathbf{F}(\mathbf{U})_x = \mathbf{S}(\mathbf{U}),$$

where the vectors $\mathbf{U}$, $\mathbf{F}(\mathbf{U})$, and $\mathbf{S}(\mathbf{U})$ are

$$\mathbf{U} = \begin{bmatrix} \rho \\ \rho u \\ E \end{bmatrix}, \qquad \mathbf{F}(\mathbf{U}) = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ (E+p)u \end{bmatrix}, \qquad \mathbf{S}(\mathbf{U}) = \begin{bmatrix} G \\ F \\ H \end{bmatrix}. \tag{3.1}$$

The right-hand sides G, F, and H represent general sources of mass, momentum, and energy, while the last two can be written as functions of G:

$$F = uG, \qquad H = (E + p/\rho)G.$$

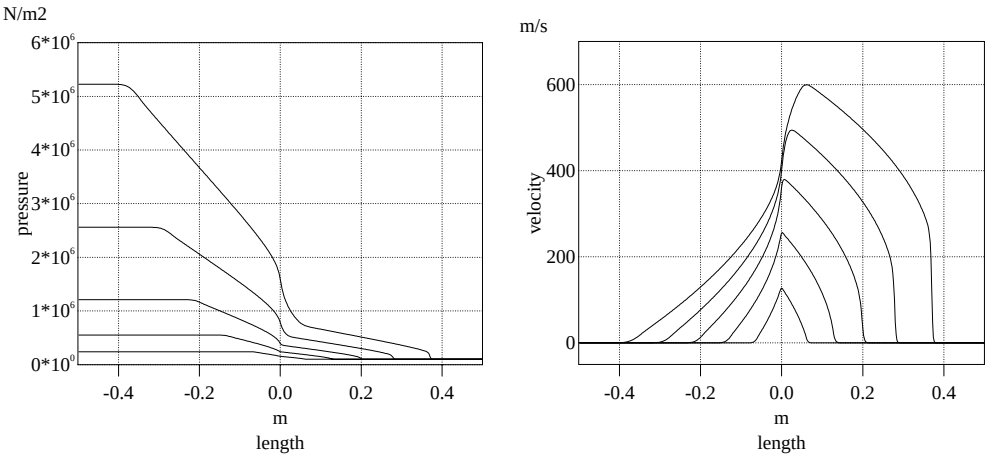


FIG. 3.1. *Superimposed numerical (using SLIC) profiles for pressure and velocity at different times, showing the evolution of the flow for the combustion problem formulated in section* 3. *An expansion wave propagates to the left in a region of increasing pressure, while a compression wave propagates to the right, eventually turning to a shock wave. Sonic flow is observed at the middle of the domain.*

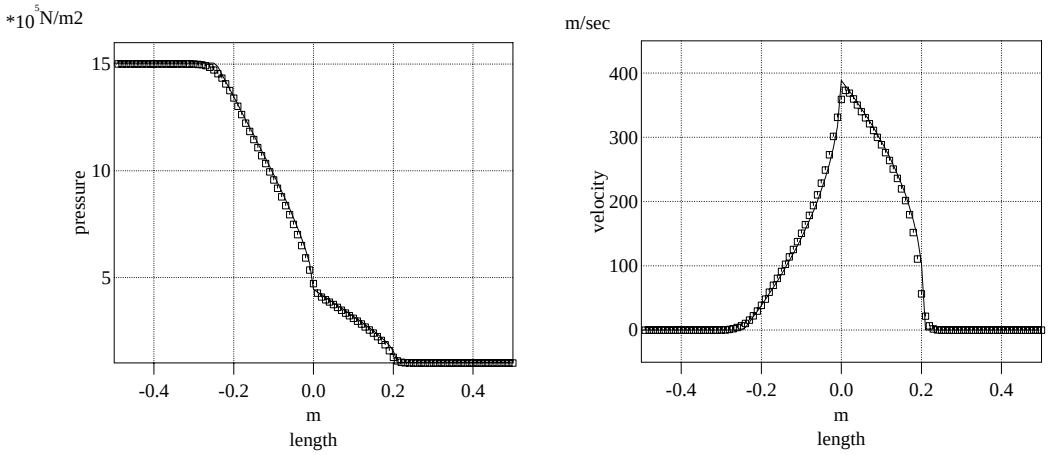


FIG. 3.2. *Superimposed exact (line) and numerical (points) solutions using SLIC for the combustion problem formulated in section* 3.

An exact solution can be derived if the source term G is chosen to be such that

$$(aG/\rho) = \bar{G} = \text{ const. in } x < 0, \text{ and } \bar{G} = 0 \text{ in } x > 0.$$

Part of the exact solution can be found in the paper by Clarke and Toro [14]. Initially pressure is one atmosphere, $u = 0$, and $a = a_0$, while the values of the constants are $a_0 = 330\text{m/sec}$ and $\bar{G} = 1294301.0\text{m/sec}^2$.

Profiles of pressure and velocity at different times, mapping the evolution of the flow, are shown in Figure 3.1; these results were obtained using SLIC. The source-driven flow is isentropic up to the time of shock formation ($t = (4a_0)/((\gamma + 1)\bar{G})$) at the head of the wave propagating to the right. Sonic flow conditions appear at $x = 0$ at $t = (4a_0)/((3 - \gamma)\bar{G})$.

Salient features of interest in the context of scheme validation are the rising pressure and the expansion wave in the left (source-filled) part of the domain and the propagating pressure wave in the right part of the domain, which steepens to form a shock wave. An interesting feature of this problem is that the flow "chokes" in the middle of the domain when sonic conditions appear, as indicated by the velocity curves which reach a limit at $x = 0$ and cusp to the right of the domain.

The accuracy of the method and the correct coupling of the hyperbolic and source-term module is demonstrated in Figure 3.2, which shows superimposed exact (lines) and numerical (points) results at $t = 0.505 \times 10^{-3}$; the two solutions are nearly identical.

4. Semiconductor equations. The EMs which we consider here were formulated by Anile, Romano, and Russo [7, 6]. The equations are of the form

$$\mathbf{U}_t + \mathbf{F}(\mathbf{U})_x = \mathbf{S}(\mathbf{U}),$$

where the vectors $U, F(U)$, and $S(U)$ are

$$\mathbf{U} = \begin{bmatrix} n \\ nv \\ nv^2 + 3p/m_* \\ \frac{2}{3}nv^2 + \sigma/m_* \\ nv^3 + 5vp/m_* + 2\sigma v/m_* + 2q/m_* \end{bmatrix}, \tag{4.1}$$

$$\mathbf{F}(\mathbf{U}) = \begin{bmatrix} nv \\ nv^2 + p/m_* + \sigma/m_* \\ nv^3 + 5vp/m_* + 2\sigma v/m_* + 2q/m_* \\ \frac{2}{3}nv^3 + \frac{4}{3}vp/m_* + \frac{7}{3}v\sigma/m_* + \frac{8}{15}q/m_* \\ nv^4 + 5p^2/n(m_*)^2 + 7\sigma p/n(m_*)^2 + \frac{32}{5}qv/m_* + \\ \qquad v^2(8p/m_* + 5\sigma/m_*) + \frac{148}{25}q^2/m_* p \end{bmatrix}, \tag{4.2}$$

$$\mathbf{S}(\mathbf{U}) = \begin{bmatrix} 0 \\ -nv/\tau_p - neE/m_* \\ -2(W - W_0)/m_*\tau_W - 2nevE/m_* \\ -1/m_*\tau_\sigma(nv^2 + \sigma/m_*) - 4nevE/3/m_* \\ -1/\tau_q(nv^3 + 5vp/m_* + 2\sigma v/m_* + 2q/m_*) \\ \qquad -eE/m_*(3nv^2 + 5p/m_* + 2\sigma/m_*) \end{bmatrix}. \tag{4.3}$$

Depending on the strength of the electric field, some of the nonlinear terms in the deviation from local thermal equilibrium can be neglected, as well as the anisotropic stresses. Thereby one obtains the following set of equations, which we refer to as the reduced EM. In this case the vectors $\mathbf{U}$, $\mathbf{F}(\mathbf{U})$, and $\mathbf{S}(\mathbf{U})$ are

$$\mathbf{U} = \begin{bmatrix} n \\ nv \\ 3p/m_* \\ 2q/m_* \end{bmatrix}, \tag{4.4}$$

$$\mathbf{F}(\mathbf{U}) = \begin{bmatrix} nv \\ p/m_* \\ 2q/m_* \\ 5p^2/n(m_*)^2 \end{bmatrix}, \tag{4.5}$$

$$\mathbf{S}(\mathbf{U}) = \begin{bmatrix} 0 \\ -nv/\tau_p - neE/m_* \\ -2(W - W_0)/m_*\tau_W - 2nevE/m_* \\ -1/\tau_q(2q/m_*) - 5eEp/(m_*)^2 \end{bmatrix}. \tag{4.6}$$

We remark that for the reduced model the interpretation of q is not that of heat flux but of total energy flux.

In both cases we solve Poisson's equation:

$$\varepsilon \frac{\partial^2 \Phi}{\partial x^2} = -e(N_D - N_A - n), \qquad E = -\frac{\partial \Phi}{\partial x}. \tag{4.7}$$

Here n is the electron density, v is the electron velocity, p is the electron fluid pressure, m_* is the effective electron mass (taken to be 0.32 me), σ is the anisotropic stress, q is the heat flux, τ_p is the relaxation time for momentum, τ_w is the relaxation time for energy, τ_σ stands for anisotropic stresses, τ_q is the relaxation time for the energy flux, e is the absolute value of the electron charge, E is the electric field, W is the energy density, $W = (1/2)m_*v^2 + (3/2)K_BT$, and W_0 is the thermal equilibrium energy density.

In both cases the relaxation times are obtained as functions of energy W from fitting to MC simulation for the same benchmark device (see [29]). The reduced model resembles strongly the so-called energy-transport models obtained from the semiclassical BTE by a Chapman–Enskog-like procedure [1].

5. A benchmark case study. As a test problem we consider a ballistic diode $n^+ - n - n^+$, which models a MOSFET channel. The diode is made of silicon, and the bulk temperature is supposed to be 300°K. The n^+ regions are 0.1μm long and the channel length L_c we consider is 0.4μm. The doping profile is $N_D^+ = 1. \times 10^{18}$ cm^{-3}, $N_D = 0.01 \times 10^{18}$ cm^{-3}.

For the electron effective mass in the approximation of parabolic band we use $m_* = 0.32\ m_e$, where m_e is the electron mass [35]. The silicon dielectric constant is given by $\epsilon = \epsilon_r\epsilon_0$, where ϵ_r =11.7 is the relative dielectric constant and $\epsilon_0 = 8.85 \times 10^{-12} F/m$ μm is the dielectric constant of vacuum.

The initial electron temperature is the lattice temperature $T_0 = 300$°K, and the charges are at rest. A bias voltage of 1 volt is applied and this determines a charge flux in the semiconductor.

Solutions for the full and the reduced formulation of the equations are shown in this section. In both cases all of the variables are assumed to be initially constant across the computational domain, save for the density of the electrons, which varies discontinuously at the junctions

$$T(x,0) = 300°\text{K}, \quad v(x,0) = 0, \quad q(x,0) = 0, \quad \sigma(x,0) = 0.$$

The initial density profile is the doping profile, which is an inverted top hat. The total length of the device is $L = L_c + 0.2\mu$m. Transmissive boundary conditions are

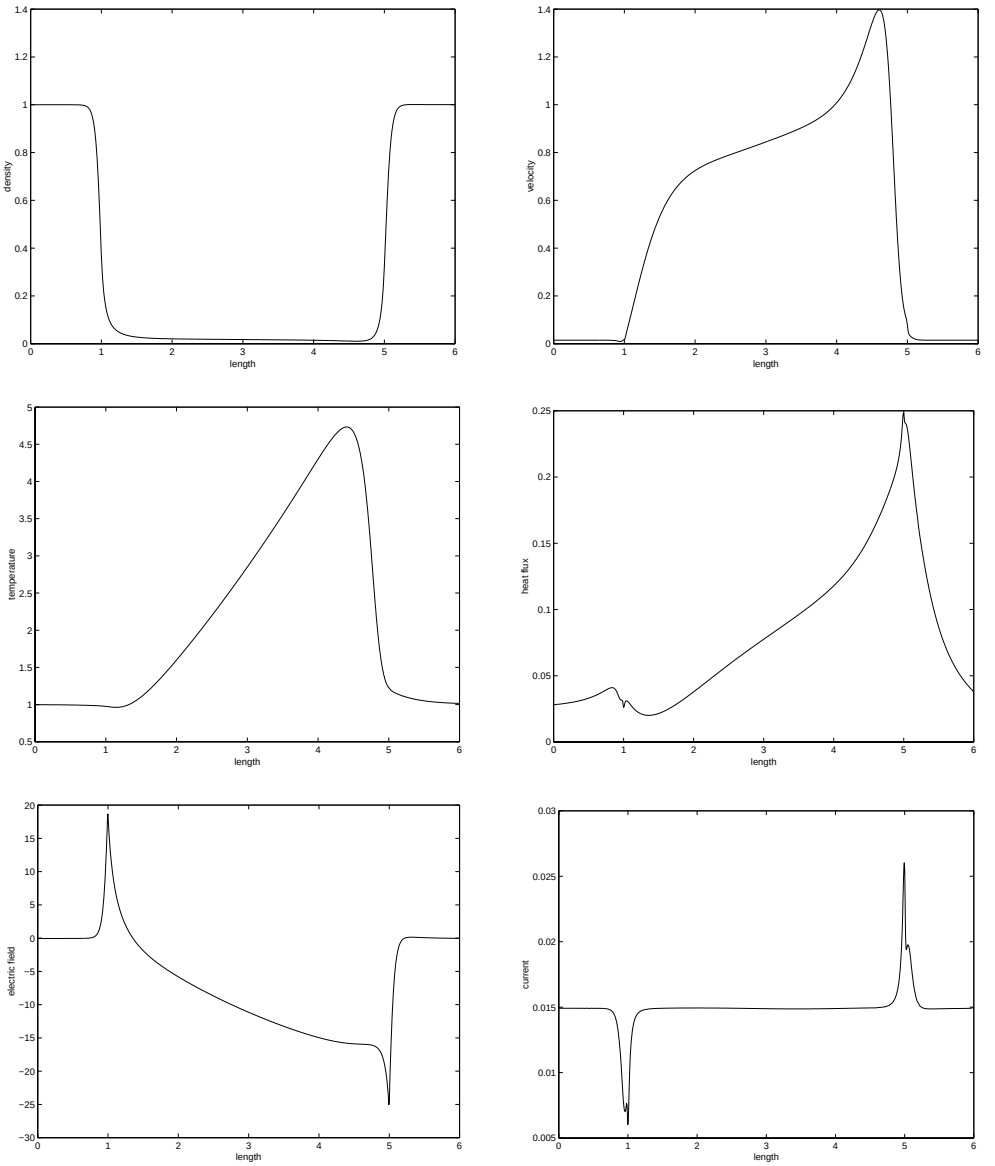


FIG. 5.1. *Numerical solutions using SLIC method solving the reduced model* $n^+ - n - n^+$ *diode formulation. The initial density distribution was discontinuous at the location of the junctions.*

applied at both ends of the computational domain. The boundary conditions for the Poisson's equation are

$$e\Phi(x_{left}) = K_B T ln(N_D/n_i), \quad e\Phi(x_{right}) = K_B T ln(N_D/n_i) + eV_{bias},$$

where n_i =1.45$\times$ 10^{10}cm^{-3} is the intrinsic concentration.

The system is left to evolve, preserving time accuracy, until steady state is reached (results shown at $t = 5$ picoseconds). The hyperbolic and the source terms are allowed to evolve at different timesteps, which are, however, matched at the end of every iteration. In this way the lower timestep of the source terms does not burden

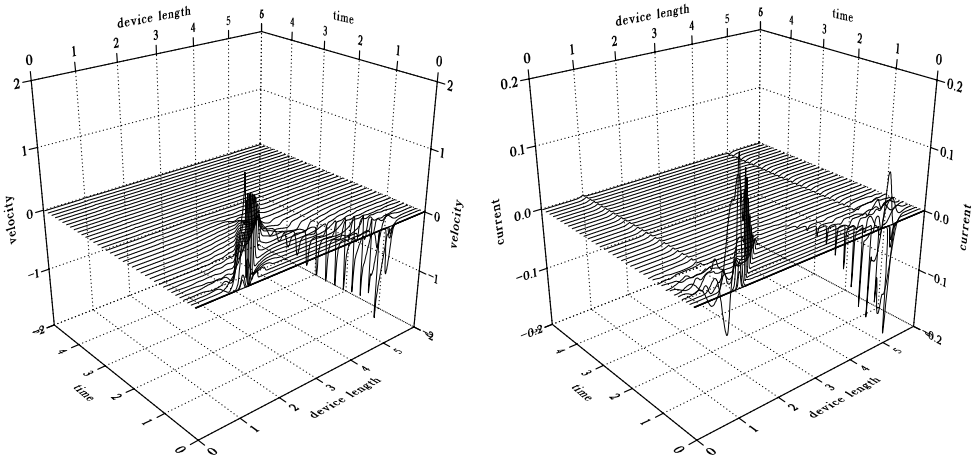


FIG. 5.2. *SLIC method solving the full EM for zero bias (discontinuous initial density). After the initial waves attenuate, velocity reaches a steady-state value of nearly zero (compared with the initial magnitude of the waves), while there is evidence of a small value of current at the location of the junctions.*

the speed of execution of the complete code. It should be noted here that apart from the efficiency issue, marching any method for hyperbolic conservation laws at a timestep smaller than the optimum one, as dictated by the Courant–Friedrichs–Levy (CFL) condition, significantly increases the truncation error of the calculation, which manifests as excessive numerical diffusion.

The thickness of the junctions is of importance to the calculations because, as we will demonstrate in a future communication, it affects some of the salient features in the steady-state profiles of the primitive variables. In any case, the method allows one to vary the junction thickness down to the width of a computational cell, i.e., a genuine discontinuous profile of the initial density, which can match the true physical width of a real device where the doping is obtained by epitaxial growth (and not by ion diffusion). To the best of our knowledge, in all other published results the junctions are smeared over several computational cells and therefore, the resulting simulations apply only to the cases in which the initial profile is obtained by some sort of diffusion.

5.1. Reduced EM calculations. The qualitative behavior of the solutions is the same as for the "energy transport" models studied by several researchers [1, 27, 25] under steady-state conditions. The new feature of our approach is that we use a method suitable for hyperbolic systems with source terms, and in this particular case study we evolved the solution as an unsteady problem marched to steady state, preserving time as well as space accuracy. Therefore, our code could be used to investigate the transient behavior of the system from an academic and an engineering point of view. The results of the simulations are shown in Figure 5.1; the initial doping profile is discontinuous at the junctions and the qualitative behavior of the fields is what is expected on physical grounds. Similar results have been obtained where the initial doping is smoothed with a convolution or hyperbolic tangent. In these latter cases the results have been compared with those obtained by a direct solution of the stationary equations using mixed finite elements [37]; the two sets of results agree completely within computer arithmetic. It appears that the current is

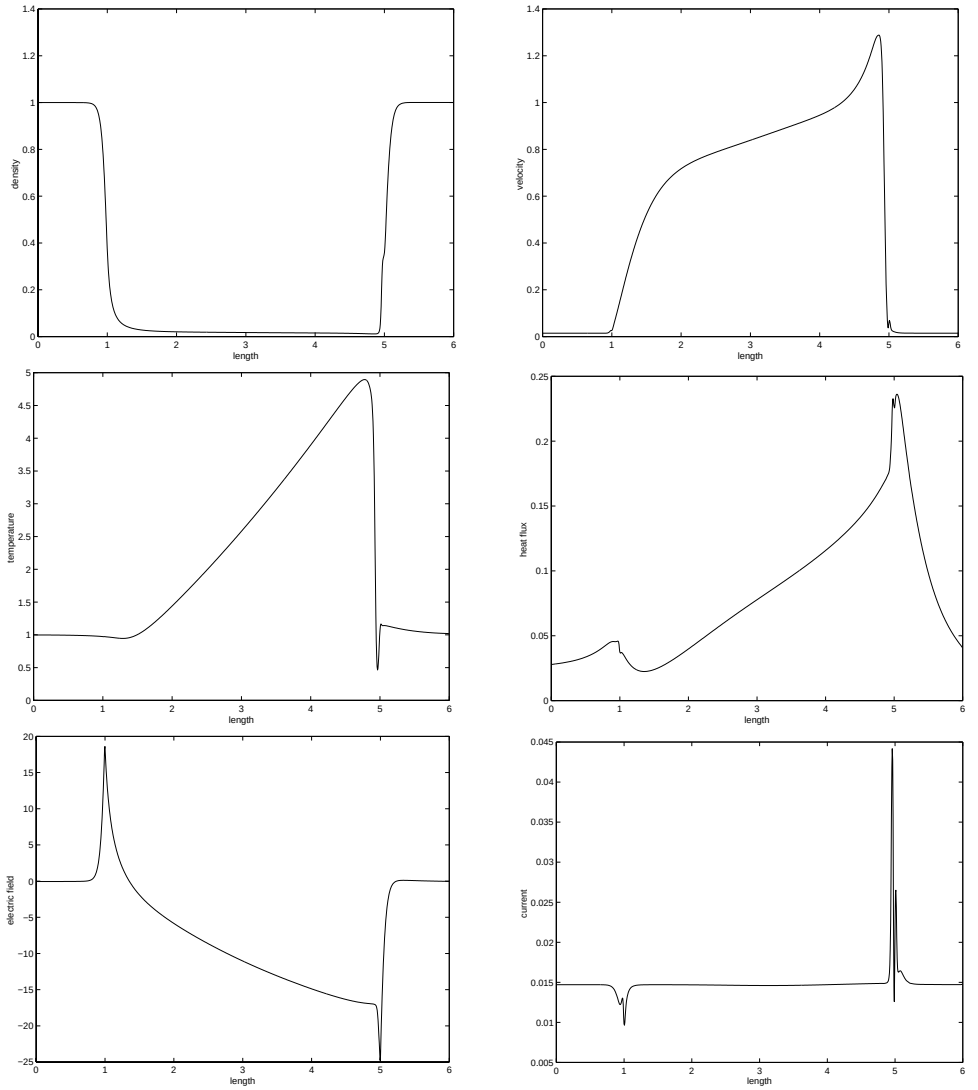


FIG. 5.3. *SLIC method solving the full EM for* 1 *volt bias (discontinuous initial density); steady-state profiles for various physical variables.*

not exactly constant in the vicinity of the junctions. Previous studies have shown that the magnitude of the error is related to the order of accuracy of the method and, consequently, to the width of the computational cells, i.e., the error can be decreased by using a method of higher order of accuracy. Alternatively, for a given order of accuracy, the error can be decreased by increasing the number of computational cells. The error apparent from Figure 5.1 is of the same order as for similar second order methods and tends to vanish as the number of computational cells increases.

5.2. Full EM calculations. The full system of equations was integrated at a zero and a finite-value bias. The former can be used to calculate the junction capacitance and the voltage across the junction. Also, since the velocity and current must go to zero, it also serves as an indication of the accuracy of the numerical model.

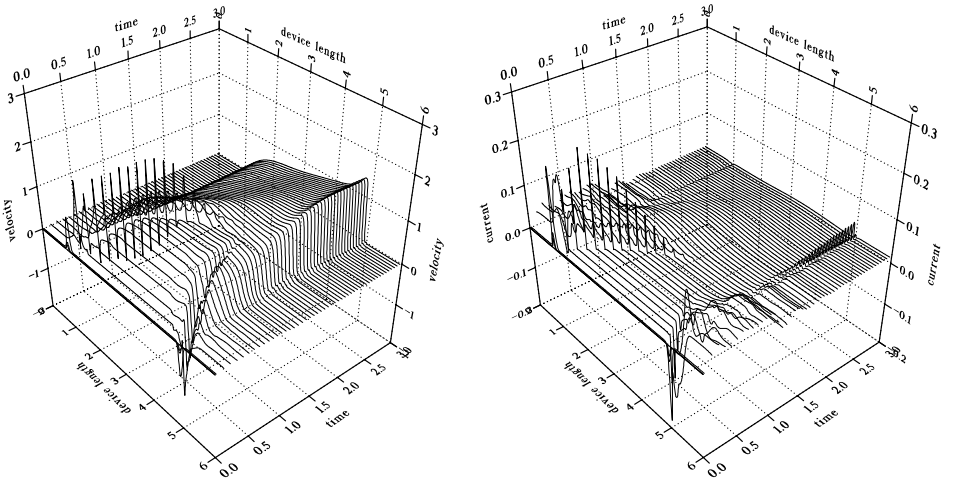


FIG. 5.4. *SLIC method solving the full EM for* 1 *V bias (discontinuous initial density); transient profiles for velocity and current.*

The evolution of the field can be seen in Figure 5.2; electrons expand in both directions of the channel, creating electron waves which reach steady state by $t = 3$ picoseconds. There are no oscillations present in the velocity field, as was reported in the paper by Fatemi, Jerome, and Osher [15]. There is, however, a residual value of current at the location of the junctions, albeit of an order of magnitude smaller than the maximum value reached during this test.

A bias of 1 volt was then applied across the device. Figure 5.3 shows the distribution of the variables at steady state (t=5 picoseconds), while results as a function of time are shown in Figure 5.4. The steady-state plots show no spurious oscillations near steep gradients of the field variables. The small unphysical negative velocity usually observed in the region of the left junction is absent from these calculations. The "bump" usually observed in the vicinity of the right junction is just visible, but not pronounced; initial results suggest that its magnitude and spacial extent is a function of the initial thickness of the junction (the wider the junction, the more pronounced the bump is).

The time-evolution plots show that during the initial stages of the evolution, the strong shock waves (emanating from the locations of the initial density discontinuities) propagate from the junctions towards the middle of the channel. At approximately t =1.5 they begin to attenuate; the wave propagating from the left adds to the constant increase of the velocity in the middle of the channel, to form the "ramp" of the steady-state velocity profile, while the one from the right contributes to the formation of the pronounced "spike." The evolution of the latter coincides with the formation of a spike in the current distribution, which persists even after steady state has been reached. If the transient behavior is to be studied in detail, the existence of these strong shock waves necessitates the use of TVD methods.

These results compare favorably to the ones presented in the paper by Anile and Muscato [4], where a direct solution of the steady-state equations is reached using a method similar to Gardner's [17]. A similar calculation for the transients has been performed by Jerome and Shu [23], albeit for a different model (Baccarani and Wordeman [9]) using an essentially nonoscillatory (ENO) scheme. In both case stud-

ies nonlinear waves arise, but detailed comparison is not meaningful because of the substantial differences in the models.

By comparing the results of the reduced model and of the full model with the same physical and computational parameters, we notice that they differ only slightly (the reduced model shows a higher velocity peak and a lower energy peak), but otherwise the bulk features are essentially the same. Therefore, since the full model presents a more significant computational challenge (in terms of the singularities arising from the nonlinearity and the added computational expense), the reduced model can be used for parametric studies, and once the relevant parameters have been estimated, more detailed simulations can be carried out using the full model.

6. Conclusions. A second order scheme for the solution of hyperbolic conservation laws, namely SLIC, has been implemented to solve the extended hydrodynamical semiconductor equations. Other conventional methods have been used to evaluate the source terms and the Poisson equation which is coupled with this system. The scheme is second order accurate in space and time, and it belongs to the high-resolution class of methods. As it is expected from schemes of this class, it is conservative, monotone, and it can resolve discontinuous solutions over very few computational cells.

Our motivation to use this scheme stems from its low computational cost, ease of implementation, and lack of arbitrary, user-adjusted parameters. The current implementation has an automatic selection of the optimum timestep, so that every iteration remains as close as possible to the scheme's stability limit, thus reducing truncation errors and keeping the CPU time as low as possible. Test problems which have an exact solution show that SLIC retains these qualities when used for the integration of a system as complex as the extended hydrodyamical semiconductor equations. Because the scheme does not have user-adjusted parameters and it does not need a special computational stencil, it can be used for multidimensional discretizations on domains of arbitrary geometry and can also be used in conjuction with adaptive mesh refinement software without any implementation complications.

The latest developments in semiconductor technology for ultrafast phenomena in electron devices require new approaches for their numerical simulation. In this article we have implemented a method used in other disciplines which deal with highly nonlinear phenomena and have shown that it is effective in accurately capturing the transient evolution of the electron flow to steady state in a submicron diode.

REFERENCES

[1] N. Ben Abdallah and P. Degond, *On a hierarchy of macroscopic models for semiconductors*, J. Math. Phys., 37 (1996), pp. 3306–3333.

[2] A. M. Anile, M. Junk, V. Romano, and G. Russo, *Cross-validation of numerical schemes for extended hydrodynamical models of semiconductors*, M^3AS, 10 (2000), pp. 833–861.

[3] A. M. Anile, C. Maccora, and R. M. Pidatella, *Simulation of $n^+ - n - n^+$ device by a hydrodynamic model: Subsonic and supersonic flow*, COMPEL, 7 (1994), pp. 1–18.

[4] A. M. Anile and O. Muscato, *Improved hydrodynamical model for carrier transport in semiconductors*, Phys. Rev. B, 51 (1995), pp. 16728–16740.

[5] A. M. Anile and S. Pennisi, *Thermodynamic derivation of the hydrodynamical model for charge transport in semiconductors*, Phys. Rev. B, 46 (1992), pp. 13186–13193.

[6] A. M. Anile, V. Romano, and G. Russo, *Hyperbolic Hydrodynamical Model of Carrier Transport in Semiconductors*, VLSI Design, 8 (1998), pp. 521–526.

[7] A. M. Anile, V. Romano, and G. Russo, *Extended hydrodynamical model of carrier transport in semiconductors*, SIAM J. Appl. Math., 61 (2000), pp. 74–101.

[8] A. M. Anile and M. Trovato, *Nonlinear closures for hydrodynamical semiconductors transport models*, Phys. Lett. A, 230 (1997), pp. 387–395.

[9] G. Baccarani and M. R. Wordeman, *An investigation on steady-state velocity overshoot in Silicon*, Solid-state Electronics, 29 (1982), pp. 970–977.

[10] A. Benvenuti, W. M. Coughran, Jr., and M. R. Pinto, *A thermally-fully hydrodynamical model for semiconductor devices and applications to* III-V *HBT simulation*, IEEE Trans. Electron. Devices, 44 (1997), pp. 1349–1359.

[11] A. M. Blokhin and A. A. Iordanovich, *Numerical investigation of a gas dynamic model for charge transport in semiconductors*, COMPEL, 18 (1999), pp. 6–37.

[12] K. Blotekjaer, *Transport equations for electron in two-valley semiconductors*, IEEE Trans. Electron. Devices, ED-17 (1970), pp. 38–47.

[13] J. F. Clarke, *Compressible Flow Produced by Distributed Sources of Mass: An Exact Solution*, Technical Report CoA Report 8710, College of Aeronautics, Cranfield Institute of Technology, Cranfield, UK, 1987.

[14] J. F. Clarke and E. F. Toro, *Gas flow generated by solid-propellant burning*, in Proceedings of the International Conference on Numerical Simulation of Combustion Phenomena, INRIA, Sophia Antipolis, France, R. Glowinski, B. Larrouturou, and R. Teman, eds., Lecture Notes in Phys. 241, Springer-Verlag, New York, 1985, pp. 192–205.

[15] E. Fatemi, J. Jerome, and S. Osher, *Solution of hydrodynamic device model using high-order nonoscillatory shock capturing algorithms*, IEEE Trans. Computer-Aided Design, 10 (1991), pp. 232–244.

[16] C. L. Gardner, J. W. Jerome, and D. J. Rose, *Numerical methods for the hydrodynamic device model: Subsonic flow*, IEEE Trans. Computer-Aided Design, 8 (1989), pp. 501–507.

[17] C. L. Gardner, *Numerical simulation of a steady-state electron shock wave in a submicrometer semiconductor device*, IEEE Trans. Electron. Devices, 38 (1991), pp. 392–398.

[18] V. Gruzinski, E. Starikov, P. Shiktorov, L. Reggiani, and L. Varani, *Linear and nonlinear analysis of microwave power generator in sub-micrometer* $n^+ - n - n^+$ *InP diodes*, J. Appl. Phys., 76 (1994), pp. 5260–5271.

[19] W. Haensch, *The Drift-Diffusion Equation and its Application in MOSFET Modeling*, Springer-Verlag, Wien, 1991.

[20] A. Harten and S. Osher, *Uniformly high-order accurate nonoscillatory schemes*, SIAM J. Numer. Anal., 24 (1987), pp. 279–309.

[21] C. D. Levermore, *Moment closure hierarchies for kinetic theories*, J. Statist. Phys., 83 (1996), pp. 331–407.

[22] C. Hirsch, *Numerical Computation of Internal and External Flows, Vol.* I, Wiley, London, 1988.

[23] J. W. Jerome and C. W. Shu, *Energy models for one carrier transport in semiconductor devices*, IMA Vol. Math. Appl. 59, W. M. Coughran, Jr., J. Cole, P. Lloyd, and J. K. White, eds., Springer-Verlag, New York, 1994, pp. 185–207.

[24] D. Jou, J. Casas-Vazquez, and G. Lebon, *Extended Irreversible Thermodynamics*, Springer-Verlag, Berlin, 1993.

[25] A. Jungel and C. Pohl, *Numerical simulation of semiconductor devices: Energy transport and quantum hydrodynamic models*, in Proceedings of the Sixth International Workshop on Computational Electronics, Osaka, Japan, IEEE, Los Alamitos, CA, 1998, pp. 230–233.

[26] R. J. LeVeque, *Numerical Methods for Conservation Laws*, Birkhäuser-Verlag, Basel, 1992.

[27] A. Marrocco and Ph. Montarnal, *Simulations des modèles "energy transport" à l'aide des éléments finis mixtes*, C. R. Acad. Sci. Paris Sér. I Math., 323 (1996), pp. 535–541.

[28] I. Müller and T. Ruggeri, *Extended Thermodynamics*, Springer Tracts in Natural Philosophy 37, Springer-Verlag, New York, 1991.

[29] O. Muscato, R. M. Pidatella, and M. Fischetti, *Monte-Carlo and hydrodynamics simulation of a one-dimensional* $n^+ - n - n^+$ *silicon diode*, Electronics VLSI Design, 6 (1998), pp. 247–250.

[30] G. Strang, *Accurate partial difference methods.* II. *Non-linear problems*, Numer. Math., 6 (1964), pp. 37–46.

[31] V. Romano and G. Russo, *Numerical solutions for hydrodynamical models of semiconductors*, M^3AS, in press.

[32] H. Nessyahu and E. Tadmor, *Non-oscillatory central differencing for hyperbolic conservation laws*, J. Comput. Phys., 87 (1990), pp. 408–463.

[33] E. F. Toro, *Riemann Solvers and Numerical Methods for Fluid Dynamics*, Springer-Verlag, Berlin, 1997.

[34] E. F. Toro and S. J. Billett, *Centred TVD Schemes for Hyperbolic Conservation Laws*, IMA J. Numer. Anal., 20 (2000), pp. 47-79.

[35] T. Vogelsang and W. Haensch, *A novel approach for including band structure effects in a Monte-Carlo simulation of electron transport in silicon*, J. Appl. Phys., 70 (1991), pp. 1493–1499.
[36] B. Van Leer, *On the relation between the upwind-differencing schemes of Godunov, Enguist-Osher and Roe*, SIAM J. Sci. Stat. Comput., 5 (1984), pp. 1–20.
[37] A. M. Anile and P. Pietra, *private communication*, 1999.

SIAM J. SCI. COMPUT.
Vol. 22, No. 5, pp. 1549–1569
© 2000 Society for Industrial and Applied Mathematics

A LEGENDRE SPECTRAL GALERKIN METHOD FOR THE BIHARMONIC DIRICHLET PROBLEM*

BERNARD BIALECKI† AND ANDREAS KARAGEORGHIS‡

Abstract. A Legendre spectral Galerkin method is presented for the solution of the biharmonic Dirichlet problem on a square. The solution and its Laplacian are approximated using the set of basis functions suggested by Shen, which are linear combinations of two Legendre polynomials. A Schur complement approach is used to reduce the resulting linear system to one involving the approximation of the Laplacian of the solution on the two vertical sides of the square. The Schur complement system is solved by a preconditioned conjugate gradient method or the Cholesky method. The total cost of the algorithm is $O(N^3)$. Numerical results demonstrate the spectral convergence of the method.

Key words. biharmonic Dirichlet problem, spectral Galerkin method, Schur complement matrix, preconditioned conjugate gradient method, Cholesky method

AMS subject classifications. 65N35, 65N22

PII. S1064827598342407

1. Introduction.

The aim of this work is to propose a Legendre spectral Galerkin method for the solution of the biharmonic Dirichlet problem. This problem has been the subject of several studies in recent years. Our work is related to the papers of Shen [7] and Bjørstad and Tjøstheim [3], whose approach is based on the standard variational formulation of the fourth order differential equation. Our approach is based on the mixed method of Ciarlet and Raviart [4] which gives rise to a variational formulation for two second order differential equations. A similar approach has been recently applied in the Legendre spectral collocation solution of the same problem [2].

The formulation of the biharmonic Legendre spectral Galerkin problem and the method of solution are similar to those developed in [5] for the finite element Galerkin method with piecewise Hermite bicubics.

In this study we consider the biharmonic Dirichlet problem

$$\Delta^2 u = f \text{ in } \Omega, \qquad u = \partial u/\partial n = 0 \text{ on } \partial\Omega, \tag{1.1}$$

where Δ denotes the Laplacian, $\Omega = (-1,1)\times(-1,1)$, $\partial\Omega$ is the boundary of Ω, and $\partial/\partial n$ is the outward normal derivative on $\partial\Omega$.

We set $v = \Delta u$ and discretize a coupled pair of Poisson's equations in u and v using a Galerkin method with polynomials of degree $\leq N$. Employing a Schur complement approach, we reduce the Galerkin problem to a Schur complement system involving an approximation to v on the two vertical sides of $\partial\Omega$ and an auxiliary Galerkin problem for a related biharmonic problem with v, instead of $\partial u/\partial n$, specified on the two vertical sides of $\partial\Omega$. The Schur complement system with a symmetric positive definite matrix is solved by the preconditioned conjugate gradient (PCG) method

*Received by the editors July 27, 1998; accepted for publication (in revised form) June 22, 2000; published electronically November 17, 2000.

http://www.siam.org/journals/sisc/22-5/34240.html

†Department of Mathemical and Computer Sciences, Colorado School of Mines, Golden, CO 80401 (bbialeck@mines.edu). The work of this author was supported in part by National Science Foundation grant DMS-9805827.

‡Department of Mathematics and Statistics, University of Cyprus, P.O. Box 20537, 1678 Nicosia, Cyprus (andreask@ucy.ac.cy). Parts of this author's work were performed while he was a Visiting Associate Professor in the Department of Mathematical and Computer Sciences, Colorado School of Mines, Golden, CO. Support from the University of Cyprus is also gratefully acknowledged.

or the Cholesky method. A preconditioner for PCG is obtained from the Galerkin problem for a related biharmonic problem with v instead of $\partial u/\partial n$ specified on the two horizontal sides of $\partial\Omega$. We conjecture that the preconditioner is spectrally equivalent to the Schur complement matrix. The cost of multiplying the Schur complement matrix by a vector is $4N^2 + O(N)$ and the cost of solving a linear system with the preconditioner is $O(N)$. With the number of PCG iterations equal to $c \log N$, the dominant cost of solving the Schur complement system is $c4N^2 \log N$. The solution of the auxiliary Galerkin problem is obtained at a cost $2N^3 + O(N^2)$ using separation of variables and the solution of a simple eigenvalue problem which reduces to two symmetric eigenvalue problems with tridiagonal matrices. Hence the dominant cost of our PCG algorithm is $2N^3 + c4N^2 \log N$. The dominant cost of our Cholesky algorithm is $8N^3/3$. Both algorithms are well suited for parallel implementation since many of their steps involve independent matrix-vector multiplications. Numerical results demonstrate the spectral convergence rate of the approximations to u and v in the maximum norm.

In section 2 we introduce two polynomial spaces, the corresponding basis functions, and Galerkin matrices. We also discuss the solution of the linear system corresponding to a one-dimensional biharmonic problem. The spectral Galerkin biharmonic problem and its solution are discussed in sections 3 and 4, respectively. Numerical results are presented in section 5 and concluding remarks are given in section 6.

2. Preliminaries. For an integer $N \geq 2$, let

$$P_N^0 = \{p \in P_N : p(\pm 1) = 0\},$$

where P_k denotes the set of polynomials of degree $\leq k$ on $(-1, 1)$. (Note that the dimensions of P_N and P_N^0 are $N + 1$ and $N - 1$, respectively.) Following (2.7) of [7], we introduce the basis $\{\phi_k\}_{k=2}^N$ for P_N^0 with

$$\phi_k(x) = \frac{1}{\sqrt{4k-2}}[L_{k-2}(x) - L_k(x)], \qquad k = 2, \ldots, N,$$

where $L_k(x)$ is the kth degree Legendre polynomial normalized by $\int_{-1}^1 L_k^2(x)\,dx = 2/(2k+1)$. Augmenting the basis $\{\phi_k\}_{k=2}^N$ for P_N^0 by

$$\phi_0(x) = \frac{\sqrt{6}}{2}L_0(x), \qquad \phi_1(x) = \frac{3\sqrt{10}}{2}L_1(x), \tag{2.1}$$

we obtain the basis $\{\phi_k\}_{k=0}^N$ for P_N.

With $(p, q) = \int_{-1}^1 (pq)(x)\,dx$, we introduce the Galerkin matrices

$$A = [(\phi_i', \phi_k')]_{i,k=2}^N, \qquad B = [(\phi_i, \phi_k)]_{i,k=2}^N, \tag{2.2}$$

$$A_t = [(\phi_i', \phi_k')]_{i=2,k=0}^{N,1}, \qquad B_t = [(\phi_i, \phi_k)]_{i=2,k=0}^{N,1}, \tag{2.3}$$

and

$$B_s = [(\phi_i, \phi_k)]_{i,k=0}^1, \tag{2.4}$$

where i and k are the row and column indices, respectively. It follows from Lemma 2.1 in [7] that

$$A = I_{N-1}, \qquad B = \begin{bmatrix} \times & & \times & & & & \\ & \times & & \times & & & \\ \times & & \times & & \times & & \\ & \ddots & & \ddots & & \ddots & \\ & & \times & & \times & & \times \\ & & & \times & & \times & \\ & & & & \times & & \times \end{bmatrix}, \tag{2.5}$$

where I_k is the $k \times k$ identity matrix and the symbols $\times$ denote the only nonzero matrix coefficients. The matrix B is clearly symmetric and positive definite.

Since $\phi_k(\pm 1) = 0$, $k = 2, \ldots, N$, and $\phi_0'' = \phi_1'' = 0$, using integration by parts and the orthonormality of the Legendre polynomials we obtain

$$A_t = O, \qquad B_t = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ \vdots & \vdots \\ 0 & 0 \end{bmatrix}, \qquad B_s = \begin{bmatrix} 3 & 0 \\ 0 & 15 \end{bmatrix}. \tag{2.6}$$

For $\lambda > 0$ consider the linear system

$$\begin{aligned} R_{11}\,[\vec{u}, \vec{v}]^T + R_{12}\,[v_0, v_1]^T &= \left[\vec{g}, \vec{f}\right]^T, \\ R_{21}\,[\vec{u}, \vec{v}]^T + R_{22}[v_0, v_1]^T &= [g_0, g_1]^T, \end{aligned} \tag{2.7}$$

where

$$\vec{u} = [u_2, \ldots, u_N]^T, \qquad \vec{v} = [v_2, \ldots, v_N]^T,$$

$$\vec{g} = [g_2, \ldots, g_N]^T, \qquad \vec{f} = [f_2, \ldots, f_N]^T,$$

$$R_{11} = \begin{bmatrix} B + \lambda I_{N-1} & \lambda B \\ O & B + \lambda I_{N-1} \end{bmatrix}, \tag{2.8}$$

and

$$R_{12} = \begin{bmatrix} \lambda B_t \\ B_t \end{bmatrix}, \qquad R_{21} = \begin{bmatrix} B_t^T, \lambda B_t^T \end{bmatrix}, \qquad R_{22} = \lambda B_s. \tag{2.9}$$

Assume that N is even and introduce the vectors

$$\vec{u}^{(0)} = [u_2, u_4, \ldots, u_N]^T, \qquad \vec{u}^{(1)} = [u_3, u_5, \ldots, u_{N-1}]^T.$$

For $i = 0, 1$, let the vectors $\vec{v}^{(i)}$, $\vec{g}^{(i)}$, and $\vec{f}^{(i)}$ be defined in a similar way. Since B in (2.5) consists of two tridiagonal matrices B_0 and B_1 and since B_s in (2.6) is diagonal, (2.7) splits into two systems

$$\begin{aligned} A_i \left[\vec{u}^{(i)}, \vec{v}^{(i)}\right]^T + \vec{p} v_i &= \left[\vec{g}^{(i)}, \vec{f}^{(i)}\right]^T, \\ \vec{q} \left[\vec{u}^{(i)}, \vec{v}^{(i)}\right]^T + d_i v_i &= g_i \end{aligned} \tag{2.10}$$

for $i = 0, 1$, where

$$A_i = \begin{bmatrix} B_i + \lambda I_{N/2-i} & \lambda B_i \\ O & B_i + \lambda I_{N/2-i} \end{bmatrix} \tag{2.11}$$

and

$$\vec{p} = [\lambda, 0, \ldots, 0, 1, 0, \ldots, 0]^T, \quad \vec{q} = [1, 0, \ldots, 0, \lambda, 0, \ldots, 0], \quad d_0 = 3\lambda, \quad d_1 = 15\lambda. \tag{2.12}$$

It can be shown that the matrix in (2.10) is nonsingular. Moreover, A_i is nonsingular since $B_i + \lambda I_{N/2-i}$ is positive definite. Solving the first equation of (2.10) for $[\vec{u}^{(i)}, \vec{v}^{(i)}]$ and substituting it into the second equation of (2.10), we obtain

$$v_i = (g_i - \vec{s}^{(i)}[\vec{g}^{(i)}, \vec{f}^{(i)}]^T)/\delta_i, \tag{2.13}$$

where

$$\vec{s}^{(i)} = \vec{q} A_i^{-1}, \quad \delta_i = d_i - \vec{s}^{(i)}\vec{p}. \tag{2.14}$$

The vector $\vec{s}^{(i)}$ and the number δ_i can be computed with costs $O(N)$ and $O(1)$, respectively. Once the number v_i of (2.13) has been evaluated at cost $O(N)$, $\vec{u}^{(i)}$ and $\vec{v}^{(i)}$ of (2.10) can be obtained with cost $O(N)$ by solving two systems with the same tridiagonal positive definite matrix $B_i + \lambda I_{N/2-i}$. Therefore, the cost of solving (2.7) is $O(N)$.

Throughout the paper we use the symbol $\otimes$ to denote the tensor product of matrices and the tensor product of function spaces. Let I, J, K, and L be finite sets of increasing indices. Without loss of generality we assume

$$I = \{1, \ldots, M_1\}, \quad K = \{1, \ldots, N_1\}, \quad J = \{1, \ldots, M_2\}, \quad L = \{1, \ldots, N_2\}.$$

Then the matrix-vector form of

$$z_{i,j} = \sum_{k \in K} c^{(1)}_{i,k} \sum_{l \in L} c^{(2)}_{j,l} w_{k,l}, \quad i \in I, \quad j \in J, \tag{2.15}$$

is

$$\vec{z} = (C_1 \otimes C_2)\vec{w}, \tag{2.16}$$

where

$$C_1 = (c^{(1)}_{i,k})_{i \in I, k \in K}, \quad C_2 = (c^{(2)}_{j,l})_{j \in J, l \in L},$$

and

$$\vec{z} = [z_{1,1}, \ldots, z_{1,M_2}, \ldots, z_{M_1,1}, \ldots, z_{M_1,M_2}]^T,$$
$$\vec{w} = [w_{1,1}, \ldots, w_{1,N_2}, \ldots, w_{N_1,1}, \ldots, w_{N_1,N_2}]^T.$$

3. Biharmonic spectral Galerkin problem. Introducing $v = \Delta u$ in (1.1), we obtain the coupled problem (see Figure 1)

$$\begin{aligned} -\Delta u + v = 0 \text{ in } \Omega, \quad u = 0 \text{ on } \partial\Omega, \quad \partial u/\partial n = 0 \text{ on } \partial\Omega, \\ -\Delta v = -f \text{ in } \Omega. \end{aligned} \tag{3.1}$$

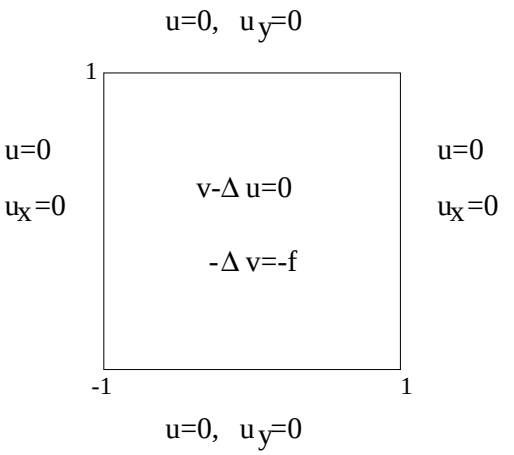


FIG. 1. *Coupled problem* (3.1).

The weak form of (3.1), obtained using Green's formula, is

$$\int_\Omega \nabla u \nabla \eta \, d\Omega + \int_\Omega v\eta \, d\Omega = 0, \qquad \eta \in H^1(\Omega),$$
$$\int_\Omega \nabla v \nabla \delta \, d\Omega = -\int_\Omega f\delta \, d\Omega, \qquad \delta \in H_0^1(\Omega).$$

Let

$$X_N = \{w \in P_N \otimes P_N : w(\alpha,\beta) = 0, \alpha,\beta = \pm 1\}, \qquad X_N^0 = P_N^0 \otimes P_N^0.$$

The corner conditions $w(\alpha,\beta) = 0$, $\alpha,\beta = \pm 1$, in the definition of X_N are motivated by the fact that

$$v(\alpha,\beta) = u_{xx}(\alpha,\beta) + u_{yy}(\alpha,\beta) = 0, \qquad \alpha,\beta = \pm 1.$$

The Legendre spectral Galerkin problem for (3.1) consists of finding $U \in X_N^0$ and $V \in X_N$ such that

$$\int_\Omega \nabla U \nabla \eta \, d\Omega + \int_\Omega V\eta \, d\Omega = 0, \qquad \eta \in X_N, \tag{3.2}$$

$$\int_\Omega \nabla V \nabla \delta \, d\Omega = -\int_\Omega f\delta \, d\Omega, \qquad \delta \in X_N^0. \tag{3.3}$$

THEOREM 3.1. *There exist unique $U \in X_N^0$ and $V \in X_N$ satisfying* (3.2)–(3.3).

Proof. Since in (3.2)–(3.3) the number of constraints is equal to the number of degrees of freedom, we assume that $f = 0$ and show that $U = V = 0$. Taking $\eta = V$ in (3.2) and $\delta = U$ in (3.3), we obtain

$$\int_\Omega \nabla U \nabla V \, d\Omega + \int_\Omega V^2 \, d\Omega = 0, \qquad \int_\Omega \nabla V \nabla U \, d\Omega = 0,$$

which gives $\int_\Omega V^2 \, d\Omega = 0$, $V = 0$. With $V = 0$ and $\eta = U$, (3.2) becomes $\int_\Omega \nabla U \nabla U \, d\Omega = 0$, which implies $U = 0$ by the Poincaré inequality. ∎

Using the basis functions $\{\phi_k\}_{k=0}^N$ of section 2 for $U \in X_N^0$ and $V \in X_N$, we have

$$U(x,y) = \sum_{k=2}^{N}\sum_{l=2}^{N} u_{k,l}\phi_k(x)\phi_l(y) \tag{3.4}$$

and

$$V(x,y) = \sum_{k=2}^{N}\sum_{l=2}^{N} v_{k,l}\phi_k(x)\phi_l(y) + \sum_{k=2}^{N}\sum_{l=0}^{1} v_{k,l}\phi_k(x)\phi_l(y) + \sum_{k=0}^{1}\sum_{l=2}^{N} v_{k,l}\phi_k(x)\phi_l(y), \tag{3.5}$$

where we used properties of ϕ_0 and ϕ_1 of (2.1) in the derivation of (3.5). Corresponding to (3.4) and (3.5) we introduce the vectors

$$\vec{u} = [u_{2,2}, \ldots, u_{2,N}, \ldots, u_{N,2}, \ldots, u_{N,N}]^T, \tag{3.6}$$

$$\vec{v} = [v_{2,2}, \ldots, v_{2,N}, \ldots, v_{N,2}, \ldots, v_{N,N}]^T, \tag{3.7}$$

$$\vec{v}_h = [v_{2,0}, v_{2,1}, \ldots, v_{N,0}, v_{N,1}]^T, \tag{3.8}$$

$$\vec{v}_v = [v_{0,2}, \ldots, v_{0,N}, v_{1,2}, \ldots, v_{1,N}]^T. \tag{3.9}$$

Substituting (3.4), (3.5) into (3.2)–(3.3), taking $\eta = \phi_i(x)\phi_j(y)$, $i,j = 2,\ldots,N$, $\delta = \phi_i(x)\phi_j(y)$, $i,j = 2,\ldots,N$, $\eta = \phi_i(x)\phi_j(y)$, $i = 2,\ldots,N$, $j = 0,1$, $\eta = \phi_i(x)\phi_j(y)$, $i = 0,1$, $j = 2,\ldots,N$, and using (2.15), (2.16), (2.2)–(2.6), we obtain, respectively,

$$\begin{gathered}
(I_{N-1}\otimes B + B\otimes I_{N-1})\vec{u} + (B\otimes B)\vec{v} + (B\otimes B_t)\vec{v}_h + (B_t\otimes B)\vec{v}_v = \vec{0},\\
(I_{N-1}\otimes B + B\otimes I_{N-1})\vec{v} + (I_{N-1}\otimes B_t)\vec{v}_h + (B_t\otimes I_{N-1})\vec{v}_v = \vec{f},\\
(I_{N-1}\otimes B_t^T)\vec{u} + (B\otimes B_t^T)\vec{v} + (B\otimes B_s)\vec{v}_h + (B_t\otimes B_t^T)\vec{v}_v = \vec{0},\\
(B_t^T\otimes I_{N-1})\vec{u} + (B_t^T\otimes B)\vec{v} + (B_t^T\otimes B_t)\vec{v}_h + (B_s\otimes B)\vec{v}_v = \vec{0},
\end{gathered} \tag{3.10}$$

where

$$\vec{f} = [f_{2,2}, \ldots, f_{2,N}, \ldots, f_{N,2}, \ldots, f_{N,N}]^T \tag{3.11}$$

and

$$f_{i,j} = -\int_\Omega f\phi_i(x)\phi_j(y)\,d\Omega.$$

Using the N-point Gauss–Legendre quadrature in the x and y coordinates, with nodes $\{\xi_k\}_{k=1}^N$ and weights $\{w_k\}_{k=1}^N$, we approximate $\vec{f}$ by

$$\vec{f} \approx -(CD\otimes CD)\left[f(\xi_1,\xi_1), \ldots, f(\xi_1,\xi_N), \ldots, f(\xi_N,\xi_1), \ldots, f(\xi_N,\xi_N)\right]^T, \tag{3.12}$$

where C and D are given by

$$C = [\phi_i(\xi_k)]_{i=2,k=1}^{N,N}, \qquad D = \text{ diag } (w_1, \ldots, w_N).$$

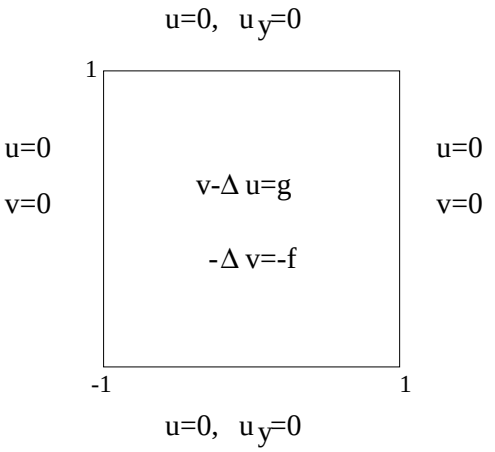


FIG. 2. *Auxiliary problem* (4.1).

The computation of the right-hand side of (3.12) requires $O(N^3)$ operations. The vector $\vec{f}$ can be also approximated in a slightly different and more efficient way using the approach described in [6].

Equations (3.10) can be written as

$$\begin{aligned} S_{11}\left[\vec{u},\vec{v},\vec{v}_h\right]^T + S_{12}\vec{v}_v &= \left[\vec{0},\vec{f},\vec{0}\right]^T, \\ S_{21}\left[\vec{u},\vec{v},\vec{v}_h\right]^T + S_{22}\vec{v}_v &= \vec{0}, \end{aligned} \tag{3.13}$$

where

$$S_{11} = \begin{bmatrix} I_{N-1}\otimes B + B\otimes I_{N-1} & B\otimes B & B\otimes B_t \\ O & I_{N-1}\otimes B + B\otimes I_{N-1} & I_{N-1}\otimes B_t \\ I_{N-1}\otimes B_t^T & B\otimes B_t^T & B\otimes B_s \end{bmatrix}, \tag{3.14}$$

$$S_{12} = \begin{bmatrix} B_t\otimes B \\ B_t\otimes I_{N-1} \\ B_t\otimes B_t^T \end{bmatrix}, \tag{3.15}$$

$$S_{21} = \begin{bmatrix} B_t^T\otimes I_{N-1}, B_t^T\otimes B, B_t^T\otimes B_t \end{bmatrix}, \tag{3.16}$$

$$S_{22} = B_s\otimes B. \tag{3.17}$$

In the remainder of the paper we describe an algorithm for solving (3.13) assuming that $\vec{f}$ or its approximation is given.

4. Solving the biharmonic linear system.

4.1. Inverse of S_{11}. The matrix S_{11} of (3.14) arises in the spectral Galerkin method for the auxiliary problem (see Figure 2)

$$\begin{aligned} -\Delta u + v = g \text{ in } \Omega, \quad u = 0 \text{ on } \partial\Omega, \quad \partial u/\partial n = 0 \text{ on } \partial\Omega_h, \\ -\Delta v = -f \text{ in } \Omega, \quad v = 0 \text{ on } \partial\Omega_v, \end{aligned} \tag{4.1}$$

where $\partial\Omega_h$ is the union of the horizontal sides of $\partial\Omega$ and $\partial\Omega_v$ is the union of the vertical sides of $\partial\Omega$. The weak form of (4.1) is

$$\int_\Omega \nabla u \nabla \eta \, d\Omega + \int_\Omega v\eta \, d\Omega = \int_\Omega g\eta \, d\Omega, \qquad \eta \in H^1(\Omega), \qquad \eta = 0 \text{ on } \partial\Omega_v,$$
$$\int_\Omega \nabla v \nabla \delta \, d\Omega = -\int_\Omega f\delta \, d\Omega, \qquad \delta \in H_0^1(\Omega).$$

The spectral Galerkin problem for (4.1) consists of finding $U \in X_N^0$ and $V \in P_N^0 \otimes P_N$ such that

$$\int_\Omega \nabla U \nabla \eta \, d\Omega + \int_\Omega V\eta \, d\Omega = \int_\Omega g\eta \, d\Omega, \qquad \eta \in P_N^0 \otimes P_N,$$
$$\int_\Omega \nabla V \nabla \delta \, d\Omega = -\int_\Omega f\delta \, d\Omega, \qquad \delta \in X_N^0. \tag{4.2}$$

LEMMA 4.1. *The matrix S_{11} of (3.14) is nonsingular.*

Proof. It can be shown, using an approach similar to the proof of Theorem 3.1, that (4.2) has a unique solution. ∎

Since B of (2.2) is symmetric, it is orthogonally similar to a diagonal matrix, that is, there exist real $Z = (z_{i,j})_{i,j=2}^N$ and real

$$\Lambda = \operatorname{diag}(\lambda_2, \ldots, \lambda_N) \tag{4.3}$$

such that

$$Z^T Z = I_{N-1}, \qquad Z^T B Z = \Lambda. \tag{4.4}$$

It follows from the structure of B in (2.5) that the computation of Λ and Z satisfying (4.3) and (4.4) reduces to solving two symmetric eigenvalue problems with tridiagonal matrices. With the use of the QR algorithm for evaluating eigenvalues and the inverse iteration for evaluating the corresponding eigenvectors, Λ and Z can be precomputed with cost $O(N^2)$. Moreover, the coefficients of Z are such that

$$z_{i,j} = 0 \quad \text{if} \quad i + j \text{ is odd}. \tag{4.5}$$

Let

$$W = \begin{bmatrix} Z \otimes I_{N-1} & O & O \\ O & Z \otimes I_{N-1} & O \\ O & O & Z \otimes I_2 \end{bmatrix}. \tag{4.6}$$

Then using (3.14) and (4.4), we obtain

$$W^T S_{11} W = G,$$

where

$$G = \begin{bmatrix} I_{N-1} \otimes B + \Lambda \otimes I_{N-1} & \Lambda \otimes B & \Lambda \otimes B_t \\ O & I_{N-1} \otimes B + \Lambda \otimes I_{N-1} & I_{N-1} \otimes B_t \\ I_{N-1} \otimes B_t^T & \Lambda \otimes B_t^T & \Lambda \otimes B_s \end{bmatrix}. \tag{4.7}$$

Since W is orthogonal, it follows that

$$S_{11}^{-1} = W G^{-1} W^T. \tag{4.8}$$

Let the vectors $\vec{u}$, $\vec{v}$, $\vec{v}_h$, and $\vec{f}$ have the forms given in (3.6), (3.7), (3.8), and (3.11), respectively, and let

$$\vec{g} = [g_{2,2}, \ldots, g_{2,N}, \ldots, g_{N,2}, \ldots, g_{N,N}]^T, \qquad \vec{g}_h = [g_{2,0}, g_{2,1}, \ldots, g_{N,0}, g_{N,1}]^T. \tag{4.9}$$

In the following, the vectors $\vec{u}'$, $\vec{v}'$, $\vec{v}'_h$, $\vec{f}'$, $\vec{g}'$, and $\vec{g}'_h$ have the same forms as $\vec{u}$, $\vec{v}$, $\vec{v}_h$, $\vec{f}$, $\vec{g}$, and $\vec{g}_h$, respectively. The components of the primed vectors are denoted by the primed letters corresponding to the unprimed vectors. For example,

$$\vec{u}' = [u'_{2,2}, \ldots, u'_{2,N}, \ldots, u'_{N,2}, \ldots, u'_{N,N}]^T, \qquad \vec{v}'_h = [v'_{2,0}, v'_{2,1}, \ldots, v'_{N,0}, v'_{N,1}]^T.$$

Consider the computation of

$$[\vec{u}', \vec{v}', \vec{v}'_h]^T = G^{-1}[\vec{g}', \vec{f}', \vec{g}'_h]^T. \tag{4.10}$$

It follows from (4.7) that (4.10) is equivalent to the system

$$\begin{aligned}
(I_{N-1} \otimes B + \Lambda \otimes I_{N-1})\vec{u}' + (\Lambda \otimes B)\vec{v}' + (\Lambda \otimes B_t)\vec{v}'_h &= \vec{g}', \\
(I_{N-1} \otimes B + \Lambda \otimes I_{N-1})\vec{v}' + (I_{N-1} \otimes B_t)\vec{v}'_h &= \vec{f}', \\
(I_{N-1} \otimes B_t^T)\vec{u}' + (\Lambda \otimes B_t^T)\vec{v}' + (\Lambda \otimes B_s)\vec{v}'_h &= \vec{g}'_h,
\end{aligned}$$

which, from (4.3), becomes

$$\begin{aligned}
R_{11}^{(k)} \left[\vec{u}'_{k,\cdot}, \vec{v}'_{k,\cdot}\right]^T + R_{12}^{(k)} \left[v'_{k,0}, v'_{k,1}\right]^T &= \left[\vec{g}'_{k,\cdot}, \vec{f}'_{k,\cdot}\right]^T, \\
R_{21}^{(k)} \left[\vec{u}'_{k,\cdot}, \vec{v}'_{k,\cdot}\right]^T + R_{22}^{(k)} [v'_{k,0}, v'_{k,1}]^T &= [g'_{k,0}, g'_{k,1}]^T
\end{aligned} \tag{4.11}$$

for $k = 2, \ldots, N$, where the $R_{ij}^{(k)}$ are given by (2.8), (2.9) with λ replaced by λ_k, and

$$\vec{u}'_{k,\cdot} = [u'_{k,2}, \ldots, u'_{k,N}]^T, \qquad \vec{v}'_{k,\cdot} = [v'_{k,2}, \ldots, v'_{k,N}]^T,$$

$$\vec{g}'_{k,\cdot} = [g'_{k,2}, \ldots, g'_{k,N}]^T, \qquad \vec{f}'_{k,\cdot} = [f'_{k,2}, \ldots, f'_{k,N}]^T.$$

Since B is positive definite, it follows from the second equation in (4.4) that Λ is positive definite and hence, by (4.3), $\lambda_k > 0$, $k = 2, \ldots, N$. Clearly (4.11) is of the same form as (2.7) with λ replaced by λ_k. Hence it follows from the discussion in section 2 that $[\vec{u}', \vec{v}', \vec{v}'_h]^T$ of (4.10) can be computed with cost $O(N^2)$. Also it follows from (2.13) that in the special case of $\vec{g}' = \vec{f}' = \vec{0}$, $\vec{v}'_h$ of (4.10) can be obtained with cost $O(N)$.

4.2. Description of the algorithm. Since S_{11} is nonsingular (see Lemma 4.1), eliminating $[\vec{u}, \vec{v}, \vec{v}_h]^T$ from (3.13), and using (4.8), (4.6), we obtain

$$S\vec{v}_v = -S_{21}WG^{-1} \left[\vec{0}, (Z^T \otimes I_{N-1})\vec{f}, \vec{0}\right]^T, \tag{4.12}$$

where S is the $2(N-1) \times 2(N-1)$ Schur complement matrix given by

$$S = S_{22} - S_{21}S_{11}^{-1}S_{12}. \tag{4.13}$$

(S is the Schur complement of S_{11} in $\left[\begin{smallmatrix} S_{11} & S_{12} \\ S_{21} & S_{22} \end{smallmatrix}\right]$.) It also follows from (3.13), (4.8), and (4.6) that

$$[\vec{u}, \vec{v}, \vec{v}_h]^T = WG^{-1}\{[\vec{0}, (Z^T \otimes \vec{f}, \vec{0})] - W^T S_{12}\vec{v}_v\}. \tag{4.14}$$

Using (4.12), (4.14), and (4.6) we arrive at the following algorithm for solving (3.13).

ALGORITHM.

Step 1. Compute $\vec{f}' = (Z^T \otimes I_{N-1})\vec{f}$.

Step 2. Compute $[\vec{u}', \vec{v}', \vec{v}_h']^T = G^{-1}[\vec{0}, \vec{f}', \vec{0}]^T$.

Step 3. Compute $\vec{r} = -S_{21}[\vec{u}, \vec{v}, \vec{v}_h]^T$, where $\vec{u} = (Z \otimes I_{N-1})\vec{u}'$, $\vec{v} = (Z \otimes I_{N-1})\vec{v}'$, $\vec{v}_h = (Z \otimes I_2)\vec{v}_h'$.

Step 4. Solve $S\vec{v}_v = \vec{r}$ for $\vec{v}_v$.

Step 5. Compute $\vec{g}' = (Z^T \otimes I_{N-1})\vec{g}$, $\vec{f}'' = (Z^T \otimes I_{N-1})\vec{f}$, $\vec{g}_h' = (Z^T \otimes I_2)\vec{g}_h$, where $[\vec{g}, \vec{f}, \vec{g}_h]^T = S_{12}\vec{v}_v$.

Step 6. Compute $[\vec{g}'', \vec{f}''', \vec{g}_h'']^T = [\vec{0}, \vec{f}', \vec{0}]^T - [\vec{g}', \vec{f}'', \vec{g}_h']^T$.

Step 7. Compute $[\vec{u}', \vec{v}', \vec{v}_h']^T = G^{-1}[\vec{g}'', \vec{f}''', \vec{g}_h'']^T$.

Step 8. Compute $\vec{u} = (Z \otimes I_{N-1})\vec{u}'$, $\vec{v} = (Z \otimes I_{N-1})\vec{v}'$,$\vec{v}_h = (Z \otimes I_2)\vec{v}_h'$.

It follows from (4.5) that the cost of Step 1 is $N^3 + O(N^2)$ and the cost of Step 8 is $2N^3 + O(N^2)$. However, if the approximation to $v = \Delta u$ is not required, then the cost of Step 8 is only $N^3 + O(N^2)$. It follows from the discussion in section 4.1 that the costs of Steps 2 and 7 are $O(N^2)$ each. Clearly, the cost of Step 6 is also $O(N^2)$.

Let $\vec{u}$, $\vec{v}$, and $\vec{v}_h$ be of the forms given in (3.6), (3.7), and (3.8), respectively, and let, for $k = 2, \ldots, N$,

$$\vec{u}_{k,\cdot} = [u_{k,2}, \ldots, u_{k,N}]^T, \qquad \vec{v}_{k,\cdot} = [v_{k,2}, \ldots, v_{k,N}]^T.$$

Then it follows from (3.16) and (2.6) that

$$S_{21}[\vec{u}, \vec{v}, \vec{v}_h]^T = [\vec{u}_{2,\cdot}, \vec{u}_{3,\cdot}]^T + [B\vec{v}_{2,\cdot}, B\vec{v}_{3,\cdot}]^T + [v_{2,0}, v_{2,1}, 0, \ldots, 0, v_{3,0}, v_{3,1}, 0, \ldots, 0]^T. \tag{4.15}$$

Hence the cost of Step 3 is $O(N^2)$.

Let $\vec{v}_v$, $\vec{f}$, and $\vec{g}$, $\vec{g}_h$ be of the forms given in (3.9), (3.11), and (4.9), respectively. Let

$$\vec{v}_v^{(0)} = [v_{0,2}, \ldots, v_{0,N}]^T, \qquad \vec{v}_v^{(1)} = [v_{1,2}, \ldots, v_{1,N}]^T, \tag{4.16}$$

and let, for $k = 2, \ldots, N$,

$$\vec{f}_{k,\cdot} = [f_{k,2}, \ldots, f_{k,N}]^T, \qquad \vec{g}_{k,\cdot} = [g_{k,2}, \ldots, g_{k,N}]^T.$$

Then it follows from (3.15) and (2.6) that for

$$[\vec{g}, \vec{f}, \vec{g}_h] = S_{12}\vec{v}_v, \tag{4.17}$$

we have

$$\vec{g}_{2,\cdot} = B\vec{v}_v^{(0)}, \quad \vec{g}_{3,\cdot} = B\vec{v}_v^{(1)}, \qquad \vec{f}_{2,\cdot} = \vec{v}_v^{(0)}, \quad \vec{f}_{3,\cdot} = \vec{v}_v^{(1)}, \qquad \vec{f}_{k,\cdot} = \vec{g}_{k,\cdot} = \vec{0}, \quad k = 4, \ldots, N, \tag{4.18}$$

$$g_{2,0} = v_{0,2}, \quad g_{2,1} = v_{0,3} \qquad g_{3,0} = v_{1,2}, \quad g_{3,1} = v_{1,3}, \qquad g_{k,0} = g_{k,1} = 0, \quad k = 4, \ldots, N. \tag{4.19}$$

Hence the cost of Step 5 is $O(N^2)$.

In the next subsection we discuss in more detail Step 4.

4.3. Solving systems with S.

THEOREM 4.1. *The matrix S of* (4.13) *is symmetric and positive definite.*

Proof. For $n = 1, 2$, and arbitrary

$$\vec{v}_v^{(n)} = [v_{0,2}^{(n)}, \ldots, v_{0,N}^{(n)}, v_{1,2}^{(n)}, \ldots, v_{1,N}^{(n)}]^T, \tag{4.20}$$

using (4.13), we obtain

$$(S\vec{v}_v^{(1)}, \vec{v}_v^{(2)})_{R^{2N-2}} = (S_{22}\vec{v}_v^{(1)}, \vec{v}_v^{(2)})_{R^{2N-2}} - (S_{21}S_{11}^{-1}S_{12}\vec{v}_v^{(1)}, \vec{v}_v^{(2)})_{R^{2N-2}}. \tag{4.21}$$

Let

$$\vec{u}^{(n)} = [u_{2,2}^{(n)}, \ldots, u_{2,N}^{(n)}, \ldots, u_{N,2}^{(n)}, \ldots, u_{N,N}^{(n)}]^T, \tag{4.22}$$

$$\vec{v}^{(n)} = [v_{2,2}^{(n)}, \ldots, v_{2,N}^{(n)}, \ldots, v_{N,2}^{(n)}, \ldots, v_{N,N}^{(n)}]^T, \tag{4.23}$$

$$\vec{v}_h^{(n)} = [v_{2,0}^{(n)}, v_{2,1}^{(n)}, \ldots, v_{N,0}^{(n)}, v_{N,1}^{(n)}]^T$$

be such that

$$S_{11}[\vec{u}^{(n)}, \vec{v}^{(n)}, \vec{v}_h^{(n)}]^T + S_{12}\vec{v}_v^{(n)} = \vec{0}. \tag{4.24}$$

Existence and uniqueness of $\vec{u}^{(n)}$, $\vec{v}^{(n)}$, and $\vec{v}_h^{(n)}$ follow from the nonsingularity of S_{11} (see Lemma 4.1). Then (4.21), (4.24), (3.17), and (3.16) give

$$\begin{aligned}(S\vec{v}_v^{(1)}, \vec{v}_v^{(2)})_{R^{2N-2}} &= (S_{22}\vec{v}_v^{(1)}, \vec{v}_v^{(2)})_{R^{2N-2}} + (S_{21}[\vec{u}^{(1)}, \vec{v}^{(1)}, \vec{v}_h^{(1)}]^T, \vec{v}_v^{(2)})_{R^{2N-2}} \\ &= ((B_s \otimes B)\vec{v}_v^{(1)}, \vec{v}_v^{(2)})_{R^{2N-2}} + ((B_t^T \otimes I_{N-1})\vec{u}^{(1)}, \vec{v}_v^{(2)})_{R^{2N-2}} \\ &\quad + ((B_t^T \otimes B)\vec{v}^{(1)}, \vec{v}_v^{(2)})_{R^{2N-2}} + ((B_t^T \otimes B_t)\vec{v}_h^{(1)}, \vec{v}_v^{(2)})_{R^{2N-2}}.\end{aligned} \tag{4.25}$$

Equation (4.24) is the matrix-vector form of the spectral Galerkin problem

$$\begin{aligned}\int_\Omega \nabla U^{(n)} \nabla \eta \, d\Omega + \int_\Omega V^{(n)} \eta \, d\Omega = 0, \quad \eta \in P_N^0 \otimes P_N, \\ \int_\Omega \nabla V^{(n)} \nabla \delta \, d\Omega = 0, \quad \delta \in X_N^0,\end{aligned} \tag{4.26}$$

where

$$U^{(n)}(x, y) = \sum_{k=2}^{N} \sum_{l=2}^{N} u_{k,l}^{(n)} \phi_k(x) \phi_l(y), \tag{4.27}$$

$$V^{(n)}(x, y) = V_i^{(n)}(x, y) + V_h^{(n)}(x, y) + V_v^{(n)}(x, y), \tag{4.28}$$

and

$$V_i^{(n)}(x, y) = \sum_{k=2}^{N} \sum_{l=2}^{N} v_{k,l}^{(n)} \phi_k(x) \phi_l(y), \tag{4.29}$$

$$V_h^{(n)}(x, y) = \sum_{k=2}^{N} \sum_{l=0}^{1} v_{k,l}^{(n)} \phi_k(x) \phi_l(y), \tag{4.30}$$

$$V_v^{(n)}(x,y) = \sum_{k=0}^{1}\sum_{l=2}^{N} v_{k,l}^{(n)}\phi_k(x)\phi_l(y). \tag{4.31}$$

Since $V_i^{(2)} + V_h^{(2)} \in P_N^0 \otimes P_N$ and $U^{(1)} \in X_N^0$, it follows from (4.28) and (4.26) that

$$\begin{aligned}\int_\Omega V^{(1)}V^{(2)}\,d\Omega &= \int_\Omega V^{(1)}V_v^{(2)}\,d\Omega + \int_\Omega V^{(1)}\left(V_i^{(2)} + V_h^{(2)}\right)d\Omega \\ &= \int_\Omega V^{(1)}V_v^{(2)}\,d\Omega - \int_\Omega \nabla U^{(1)}\nabla\left(V^{(2)} - V_v^{(2)}\right)d\Omega \\ &= \int_\Omega V_i^{(1)}V_v^{(2)}\,d\Omega + \int_\Omega V_h^{(1)}V_v^{(2)}\,d\Omega + \int_\Omega V_v^{(1)}V_v^{(2)}\,d\Omega + \int_\Omega \nabla U^{(1)}\nabla V_v^{(2)}\,d\Omega.\end{aligned} \tag{4.32}$$

Using (4.29), (4.31), (2.15), (2.16), (2.3), and (2.2) we have

$$\begin{aligned}\int_\Omega V_i^{(1)}V_v^{(2)}\,d\Omega &= \int_\Omega \left[\sum_{k=2}^{N}\sum_{l=2}^{N} v_{k,l}^{(1)}\phi_k(x)\phi_l(y)\right]\left[\sum_{i=0}^{1}\sum_{j=2}^{N} v_{i,j}^{(2)}\phi_i(x)\phi_j(y)\right]d\Omega \\ &= \sum_{i=0}^{1}\sum_{j=2}^{N}\left[\sum_{k=2}^{N}\int_{-1}^{1}\phi_i(x)\phi_k(x)\,dx\sum_{l=2}^{N}\int_{-1}^{1}\phi_j(y)\phi_l(y)\,dy v_{k,l}^{(1)}\right]v_{i,j}^{(2)} \\ &= ((B_t^T \otimes B)\vec{v}^{(1)}, \vec{v}_v^{(2)})_{R^{2N-2}}.\end{aligned} \tag{4.33}$$

In a similar way, using (4.30), (4.31), (4.27), (2.15), (2.16), and (2.2)–(2.4), we obtain

$$\int_\Omega V_h^{(1)}V_v^{(2)}\,d\Omega = ((B_t^T \otimes B_t)\vec{v}_h^{(1)}, \vec{v}_v^{(2)})_{R^{2N-2}}, \tag{4.34}$$

$$\int_\Omega V_v^{(1)}V_v^{(2)}\,d\Omega = ((B_s \otimes B)\vec{v}_v^{(1)}, \vec{v}_v^{(2)})_{R^{2N-2}}, \tag{4.35}$$

$$\int_\Omega \nabla U^{(1)}V_v^{(2)}\,d\Omega = ((B_t^T \otimes I_{N-1})\vec{u}^{(1)}, \vec{v}_v^{(2)})_{R^{2N-2}}. \tag{4.36}$$

Hence, it follows from (4.25) and (4.32)–(4.36) that

$$(S\vec{v}_v^{(1)}, \vec{v}_v^{(2)})_{R^{2N-2}} = \int_\Omega V^{(1)}V^{(2)}\,d\Omega, \tag{4.37}$$

which implies

$$(S\vec{v}_v^{(1)}, \vec{v}_v^{(2)})_{R^{2N-2}} = (S\vec{v}_v^{(2)}, \vec{v}_v^{(1)})_{R^{2N-2}}, \qquad \vec{v}_v^{(1)}, \vec{v}_v^{(2)} \in R^{2N-2}.$$

This proves the symmetry of S.

For $\vec{v}_v^{(1)} = \vec{v}_v^{(2)}$, (4.37) gives

$$(S\vec{v}_v^{(1)}, \vec{v}_v^{(1)})_{R^{2N-2}} = \int_\Omega [V^{(1)}]^2\,d\Omega.$$

This and the linear independence of the basis functions in $V^{(1)}$ of (4.28)–(4.31) show that S is positive definite. □

It follows from Theorem 4.1 that the PCG method is a good candidate for solving a linear system with S. Therefore, in the following, we discuss the matrix-vector multiplication involving S, the selection of a preconditioner, and the solution of a linear system with this preconditioner.

It follows from (4.13) that in order to multiply by S, we have to multiply by S_{22} and $S_{21}S_{11}^{-1}S_{12}$. Assume that N is even. Let $\vec{v}_v$ and $\vec{v}_v^{(0)}$, $\vec{v}_v^{(1)}$ be as in (3.9) and (4.16), respectively. Then it follows from (3.17) and (2.6) that

$$S_{22}\vec{v}_v = [3B\vec{v}_v^{(0)}, 15B\vec{v}_v^{(1)}]^T. \tag{4.38}$$

Using (4.17)–(4.19), (4.8), (4.10), (4.15), (4.6), (4.5), and (4.11) it can be shown that

$$S_{21}S_{11}^{-1}S_{12}\vec{v}_v = [\vec{u}_{2,\cdot}, \vec{u}_{3,\cdot}]^T + [B\vec{v}_{2,\cdot}, B\vec{v}_{3,\cdot}]^T + [v_{2,0}, v_{2,1}, 0, \ldots, 0, v_{3,0}, v_{3,1}, 0, \ldots, 0]^T, \tag{4.39}$$

where

$$[\vec{u}_{2,\cdot}, \vec{v}_{2,\cdot}, v_{2,0}, v_{2,1}]^T = \sum_{k=1}^{N/2} z_{2,2k}^2 [R^{(2k)}]^{-1} [B\vec{v}_v^{(0)}, \vec{v}_v^{(0)}, v_{0,2}, v_{0,3}]^T, \tag{4.40}$$

$$[\vec{u}_{3,\cdot}, \vec{v}_{3,\cdot}, v_{3,0}, v_{3,1}]^T = \sum_{k=1}^{N/2-1} z_{3,2k+1}^2 [R^{(2k+1)}]^{-1} [B\vec{v}_v^{(1)}, \vec{v}_v^{(1)}, v_{1,2}, v_{1,3}]^T, \tag{4.41}$$

$$R^{(k)} = \begin{bmatrix} R_{11}^{(k)} & R_{12}^{(k)} \\ R_{21}^{(k)} & R_{22}^{(k)} \end{bmatrix},$$

and $R_{ij}^{(k)}$ are given by (2.8), (2.9) with λ replaced by λ_k. We next analyze (4.40). The analysis of (4.41) is similar. Introducing

$$[\vec{u}^{(2k)}, \vec{v}^{(2k)}, v_0^{(2k)}, v_1^{(2k)}]^T = [R^{(2k)}]^{-1} [B\vec{v}_v^{(0)}, \vec{v}_v^{(0)}, v_{0,2}, v_{0,3}]^T$$

and using (4.40), we obtain

$$[\vec{u}_{2,\cdot}, \vec{v}_{2,\cdot}]^T = \sum_{k=1}^{N/2} z_{2,2k}^2 [R_{11}^{(2k)}]^{-1} \left\{ [B\vec{v}_v^{(0)}, \vec{v}_v^{(0)}]^T - R_{12}^{(2k)} [v_0^{(2k)}, v_1^{(2k)}]^T \right\}, \tag{4.42}$$

$$v_{2,j} = \sum_{k=1}^{N/2} z_{2,2k}^2 v_j^{(2k)}, \qquad j = 0, 1.$$

Further, (2.8) and (4.4) imply that

$$[R_{11}^{(2k)}]^{-1} = (I_2 \otimes Z) D_{2k}^{-1} (I_2 \otimes Z^T),$$

where

$$D_{2k} = \begin{bmatrix} \Lambda + \lambda_{2k} I_{N-1} & \lambda_{2k}\Lambda \\ O & \Lambda + \lambda_{2k} I_{N-1} \end{bmatrix}.$$

Hence (4.42) becomes

$$[\vec{u}_{2,\cdot}, \vec{v}_{2,\cdot}]^T = (I_2 \otimes Z) \left\{ D_0 [Z^T B \vec{v}_v^{(0)}, Z^T \vec{v}_v^{(0)}]^T - \sum_{k=1}^{N/2} z_{2,2k}^2 \left(v_0^{(2k)} \vec{p}^{(2k)} + v_1^{(2k)} \vec{q}^{(2k)} \right) \right\}, \tag{4.43}$$

where

$$D_0 = \sum_{k=1}^{N/2} z_{2,2k}^2 D_{2k}^{-1}, \qquad [\vec{p}^{(2k)}, \vec{q}^{(2k)}] = D_{2k}^{-1} (I_2 \otimes Z^T) R_{12}^{(2k)}. \tag{4.44}$$

The presence of B in (4.38), $I_2 \otimes Z$ in (4.43), and (4.4) suggest that we introduce

$$\hat{S} = (I_2 \otimes \Lambda^{-1/2} Z^T) S (I_2 \otimes Z \Lambda^{-1/2}), \quad \hat{r} = (I_2 \otimes \Lambda^{-1/2} Z^T) \vec{r}, \quad \hat{v}_v = (I_2 \otimes \Lambda^{1/2} Z^T) \vec{v}_v, \tag{4.45}$$

and solve the system

$$\hat{S} \hat{v}_v = \hat{r} \tag{4.46}$$

instead of $S\vec{v}_v = \vec{r}$. This approach for computing $\vec{v}_v$ involves one additional multiplication by $I_2 \otimes \Lambda^{-1/2} Z^T$ to obtain $\hat{r}$ and one additional multiplication by $I_2 \otimes Z\Lambda^{-1/2}$ to recover $\vec{v}_v$ from $\hat{v}_v$. However, multiplication of a vector by $\hat{S}$ takes only $4N^2 + O(N)$ operations provided that additional diagonal matrices related to D_0, D_1, and additional vectors related to $\vec{p}^{(2k)}$, $\vec{q}^{(2k)}$, $k = 1, \ldots, N/2$, $\vec{p}^{(2k+1)}$, $\vec{q}^{(2k+1)}$, $k = 1, \ldots, N/2 - 1$, are precomputed with cost $O(N^2)$. (Here D_1 and $\vec{p}^{(2k+1)}$, $\vec{q}^{(2k+1)}$ are counterparts of D_0 and $\vec{p}^{(2k)}$, $\vec{q}^{(2k)}$ for (4.41).)

In the remainder of this section we select a preconditioner for $\hat{S}$ and discuss the solution of a linear system with this preconditioner. First, to select the preconditioner for S of (4.13), we consider the linear system

$$\begin{aligned} (I_{N-1} \otimes B + B \otimes I_{N-1}) \vec{u} + (B \otimes B) \vec{v} + (B_t \otimes B) \vec{v}_v &= \vec{0}, \\ (I_{N-1} \otimes B + B \otimes I_{N-1}) \vec{v} + (B_t \otimes I_{N-1}) \vec{v}_v &= \vec{0}, \\ (B_t^T \otimes I_{N-1}) \vec{u} + (B_t^T \otimes B) \vec{v} + (B_s \otimes B) \vec{v}_v &= \vec{g}_v, \end{aligned} \tag{4.47}$$

where

$$\vec{g}_v = [g_{0,2}, \ldots, g_{0,N}, g_{1,2}, \ldots, g_{1,N}]^T.$$

Note that the matrix in (4.47) is the same as S_{11} of (3.14) but with the roles of the x and y coordinates interchanged. The matrix of (4.47) arises in the spectral Galerkin method for the auxiliary problem (see Figure 3)

$$\begin{aligned} -\Delta u + v = g \text{ in } \Omega, \quad u = 0 \text{ on } \partial\Omega, \qquad \partial u / \partial n = 0 \text{ on } \partial\Omega_v, \\ -\Delta v = -f \text{ in } \Omega, \quad v = 0 \text{ on } \partial\Omega_h, \end{aligned} \tag{4.48}$$

which only differs from (4.1) in the roles of the x and y coordinates. Equations (4.47) can be written as

$$\begin{aligned} P_{11} [\vec{u}, \vec{v}]^T + P_{12} \vec{v}_v &= \vec{0}, \\ P_{21} [\vec{u}, \vec{v}]^T + P_{22} \vec{v}_v &= \vec{g}_v, \end{aligned} \tag{4.49}$$

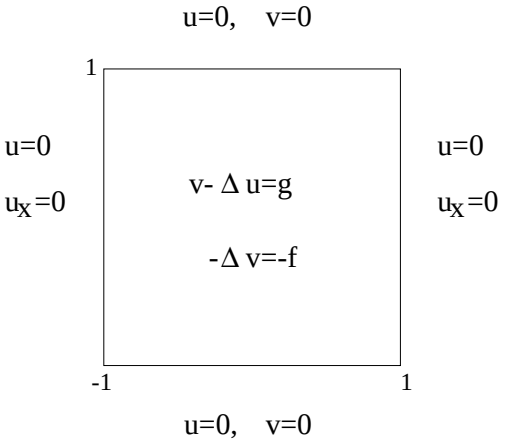


FIG. 3. *Auxiliary problem* (4.48).

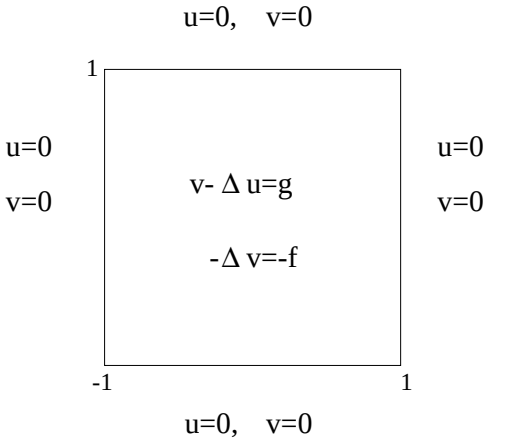


FIG. 4. *Decoupled problem* (4.52).

where

$$P_{11} = \begin{bmatrix} I_{N-1} \otimes B + B \otimes I_{N-1} & B \otimes B \\ O & I_{N-1} \otimes B + B \otimes I_{N-1} \end{bmatrix}, \tag{4.50}$$

and

$$P_{12} = \begin{bmatrix} B_t \otimes B \\ B_t \otimes I_{N-1} \end{bmatrix}, \quad P_{21} = \begin{bmatrix} B_t^T \otimes I_{N-1}, B_t^T \otimes B \end{bmatrix}, \quad P_{22} = \begin{bmatrix} B_s \otimes B \end{bmatrix}. \tag{4.51}$$

The matrix P_{11} arises in the spectral Galerkin method for the decoupled problem (see Figure 4)

$$\begin{aligned} -\Delta u + v &= g \text{ in } \Omega, \quad u = 0 \text{ on } \partial\Omega, \\ -\Delta v &= -f \text{ in } \Omega, \quad v = 0 \text{ on } \partial\Omega. \end{aligned} \tag{4.52}$$

The weak form of (4.52) is

$$\int_\Omega \nabla u \nabla \eta \, d\Omega + \int_\Omega v\eta \, d\Omega = \int_\Omega g\eta \, d\Omega, \qquad \eta \in H_0^1(\Omega),$$

$$\int_\Omega \nabla v \nabla \delta \, d\Omega = -\int_\Omega f\delta \, d\Omega, \qquad \delta \in H_0^1(\Omega),$$

and the spectral Galerkin problem consists of finding $U, V \in X_N^0$ such that

$$\begin{aligned} \int_\Omega \nabla U \nabla \eta \, d\Omega + \int_\Omega V\eta \, d\Omega &= \int_\Omega g\eta \, d\Omega, \qquad \eta \in X_N^0, \\ \int_\Omega \nabla V \nabla \delta \, d\Omega &= -\int_\Omega f\delta \, d\Omega, \qquad \delta \in X_N^0. \end{aligned} \tag{4.53}$$

LEMMA 4.2. *The matrix P_{11} of (4.50) is nonsingular.*

Proof. It is easy to show that (4.53) has a unique solution. ∎

Since P_{11} is nonsingular, eliminating $[\vec{u}, \vec{v}]^T$ from (4.49), we obtain

$$P\vec{v}_v = \vec{g}_v, \tag{4.54}$$

where the $2(N-1) \times 2(N-1)$ Schur complement matrix

$$P = P_{22} - P_{21}P_{11}^{-1}P_{12}. \tag{4.55}$$

(P is the Schur complement of P_{11} in $\left[\begin{smallmatrix} P_{11} & P_{12} \\ P_{21} & P_{22} \end{smallmatrix}\right]$.)

THEOREM 4.2. *The matrix P is symmetric and positive definite.*

Proof. Proof of this theorem is similar to that of Theorem 4.1. For $n = 1, 2$, and arbitrary $\vec{v}_v^{(n)}$ of the form (4.20), let $\vec{u}^{(n)}$ of the form (4.22) and $\vec{v}^{(n)}$ of the form (4.23) be such that

$$P_{11}[\vec{u}^{(n)}, \vec{v}^{(n)}]^T + P_{12}\vec{v}_v^{(n)} = \vec{0}. \tag{4.56}$$

Then (4.55), (4.56), and (4.51) give

$$\begin{aligned} (P\vec{v}_v^{(1)}, \vec{v}_v^{(2)})_{R^{2N-2}} &= (P_{22}\vec{v}_v^{(1)}, \vec{v}_v^{(2)})_{R^{2N-2}} + (P_{21}[\vec{u}^{(1)}, \vec{v}^{(1)}]^T, \vec{v}_v^{(2)})_{R^{2N-2}} \\ &= ((B_s \otimes B)\vec{v}_v^{(1)}, \vec{v}_v^{(2)})_{R^{2N-2}} + ((B_t^T \otimes I_{N-1})\vec{u}^{(1)}, \vec{v}_v^{(2)})_{R^{2N-2}} \\ &\quad + ((B_t^T \otimes B)\vec{v}^{(1)}, \vec{v}_v^{(2)})_{R^{2N-2}}. \end{aligned} \tag{4.57}$$

Equation (4.56) is the matrix-vector form of the spectral Galerkin problem

$$\begin{aligned} \int_\Omega \nabla U^{(n)} \nabla \eta \, d\Omega + \int_\Omega V^{(n)}\eta \, d\Omega = 0, \quad \eta \in X_N^0, \\ \int_\Omega \nabla V^{(n)} \nabla \delta \, d\Omega = 0, \quad \delta \in X_N^0, \end{aligned} \tag{4.58}$$

where $U^{(n)}(x, y)$ is of the from (4.27),

$$V^{(n)}(x, y) = V_i^{(n)}(x, y) + V_v^{(n)}(x, y), \tag{4.59}$$

and $V_i^{(n)}(x, y)$, $V_v^{(n)}(x, y)$ are of the forms (4.29), (4.31), respectively. Since $V_i^{(2)}, U^{(1)} \in X_N^0$, it follows from (4.59) and (4.58) that

$$\begin{aligned} \int_\Omega V^{(1)}V^{(2)} \, d\Omega &= \int_\Omega V^{(1)}V_v^{(2)} \, d\Omega + \int_\Omega V^{(1)}V_i^{(2)} \, d\Omega \\ &= \int_\Omega V^{(1)}V_v^{(2)} \, d\Omega - \int_\Omega \nabla U^{(1)} \nabla \left(V^{(2)} - V_v^{(2)}\right) d\Omega \\ &= \int_\Omega V_i^{(1)}V_v^{(2)} \, d\Omega + \int_\Omega V_v^{(1)}V_v^{(2)} \, d\Omega + \int_\Omega \nabla U^{(1)} \nabla V_v^{(2)} \, d\Omega. \end{aligned} \tag{4.60}$$

TABLE 1
$\kappa_2(\hat{P}^{-1/2}\hat{S}\hat{P}^{-1/2})$ *and* $\kappa_2(\hat{S}_i)$, $i = 1, 2, 3, 4$.

N	16	32	64	128
$\kappa_2(\hat{P}^{-1/2}\hat{S}\hat{P}^{-1/2})$	1.65	1.67	1.68	1.68
$\kappa_2(\hat{S}_1)$	28.68	104.91	400.84	1566.64
$\kappa_2(\hat{S}_2)$	24.39	90.99	350.59	1375.56
$\kappa_2(\hat{S}_3)$	13.93	53.43	209.14	827.35
$\kappa_2(\hat{S}_4)$	13.57	52.54	206.82	820.78

Hence, it follows from (4.57), (4.60), (4.33), (4.35), and (4.36) that

$$(P\vec{v}_v^{(1)}, \vec{v}_v^{(2)})_{R^{2N-2}} = \int_\Omega V^{(1)}V^{(2)}\, d\Omega,$$

which implies the symmetry and positive definiteness of P. □

We take P of (4.55) as a preconditioner for S and

$$\hat{P} = (I_2 \otimes \Lambda^{-1/2}Z^T)P(I_2 \otimes Z\Lambda^{-1/2})$$

as a preconditioner for $\hat{S}$ of (4.45). For arbitrary $\vec{g}_v$, the solution of (4.54) can be obtained by solving (4.49) for $\vec{v}_v$. Equivalently, we compute $\vec{v}_h$ of

$$[\vec{u}, \vec{v}, \vec{v}_h]^T = S_{11}^{-1}[\vec{0}, \vec{0}, \vec{g}_h]^T.$$

(Note that the roles of the x and y coordinates are interchanged, that is, for $k = 2, \ldots, N$, the components $g_{k,0}$, $g_{k,1}$ of $\vec{g}_h$ are equal to the components $g_{0,k}$, $g_{1,k}$ of $\vec{g}_v$ and the components $v_{0,k}$, $v_{1,k}$ of $\vec{v}_v$ are equal to the components $v_{k,0}$, $v_{k,1}$ of $\vec{v}_h$.) It follows from (4.8), (4.6), and the discussion in section 4.1 of the special case of (4.10) that the cost of solving a linear system with $\hat{P}$ is $O(N)$.

With the preconditioner $\hat{P}$, the convergence rate of the PCG method applied to a linear system with $\hat{S}$ depends on $\kappa_2(\hat{P}^{-1/2}\hat{S}\hat{P}^{-1/2})$, where, for a symmetric and positive definite matrix M, $\kappa_2(M) = \lambda_{max}(M)/\lambda_{min}(M)$. A careful analysis shows that $\hat{S}$ splits into four symmetric positive definite matrices $\hat{S}_i$, $i = 1, 2, 3, 4$, and $\hat{P}$ splits into four positive definite diagonal matrices $\hat{P}_i$, $i = 1, 2, 3, 4$. ($\hat{S}_i$, $\hat{P}_i$, $i = 1, 2$, are of order $N/2 \times N/2$ and $\hat{S}_i$, $\hat{P}_i$, $i = 3, 4$, are of order $(N/2-1) \times (N/2-1)$.) Since our preconditioning of $\hat{S}$ by $\hat{P}$ is equivalent to preconditioning each $\hat{S}_i$ by $\hat{P}_i$, we have

$$\kappa_2(\hat{P}^{-1/2}\hat{S}\hat{P}^{-1/2}) = \max_{i=1,2,3,4} \lambda_{max}(\hat{P}_i^{-1/2}\hat{S}_i\hat{P}_i^{-1/2}) / \min_{i=1,2,3,4} \lambda_{min}(\hat{P}_i^{-1/2}\hat{S}_i\hat{P}_i^{-1/2}).$$

This last formula was used to compute $\kappa_2(\hat{P}^{-1/2}\hat{S}\hat{P}^{-1/2})$ numerically for several values of N. Based on the results presented in Table 1, we conjecture that $\kappa_2(\hat{P}^{-1/2}\hat{S}\hat{P}^{-1/2})$ is bounded from above by a positive constant which is independent of N while $\kappa(\hat{S}_i)$, $i = 1, 2, 3, 4$, grows quadratically with N.

Since the matrix $\hat{S}$ of (4.45) is symmetric and positive definite, the system (4.46), equivalently the corresponding systems with $\hat{S}_i$, $i = 1, 2, 3, 4$, can also be solved by Cholesky's method. The columns of $\hat{S}$, equivalently the columns of $\hat{S}_i$, $i = 1, 2, 3, 4$, can be obtained using (4.38) and (4.39) with one component of $\vec{v}_v$ equal to 1 and all remaining components equal to 0. All four matrices $\hat{S}_i$ can be formed with cost $N^3/2 + O(N^2)$ and factored out with cost $N^3/6 + O(N^2)$. Clearly, the cost of the solution stage is $O(N^2)$.

4.4. Cost and memory requirements of the algorithm. We now discuss the cost and memory requirements of solving (3.13) using the algorithm of section 4.2.

The preprocessing stage consists of computing

Λ, Z of (4.3), (4.4) using the LAPACK [1] routines dsteqr and dstein;

$\vec{s}^{(i)}$, δ_i of (2.14) and the factorization of $B_i + \lambda I_{N/2-i}$ for $\lambda = \lambda_k$, $k = 2, \ldots, N$, and $i = 0, 1$;

the diagonal matrices related to D_0, D_1 and the vectors related to $\vec{p}^{(2k)}$, $\vec{q}^{(2k)}$, $k = 1, \ldots, N/2$, $\vec{p}^{(2k+1)}$, $\vec{q}^{(2k+1)}$, $k = 1, \ldots, N/2 - 1$, (cf. (4.44)).

The cost of these computations is $O(N^2)$ and the memory requirements are $11N^2/2$.

In the case of the Cholesky method for solving (4.46), the preprocessing stage also includes the computation of the matrices $\hat{S}_i$, $i = 1, 2, 3, 4$, and their factorizations. The cost of these computations is $2N^3/3$ and the memory requirements are N^2.

From the discussions in sections 4.2 and 4.3, it follows that the cost of the remaining portion of the PCG algorithm for finding $\vec{u}$ is

$$2N^3 + (\text{number of PCG iterations}) \times 4N^2, \tag{4.61}$$

where $4N^2$ is the cost of one PCG iteration. We select $\vec{0}$ as the initial guess for the PCG method. Our numerical tests indicate that the number of PCG iterations should be an integer multiple of $\log N$ if the accuracy of the PCG algorithm is to be comparable with that of the Cholesky algorithm. Hence the total cost of the PCG algorithm for finding $\vec{u}$ is given by (4.61) with the second term dominating the first one for small values of N.

The cost of the remaining portion of the Cholesky algorithm for finding $\vec{u}$ is $2N^3$ which gives the total cost $8N^3/3$ of this algorithm.

The memory requirements for the remaining portions of the PCG and Cholesky algorithms are $3N^2$.

For both the PCG and Cholesky algorithms, the additional cost of computing the vector $\vec{v}$ in Step 8 is $N^3 + O(N^2)$.

5. Numerical results. We solved (1.1) with

$$f(x, y) = 128\pi^4[\cos(4\pi x)\cos(4\pi y) - \sin^2(2\pi x)\cos(4\pi y) - \cos(4\pi x)\sin^2(2\pi y)].$$

The exact solution of this problem, which was also considered by Shen [7] and Bjørstad and Tjøstheim [3], is $u = \sin^2(2\pi x)\sin^2(2\pi y)$.

Our PCG and Cholesky algorithms were implemented in Fortran 77 and numerical experiments were carried out on an IBM RS6000 (processor type: Power PC 604/100MHz), for which the LINPACK TPP benchmark in MFLOPS is 56.4.

The number of PCG iterations in Step 4 of our PCG algorithm was taken to be $3 \log N$. In Table 2 we present the maximum absolute error in u and $v = \Delta u$ on a uniform $(0.02) \times (0.02)$ grid for different values of N. Comparable errors were obtained for the Cholesky algorithm. The exponential convergence achieved is shown in Figure 5 where we present the graph of the logarithm of the maximum absolute error versus N.

Following the format of Table 3 in [7], in Table 3 we present the CPU times in seconds (excluding preprocessing times) for computing $\vec{u}$ by the PCG and Cholesky algorithms. The preprocessing times are given separately in parentheses. As expected from the discussion in section 4.4, preprocessing for the Cholesky algorithm takes

TABLE 2
Maximum absolute error for numerical example.

N	16	20	24	28	32	128	512
$\|u-U\|_\infty$	0.21(−1)	0.22(−3)	0.13(−5)	0.34(−8)	0.51(−11)	0.30(−12)	0.62(−12)
$\|v-V\|_\infty$	0.60(+1)	0.77(−1)	0.53(−3)	0.15(−5)	0.23(−8)	0.93(−10)	0.12(−9)

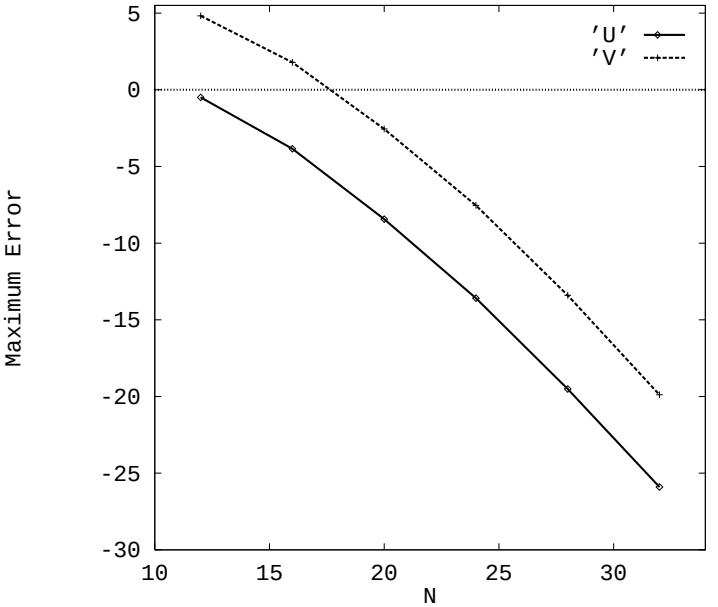


FIG. 5. *Logarithm of the maximum absolute error versus N.*

TABLE 3
Execution time for PCG and Cholesky algorithms.

N	16	32	64	128	512
PCG: CPU(Pre-P)	0.0096 (0.0012)	0.034 (0.012)	0.176 (0.076)	1.076 (0.44)	54.840 (24.844)
Chol: CPU (Pre-P)	0.0072 (0.0052)	0.026 (0.020)	0.134 (0.116)	0.892 (0.70)	48.224 (40.674)

more time than preprocessing for the PCG algorithm, whereas the CPU time for the Cholesky algorithm is smaller than the CPU time for the PCG algorithm. Also, for large values of N the total execution time (CPU + Pre-P) for the PCG algorithm is smaller than that for the Cholesky algorithm.

6. Concluding remarks. As shown in (3.9) of [7], the Legendre spectral Galerkin method based on the standard weak form

$$\int_\Omega \Delta u \Delta \eta \, d\Omega = \int_\Omega f\eta \, d\Omega, \qquad \eta \in H_0^2(\Omega),$$

of (1.1) leads to the linear system

$$(I_{N-3} \otimes B + 2C \otimes C + B \otimes I_{N-3})\vec{u} = \vec{f}, \tag{6.1}$$

where B and C are the Galerkin matrices corresponding to the derivatives of order zero and two in the space $\{p \in P_N^0 : p'(\pm 1) = 0\}$. Shen [7] solves (6.1) using a

capacitance matrix approach with the auxiliary matrix

$$I_{N-3} \otimes B + 2C \otimes C + \tilde{B} \otimes I_{N-3}.$$

The matrix $\tilde{B}$, a modification of B, is such that C and $\tilde{B}$ commute and hence diagonalization in the x direction is possible. The capacitance matrix of [7], which is neither symmetric nor positive definite, is formed explicitly and then factored out using Gauss elimination. The cost of Shen's algorithm is $4N^3 + O(N^2)$ and this count does not include formation and factorization of the capacitance matrix. Both formation and factorization require $O(N^3)$ operations, where the constant multiple of N^3 is not given in [7].

In [3], Bjørstad and Tjøstheim solve (6.1) using the Sherman–Morrison formula with the auxiliary matrix

$$I_{N-3} \otimes B + 2C \otimes C + C^2 \otimes I_{N-3}.$$

Their counterpart of our Schur complement matrix S is symmetric and it is preconditioned by another symmetric matrix. Numerical tests indicate that both matrices are positive definite. The dominant cost of the PCG algorithm of [3] is given by (4.61), where $4N^2$ should be replaced with $24N^2$ [8]. The cost of the Cholesky algorithm in [3] is $16N^3/3 + O(N^2)$.

It follows from section 4.4 that our algorithms, based on discretizing the coupled problem (3.1), are more efficient with respect to operation counts. Our Schur complement matrices S of (4.13) and P of (4.55) are closely related to boundary value problems (3.1), (4.1), and (4.48), (4.52), respectively. This allows us to prove that they are both symmetric and positive definite. Moreover, if required, we also obtain an approximation to $v = \Delta u$ with an additional cost of $N^3 + O(N^2)$.

The algorithms of [3] and [7] were developed for the fourth order problem of the form

$$\Delta^2 u - \beta \Delta u + \alpha u = f \text{ in } \Omega, \qquad u = \partial u/\partial n = 0 \text{ on } \partial\Omega, \tag{6.2}$$

where α and β are nonnegative constants. Our approach, based on introducing $v = \Delta u$, and the corresponding PCG and Cholesky algorithms generalize, with the same dominant costs, to (6.2). In this more general case, the blocks S_{11}, S_{12} of (3.14), (3.15) are to be replaced, respectively, by

$$S_{11} = \begin{bmatrix} I_{N-1} \otimes B + B \otimes I_{N-1} & B \otimes B & B \otimes B_t \\ -\beta(B \otimes B) & (I_{N-1} + \alpha B) \otimes B + B \otimes I_{N-1} & (I_{N-1} + \alpha B) \otimes B_t \\ I_{N-1} \otimes B_t^T & B \otimes B_t^T & B \otimes B_s \end{bmatrix},$$

$$S_{12} = \begin{bmatrix} B_t \otimes B \\ B_t \otimes (I_{N-1} + \alpha B) \\ B_t \otimes B_t^T \end{bmatrix}.$$

Similarly, A_i of (2.11) and $\vec{p}$ of (2.12) are to replaced with

$$A_i = \begin{bmatrix} B_i + \lambda I_{N/2-i} & \lambda B_i \\ -\beta\lambda B_i & (1 + \alpha\lambda) B_i + \lambda I_{N/2-i} \end{bmatrix},$$

$$\vec{p} = [\lambda, 0, \ldots, 0, 1 + \alpha\lambda, 0, \ldots, 0]^T.$$

The new matrix A_i reduces to a block tridiagonal matrix with 2×2 blocks.

Our approach allows also for an efficient treatment of some other boundary conditions, such as $u = \Delta u = 0$ on $\partial\Omega$ or $u = 0$ on $\partial\Omega$, $\partial u/\partial n = 0$ on the horizontal sides of $\partial\Omega$ and $\Delta u = 0$ on the vertical sides of $\partial\Omega$. In fact, as discussed in section 4.1, the last set of boundary conditions gives rise to a linear system with matrix S_{11} of (3.14) and hence its solution, for the approximation to u, can be obtained at a cost of only $2N^3 + O(N^2)$.

REFERENCES

[1] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, and D. Sorensen, *LAPACK Users' Guide*, SIAM, Philadelphia, 1992.

[2] B. Bialecki and A. Karageorghis, *A Legendre spectral collocation method for the biharmonic Dirichlet problem*, Math. Model. Numer. Anal., 34 (2000), pp. 637–662.

[3] P.E. Bjørstad and B.P. Tjøstheim, *Efficient algorithms for solving a fourth-order equation with the spectral-Galerkin method*, SIAM J. Sci. Comput., 18 (1997), pp. 621–632.

[4] P.G. Ciarlet and P.-A. Raviart, *A mixed finite element method for the biharmonic equation*, in Symposium on Mathematical Aspects of Finite Elements in Partial Differential Equations, C. de Boor, ed., Academic Press, New York, 1974, pp. 125–143.

[5] D.B. Knudson, *A Piecewise Hermite Bicubic Finite Element Galerkin Method for the Biharmonic Dirichlet Problem*, Ph.D. thesis, Colorado School of Mines, Golden, CO, 1997.

[6] J. Shen, *Efficient Chebyshev-Legendre Galerkin methods for elliptic problems*, in Proceedings of the Third International Conference on Spectral and High Order Methods, A. V. Ilin and L. R. Scott, eds., Houston J. Math., Houston, TX, 1996, pp. 233–239.

[7] J. Shen, *Efficient spectral-Galerkin method.* I. *Direct solvers of second- and fourth-order equations using Legendre polynomials*, SIAM J. Sci. Comput., 15 (1994), pp. 1489–1505.

[8] B.P. Tjøstheim, *private communication.*

SIAM J. SCI. COMPUT.
Vol. 22, No. 5, pp. 1570–1592
© 2000 Society for Industrial and Applied Mathematics

ALGEBRAIC MULTIGRID BASED ON ELEMENT INTERPOLATION (AMGe)*

M. BREZINA†, A. J. CLEARY‡, R. D. FALGOUT‡, V. E. HENSON‡, J. E. JONES‡, T. A. MANTEUFFEL†, S. F. MCCORMICK†, AND J. W. RUGE†

Abstract. We introduce AMGe, an algebraic multigrid method for solving the discrete equations that arise in Ritz-type finite element methods for partial differential equations. Assuming access to the element stiffness matrices, we have that AMGe is based on the use of two local measures, which are derived from global measures that appear in existing multigrid theory. These new measures are used to determine local representations of algebraically "smooth" error components that provide the basis for constructing effective interpolation and, hence, the coarsening process for AMG. Here, we focus on the interpolation process; choice of the coarse "grids" based on these measures is the subject of current research. We develop a theoretical foundation for AMGe and present numerical results that demonstrate the efficacy of the method.

Key words. algebraic multigrid, finite elements, interpolation weights

AMS subject classifications. 65F10, 65N20, 65N30

PII. S1064827598344303

1. Introduction.

Computer simulations play an increasingly important role in scientific investigations. Indeed, as experimentation becomes more expensive, impracticable, or even proscribed, scientists are turning more and more to numerical simulation. Modern simulation packages are extremely complex, with components spanning many disciplines (e.g., hydrodynamics, radiation transport, structures, thermodynamics, chemistry, and electromagnetics). Also, the problems are frequently posed in multimaterial regimes, with contact surfaces, interpenetrability constraints, and intricate geometries. As a result, codes are being developed to solve complex multiphysics problems on highly resolved, unstructured grids. Such large-grid simulations require the efficient union of massively parallel computing with scalable numerical algorithms such as multigrid (see, e.g., [2]).

An especially effective method for many of the problems that arise in these applications is algebraic multigrid (AMG) [5, 4, 6, 20, 17, 19, 18]. AMG is a method for solving matrix equations that is based on multigrid concepts, but constructs the coarsening process in an algebraic way that requires no explicit knowledge of the geometry. It examines the matrix entries to determine a sequence of smaller matrix

*Received by the editors September 4, 1998; accepted for publication (in revised form) April 18, 2000; published electronically November 17, 2000. This work was performed by an employee of the U.S. Government or under U.S. Government contact. The U.S. Government retains a nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or allow others to do so, for U.S. Government purposes. Copyright is owned by SIAM to the extent not limited by these rights.

http://www.siam.org/journals/sisc/22-5/34430.html

†Applied Math Department, Campus Box 526, University of Colorado at Boulder, Boulder, CO 80309-0526 (mbrezina@colorado.edu, tmanteuf@colorado.edu, stevem@colorado.edu, jruge@colorado.edu). The work of these authors was sponsored in part by the National Science Foundation under grant DMS-9706866 and by the Department of Energy under grant DE-FG03-93ER25165.

‡Center for Applied Scientific Computing, Lawrence Livermore National Laboratory, P.O. Box 808, L-561, Livermore, CA 94551 (cleary@llnl.gov, rfalgout@llnl.gov, vhenson@llnl.gov, jjones@llnl.gov). The work of these authors was performed under the auspices of the U.S. Department of Energy by University of California Lawrence Livermore National Laboratory under contract W-7405-ENG-48.

problems that serve as coarse-level equations. AMG also determines associated inter-level transfer operators (restriction and prolongation), then solves the original matrix equation in a multigrid-like process based on these automatically constructed components. AMG has been shown to be well suited for solving unstructured grid problems and to work well over a wide variety of applications (see, e.g., [10]).

It has been applied successfully to M-matrix problems where the so-called strength of connection is easily measured (this measure is used to determine which variables are strongly representative of the errors left by relaxation, so that they can be used to construct the coarse levels). It also applies well to scalar problems that depart substantially from M-matrix discretizations. However, for problems where strength of connection is not easily measured, AMG is not effective without certain problem-specific modifications or careful parameter tuning. For such cases, there is no systematic AMG approach that has proven effective in any kind of general context. There are still other problems (e.g., thin-body elasticity on unstructured grids) for which AMG and other iterative methods in general have failed to achieve full optimality (i.e., convergence factors bounded uniformly in the size of the problem). The goal of our research is to develop a more robust AMG for solving these difficult problems.

This paper introduces an AMG method for solving partial differential equations discretized by Ritz-type finite element methods. As a departure from standard AMG, where only the operator matrix is required, this approach assumes access to element stiffness matrices. We thus refer to it as AMGe (AMG henceforth refers to the standard scheme). This new approach is based on the use of either of two measures (derived from global measures used in existing theory) to determine algebraically "smooth" error and to construct effective interpolation. AMGe uses a minimization principle based on the *element interpolation* scheme first introduced in [16]. Other multigrid methods, using minimization principles for constructing energetically stable intergrid transfer operators, have recently appeared in [23, 24, 12].

Some notation and the key ideas behind AMG are summarized in the next section. (Nevertheless, we assume that the reader is familiar with AMG methods and terminology. For more details, see [10] and [18].) In particular, we discuss the notion of *strength of dependence* and its role in defining the basic AMG components. In section 3, we define a heuristic based on two *global measures* and establish a corresponding two-level convergence result. We "localize" these measures in section 4 and describe how they can be used to compute the interpolation operator for AMGe. We also discuss the relationship between the local and global measures in subsection 4.3. Section 5 contains numerical results supporting the theory and demonstrating the efficacy of the approach. Concluding remarks are made in section 6.

2. Preliminaries. We begin this section by describing notation. Capital italic Roman letters (A, B, P, R) denote matrices and bold lowercase Roman and Greek letters denote vectors ($\mathbf{u}, \mathbf{v}, \boldsymbol{\varepsilon}$). The ith component of the vector $\mathbf{q}$ is denoted by q_i. Other lowercase letters denote scalars, while capital calligraphic letters denote sets and spaces ($\mathcal{C}, \mathcal{F}, \mathcal{S}$), with the singular exception that $\mathcal{A}$ is used to denote finite element stiffness matrices. We define the A-inner product by $\langle \cdot,\ \cdot \rangle_A := \langle A\cdot,\ \cdot \rangle$, where $\langle \cdot,\ \cdot \rangle$ is the standard Euclidean inner product, and the A-norm (also called the energy norm) by $\|\cdot\|_A := \langle \cdot,\ \cdot \rangle_A^{1/2}$.

Assume that we are given an $n \times n$ symmetric positive definite matrix A expressed as the sum of a given set of finite element stiffness matrices: $A = \sum_{\alpha \in \mathcal{T}} \mathcal{A}_\alpha$, where $\mathcal{T}$ is the set of finite elements used to discretize the problem and each $\mathcal{A}_\alpha$ is symmetric positive semidefinite. We do not assume access to a spatial grid or the ability to create

new finite element stiffness matrices.

We seek the solution $\mathbf{u} \in \mathbb{R}^n$ to the linear system

$$A\mathbf{u} = \mathbf{f} \tag{2.1}$$

for a given $\mathbf{f} \in \mathbb{R}^n$. Standard iterative schemes, like Gauss–Seidel and Krylov space methods, tend to converge slowly for large-scale problems of this type that arise from partial differential equations. The difficulty is that smooth error components are typically attenuated very slowly by these simple processes, because they are based on local properties (i.e., local connections in A). Multigrid methods attempt to correct this limitation by representing the smooth errors on increasingly coarser, and, therefore, more global levels.

To describe how system (2.1) could be solved by a multilevel method, let P be an $n \times n_c$ *interpolation* or *prolongation* matrix that transfers level n_c corrections to level n, with $n_c < n$. P could be determined *geometrically* by, say, linear interpolation (cf. [7]) when $\mathbb{R}^{n_c}$ and $\mathbb{R}^n$ represent grids whose nodal positions are accessible. P could instead be determined *algebraically* by so-called operator interpolation (cf. [1]), which is based on the entries of A. In any event, we choose P^T as the *restriction* matrix that transfers level n residuals to level n_c. The two-grid method for solving (2.1) is then defined as follows:

(2.2a) Relax ν_1 times on $A\mathbf{u} = \mathbf{f}$.

(2.2b) Correct $\mathbf{u} \leftarrow \mathbf{u} + P(P^TAP)^{-1}P^T(\mathbf{f} - A\mathbf{u})$.

(2.2c) Relax ν_2 times on $A\mathbf{u} = \mathbf{f}$.

Note the use of P^TAP in correction step (2.2b). This so-called Galerkin coarse-grid operator, together with the use of P^T as the restriction operator, amounts to a *variational* form of multigrid. When A is symmetric, the correction step minimizes the energy norm of the fine-grid error over all possible corrections from the range of P (cf. [7]). To solve (2.1) in practice, one would use a multilevel method that recursively applies algorithm (2.2) to solve the linear system involving P^TAP in correction step (2.2b).

Further examination of (2.2) reveals that relaxation and coarse-grid correction must be chosen to complement each other: An error not reduced by one must be reduced by the other. In this paper, we fix the choice of relaxation, then determine interpolation. The relaxation we choose is a simple pointwise method, like Richardson, damped Jacobi, or Gauss–Seidel, that satisfies the following heuristic:

H1: *Error in the direction of an eigenvector associated with a large eigenvalue is rapidly reduced by relaxation, while error in the direction of an eigenvector associated with a small eigenvalue is reduced by a factor that may approach* 1 *as the eigenvalue approaches* 0.

An error that is not rapidly reduced by relaxation is called *algebraically smooth*. The actual character of algebraically smooth error depends on the operator and the type of relaxation, but it loosely means that the residual is small when compared with the error itself (we will be more precise about this shortly). This does not mean that the error is smooth in any geometric sense. Thus, an error at a point may be very different from the errors at neighboring points, yet it might be difficult to reduce the error by relaxation. Such is the case for anisotropic problems, where an algebraically smooth error that pointwise Gauss–Seidel relaxation cannot effectively

reduce can be geometrically oscillatory in the direction of small coefficients of the differential equation. In any case, the interpolation matrix, P, must be defined so that an algebraically smooth error is effectively eliminated in step (2.2b) and the coarse-grid equations, which involve $P^T AP$, are amenable to solution.

2.1. AMG. To define the multigrid components in AMG, we use the following heuristic (cf. [5, **?**, 18]) based on special properties of M-matrices:

H2: *Smooth error varies slowest in the direction of strong dependence.*

Here, we say that *unknown i strongly depends on unknown j* if

$$-a_{i,j} \geq \theta \max_{k \neq i} \{-a_{i,k}\} \quad \text{for some fixed } \theta \in (0,1). \tag{2.3}$$

Thus, strong dependence is characterized by matrix coefficients that are large in the sense of (2.3). A typical choice for parameter θ is 0.25.

Although AMG was developed with M-matrices in mind, in practice it is not limited to this class of problems. However, the standard method does rely on **H2**, and our sense of strong dependence may not be suitable for many important classes of problems. For example, one simple problem with which standard AMG has difficulty is the Poisson equation on a rectangular grid, discretized with bilinear quadrilateral elements, where the fine-grid elements are stretched to a 10:1 aspect ratio. This yields the coefficient stencil

$$\begin{bmatrix} -1 & -3.9 & -1 \\ 1.9 & 8 & 1.9 \\ -1 & -3.9 & -1 \end{bmatrix}. \tag{2.4}$$

In (2.4), it is not readily apparent from the size of the off-diagonal entries that the direction of strongest dependence is vertical. Since **H2** is used to define all the AMG components, and it requires a clear understanding of strong dependence, AMG can exhibit degraded performance (see Table 5.2). For this simple case, slow convergence of AMG can be ameliorated by simply tuning its parameters (e.g., setting $\theta = 0.5$) or by more elaborate algorithmic "fixes" (e.g., *iterative weight interpolation* [10] or geometric/algebraic interpolation methods [11, 9, 8]). Another approach is to replace **H2** by a heuristic that leads to a more robust AMG algorithm. Exploring this possibility, as we begin to do in the next section, is the primary aim of this paper.

3. Global measures and convergence bounds. This paper takes a slightly different approach, using a heuristic based not on M-matrices, but on the eigenvectors of A. In a two-grid scheme, coarse-grid correction will completely eliminate error in Range (P), the range of the interpolation operator. To complement the action of relaxation, which satisfies **H1**, the interpolation matrix must satisfy the following heuristic:

H3: *Interpolation must be able to approximate an eigenvector with error bound proportional to the size of the associated eigenvalue.*

To make **H3** more rigorous, define $Q : \mathbb{R}^n \to \mathbb{R}^n$ to be a convenient projection onto Range (P), that is,

$$Q = PR \tag{3.1}$$

for some restriction operator $R : \mathbb{R}^n \to \mathbb{R}^{n_c}$ such that $RP = I_c$, the identity on $\mathbb{R}^{n_c}$. The specific form for Q (and, hence, R) will not become important until section 4.

For any vector $\mathbf{e} \in \text{Range}\,(P)$, we have $Q\mathbf{e} = \mathbf{e}$. Thus, $I - Q$ can be used to measure the defect of interpolation. With this in mind, we now define two measures of how well **H3** is satisfied:

$$M_1(Q, \mathbf{e}) := \frac{\langle (I-Q)\mathbf{e},\ (I-Q)\mathbf{e}\rangle}{\langle A\mathbf{e},\ \mathbf{e}\rangle}, \tag{3.2}$$

$$M_2(Q, \mathbf{e}) := \frac{\langle A(I-Q)\mathbf{e},\ (I-Q)\mathbf{e}\rangle}{\langle A\mathbf{e},\ A\mathbf{e}\rangle}. \tag{3.3}$$

Measure M_2 was used in the early multigrid theory [15, 13, 14] to establish optimal convergence of the V-cycle algorithm under full regularity assumptions on the associated partial differential equation. Measure M_1 was introduced in [4] and used more recently to establish convergence, independent of the coarse-grid size, of a two-level method for linear elasticity [22]. It is also an essential ingredient of the regularity-free multilevel theory found in [3]. We develop the relevant two-grid theory here for both measures so that we can tailor the results to our needs.

It has not been our practice to use diagonal conditioning of A in standard AMG. Such a scaling generally changes the nature of smooth errors. Since current schemes at some point rely on a premise of how a smooth error behaves (e.g., that it is locally constant), then diagonal scaling can make it more difficult for AMG to handle. However, no such premise of smoothness is made anywhere in AMGe. Thus, in the remainder of this paper, we are free to assume for convenience that matrix A has been scaled so that its diagonal is the identity. For a general symmetric positive-definite matrix with diagonal $D \neq I$, this can be assured by a diagonal scaling that replaces A by $D^{-1/2}AD^{-1/2}$. Note that this transformation must be considered in the representation of A as a sum of local stiffness matrices, but this is just a straightforward rescaling of the variables. This scaling does, however, bear on the practicality of our results because we analyze AMG based on Richardson iteration, which is not generally a good smoother for matrices that have widely varying diagonal entries. Thus, if diagonal scaling is not used, then in general it would be wise to use a relaxation scheme like damped Jacobi and adjust measures M_1 and M_2 accordingly.

Our theory assumes that either M_1 or M_2 is bounded uniformly in $\mathbf{e} \in \mathbb{R}^n \backslash \{\mathbf{0}\}$. To see how this assumption relates to **H3**, suppose that $\mathbf{e}$ is an eigenvector of A corresponding to a small eigenvalue. Then, for M_1 or M_2 to be bounded, since the denominators of the two measures are small, the numerators must also be small. Thus, Q must accurately interpolate eigenvectors belonging to small eigenvalues. On the other hand, if $\mathbf{e}$ is an eigenvector of A corresponding to a large eigenvalue, then the denominators of the two measures are large, so the numerators may be large. Thus, Q need not accurately interpolate eigenvectors belonging to large eigenvalues.

We now prove convergence results based on M_1 or M_2 for two-level algorithm (2.2).

LEMMA 3.1. *Let Q be any projection onto* Range (P). *Assume that either of the following two approximation properties are satisfied for some constant K:*

$$M_1(Q, \mathbf{e}) \leq K \quad \forall \mathbf{e} \in \mathbb{R}^n \backslash \{\mathbf{0}\}, \tag{3.4}$$

$$M_2(Q, \mathbf{e}) \leq K \quad \forall \mathbf{e} \in \mathbb{R}^n \backslash \{\mathbf{0}\}. \tag{3.5}$$

If $\mathbf{e} \neq 0$ is A-orthogonal to Range (P), *then*

$$\frac{1}{K} \leq \frac{\|A\mathbf{e}\|^2}{\langle A\mathbf{e},\ \mathbf{e}\rangle} \leq \|A\|. \tag{3.6}$$

Proof. The upper bound in (3.6) follows easily from the definition of the matrix norm. To prove the lower bound, note that Range (Q) = Range (P). Hence, if $\mathbf{e}$ is A-orthogonal to Range (P), then

$$\langle A\mathbf{e},\ Q\mathbf{v}\rangle = 0 \quad \forall \mathbf{v} \in \mathbb{R}^n. \tag{3.7}$$

First, assume that (3.4) holds. From (3.7) and the Cauchy–Schwarz inequality, we have

$$\begin{aligned}\langle A\mathbf{e},\ \mathbf{e}\rangle &= \langle A\mathbf{e},\ (I-Q)\mathbf{e}\rangle \\ &\le \|A\mathbf{e}\|\,\|(I-Q)\mathbf{e}\| \\ &\le \|A\mathbf{e}\|\,\langle A\mathbf{e},\ \mathbf{e}\rangle^{1/2}\,K^{1/2}.\end{aligned}$$

The lower bound in (3.6) now follows by dividing through by $\langle A\mathbf{e},\ \mathbf{e}\rangle\,K^{1/2}$ and squaring the result.

Now, assume that (3.5) holds. From (3.7) and the Cauchy–Schwarz inequality, we have

$$\begin{aligned}\langle A\mathbf{e},\ \mathbf{e}\rangle &\le \langle A\mathbf{e},\ \mathbf{e}\rangle + \langle AQ\mathbf{e},\ Q\mathbf{e}\rangle \\ &= \langle A\mathbf{e},\ \mathbf{e}\rangle - \langle A\mathbf{e},\ Q\mathbf{e}\rangle - \langle AQ\mathbf{e},\ \mathbf{e}\rangle + \langle AQ\mathbf{e},\ Q\mathbf{e}\rangle \\ &= \langle A(I-Q)\mathbf{e},\ (I-Q)\mathbf{e}\rangle \\ &\le \|A\mathbf{e}\|^2\,K.\end{aligned}$$

The lower bound in (3.6) now follows by dividing through by $\langle A\mathbf{e},\ \mathbf{e}\rangle\,K$. ∎

Denote the A-orthogonal projection onto the Range (P) by S. Thus,

$$S := P(P^TAP)^{-1}P^TA. \tag{3.8}$$

The error propagation matrix for the coarse-grid correction step (2.2b) is $I-S$. A Richardson iteration with step-size parameter $s = \omega/\|A\|$, $\omega \in (0,2)$, has the error propagation matrix $G = I - sA$. If we choose $(\nu_1,\nu_2) = (0,1)$ in (2.2), then the associated error propagation matrix for this simple two-grid scheme is $G(I-S)$. The following theorem analyzes its convergence by bounding its error propagation matrix in the A-norm. Convergence results for other values of (ν_1,ν_2) then follow naturally [14].

Analogous multilevel results can be found in [15, 13, 14] for approximation property (3.5), and in [3, 21] for (3.4) under the additional assumption of *energetic stability of interpolation*, a sufficient condition for which is that $\|P(P^TP)^{-1}P^T\|_A$ be bounded uniformly on all levels.

THEOREM 3.2. *Assume that either approximation property* (3.4) *or* (3.5) *is satisfied for some constant* K. *Then*

$$\|G(I-S)\|_A \le \left(1 - \frac{\omega(2-\omega)}{K\,\|A\|}\right)^{1/2}. \tag{3.9}$$

Proof. First note that (3.6) implies $K \ge 1/\|A\| \ge \omega(2-\omega)/\|A\|$, so that (3.9) makes sense. We have

$$\begin{aligned}\langle AG\mathbf{e},\ G\mathbf{e}\rangle &= \langle A\mathbf{e},\ \mathbf{e}\rangle - 2s\,\langle A\mathbf{e},\ A\mathbf{e}\rangle + s^2\,\langle A^2\mathbf{e},\ A\mathbf{e}\rangle \\ &\le \langle A\mathbf{e},\ \mathbf{e}\rangle - \frac{\omega(2-\omega)}{\|A\|}\,\langle A\mathbf{e},\ A\mathbf{e}\rangle.\end{aligned}$$

Replacing $\mathbf{e}$ with $(I-S)\mathbf{e}$ and applying the result in Lemma 3.1 yields

$$\begin{aligned}\|G(I-S)\mathbf{e}\|_A^2 &\leq \langle A(I-S)\mathbf{e},\ (I-S)\mathbf{e}\rangle - \frac{\omega(2-\omega)}{\|A\|}\|A(I-S)\mathbf{e}\|^2 \\ &\leq \left(1-\frac{\omega(2-\omega)}{K\|A\|}\right)\|\mathbf{e}\|_A^2 . \qquad \square\end{aligned}$$

Notice that the bound on the convergence factor approaches 1 as K becomes large. Conversely, smaller K yields a smaller bound on the convergence factor. Our aim is to determine P so that, for some appropriate Q, either (3.4) or (3.5) is satisfied for a reasonably small K.

We also remark that the above results can be generalized to apply when (2.1) is a consistent system with symmetric positive semidefinite matrix A. Measures M_1 and M_2 must be restricted to $\mathbf{e} \notin \text{Null}(A)$. A finite bound K in (3.4) or (3.5) then implies that interpolation is exact for $\mathbf{e} \in \text{Null}(A)$, which in turn implies that the correction step involves a consistent system. A zero initial guess and relaxation using a polynomial method like Richardson iteration ensures that the approximate solution remains orthogonal to Null(A).

4. Interpolation using local measures. Quantities M_1 and M_2 are global measures of the quality of interpolation. Our intent is to use these measures to determine an effective strategy for constructing interpolation in AMG, but it is not practical to do this globally. In this section, we discuss an approach for localizing these measures for linear systems (2.1) that arise from finite element discretizations.

Recall that A is given as the sum of finite element stiffness matrices: $A = \sum_{\alpha\in\mathcal{T}} \mathcal{A}_\alpha$. Now, we do not assume access to an underlying spatial grid. However, we can construct an artificial grid based on the graph associated with A, with vertices $\mathcal{G} := \{1, 2, \ldots, n\}$ and edges $\mathcal{E} := \{(i,j) \ : \ a_{ij} \neq 0 \ \text{ for } i \neq j\}$. Grid point (vertex) $i \in \mathcal{G}$ is associated with unknown u_i.

We first define the point set of an element:

$$\mathcal{M}_\alpha := \left\{ j \ : \ \boldsymbol{\varepsilon}_j^T \mathcal{A}_\alpha \boldsymbol{\varepsilon}_j \neq 0 \right\}, \tag{4.1}$$

where $\boldsymbol{\varepsilon}_j$ is the canonical basis vector associated with unknown j. Next, define the neighborhood of grid point i as the set of elements and set of points

$$\mathcal{T}_i := \{\alpha \in \mathcal{T} : \boldsymbol{\varepsilon}_i^T \mathcal{A}_\alpha \boldsymbol{\varepsilon}_i \neq 0\}, \tag{4.2}$$

$$\mathcal{N}_i := \cup_{\alpha\in\mathcal{T}_i} \mathcal{M}_\alpha, \tag{4.3}$$

respectively (see Figure 4.1). Define the local matrices on neighborhood i by

$$A_i = \sum_{\alpha\in\mathcal{T}_i} \mathcal{A}_\alpha. \tag{4.4}$$

We also assume that a coarse grid has been selected; that is, the points in $\mathcal{G}$ have been partitioned into coarse-grid points $\mathcal{C}$ and fine-grid points $\mathcal{F}$ such that $\mathcal{C} \cup \mathcal{F} = \mathcal{G}$ and $\mathcal{C} \cap \mathcal{F} = \emptyset$. We now seek the $n \times n_c$ interpolation matrix P, where $n_c = |\mathcal{C}|$, that interpolates from the coarse-grid points $\mathcal{C}$ to the entire grid $\mathcal{G}$.

Two conflicting goals drive the construction of P. The first is to minimize the bound on measure M_1 or M_2, while the second is to control the sparsity of the coarse-grid system involving P^TAP. Focusing on the second goal first, we assume that

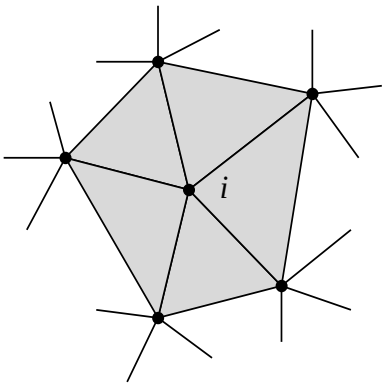


FIG. 4.1. *Local neighborhoods.*

the coarse-grid points interpolate to themselves exactly; that is, P restricted to $\mathcal{C}$ is the identity, while fine-grid points interpolate only from coarse-grid points in their neighborhood, that is, from $\mathcal{C}_i := \mathcal{N}_i \cap \mathcal{C}$.

To make the construction more clear, suppose that the rows and columns of A have been arranged so that the fine-grid points come first, followed by the coarse-grid points. We may then write A in block form as follows:

$$A = \begin{bmatrix} A_{ff} & A_{fc} \\ A_{cf} & A_{cc} \end{bmatrix}. \tag{4.5}$$

In this context, the interpolation matrix has the block form

$$P = \begin{bmatrix} P_{fc} \\ I_c \end{bmatrix}. \tag{4.6}$$

Alternatively, we may define the projection

$$Q = \begin{bmatrix} 0 & P_{fc} \\ 0 & I_c \end{bmatrix}, \tag{4.7}$$

which implies the choice of $R = [0, I_c]$ as the restriction in (3.1).

In what follows, we develop a strategy for constructing the rows of P_{fc}, that is, the rows of Q corresponding to each point $i \in \mathcal{F}$, which we denote

$$\mathbf{q}_i^T := \boldsymbol{\varepsilon}_i^T Q. \tag{4.8}$$

Restricting interpolation to a neighborhood of coarse-grid points is equivalent to choosing

$$\mathbf{q}_i \in \mathcal{Z}_i := \{\mathbf{v} \in \mathbb{R}^n \ : \ v_j = 0 \text{ for } j \notin \mathcal{C}_i\}. \tag{4.9}$$

We now localize measures M_1 and M_2 by defining

$$M_{i,1}(Q, \mathbf{e}) := \frac{\langle \boldsymbol{\varepsilon}_i \boldsymbol{\varepsilon}_i^T (I - Q)\mathbf{e},\ \boldsymbol{\varepsilon}_i \boldsymbol{\varepsilon}_i^T (I - Q)\mathbf{e} \rangle}{\langle A_i \mathbf{e},\ \mathbf{e} \rangle}, \tag{4.10}$$

$$M_{i,2}(Q, \mathbf{e}) := \frac{\langle A_i \boldsymbol{\varepsilon}_i \boldsymbol{\varepsilon}_i^T (I - Q)\mathbf{e},\ \boldsymbol{\varepsilon}_i \boldsymbol{\varepsilon}_i^T (I - Q)\mathbf{e} \rangle}{\langle A_i \mathbf{e},\ A_i \mathbf{e} \rangle} \tag{4.11}$$

for any $\mathbf{e} \notin \text{Null}(A_i)$. Notice for $i \in \mathcal{C}$ that $M_{i,1} = M_{i,2} = 0$, while for $i \in \mathcal{F}$ the above measures depend only on the ith row of Q, which is to be chosen in $\mathcal{Z}_i$. To emphasize this dependence, when the meaning is clear we write

$$M_{i,1}(\mathbf{q_i}, \mathbf{e}) = \frac{\langle (\boldsymbol{\varepsilon}_i - \mathbf{q_i})^T \mathbf{e},\ (\boldsymbol{\varepsilon}_i - \mathbf{q_i})^T \mathbf{e} \rangle}{\langle A_i \mathbf{e},\ \mathbf{e} \rangle}, \tag{4.12}$$

$$M_{i,2}(\mathbf{q_i}, \mathbf{e}) = \frac{\langle (\boldsymbol{\varepsilon}_i - \mathbf{q_i})^T \mathbf{e},\ (\boldsymbol{\varepsilon}_i - \mathbf{q_i})^T \mathbf{e} \rangle}{\langle A_i \mathbf{e},\ A_i \mathbf{e} \rangle} \tag{4.13}$$

for $\mathbf{q_i} \in \mathcal{Z}_i$ and $\mathbf{e} \notin \text{Null}(A_i)$. (Recall that A has unit diagonal.)

Heuristic **H3**, as applied to these local measures, now relates interpolation accuracy to local eigenvectors of A_i. This makes it practical to use $M_{i,1}$ and $M_{i,2}$ to compute interpolation. Since we wish to make these local measures small, interpolation is defined so that the $\mathbf{q}_i$ in (4.8) is the arg min (that is, the argument that attains the minimum) of one of the following min-max problems:

$$K_{i,p} := \min_{\mathbf{q_i} \in \mathcal{Z}_i} \max_{\mathbf{e} \notin \text{Null}(A_i)} M_{i,p}(\mathbf{q_i}, \mathbf{e}) \tag{4.14}$$

for $p = 1$ or 2. Note that if there exists a $\mathbf{q_i} \in \mathcal{Z}_i$ that yields $K_{i,p} < \infty$, then $\mathbf{q_i}$ satisfies the constraint

$$(\boldsymbol{\varepsilon}_i - \mathbf{q_i})^T \mathbf{e} = 0 \quad \forall \mathbf{e} \in \text{Null}(A_i).$$

Thus, min-max problem (4.14) can be restated as the constrained min-max problem

$$\begin{aligned} K_{i,p} &= \min_{\mathbf{q_i} \in \mathcal{Z}_i} \max_{\mathbf{e} \perp \text{Null}(A_i)} M_{i,p}(\mathbf{q_i}, \mathbf{e}) \\ &\text{subject to} \quad (\boldsymbol{\varepsilon}_i - \mathbf{q_i})^T \mathbf{e} = 0 \quad \forall \mathbf{e} \in \text{Null}(A_i) \end{aligned} \tag{4.15}$$

for $p = 1$ or 2. The next two subsections focus on solving these min-max problems. In section 4.3, we relate the local measures to the global measures.

4.1. Computing interpolation by fitting eigenvectors. One way to compute the $\mathbf{q}_i$ in (4.14) or (4.15) is to "fit" the eigenvectors of A_i, as quantified in the following theorem.

THEOREM 4.1. *Suppose we have computed the eigendecomposition*

$$A_i V_i = V_i \Lambda_i, \quad V_i^T V_i = I. \tag{4.16}$$

The columns of V_i are the orthonormalized eigenvectors of A_i, and Λ_i is the diagonal matrix formed from the corresponding eigenvalues. Assume that this eigen-decomposition is ordered to distinguish between zero eigenvalues and positive eigenvalues (that form the diagonal matrix Λ_{i+}):

$$V_i = \begin{bmatrix} V_{i0} & V_{i+} \end{bmatrix}, \quad \Lambda_i = \begin{bmatrix} 0 & 0 \\ 0 & \Lambda_{i+} \end{bmatrix}. \tag{4.17}$$

Then min-max *problem* (4.15) *is equivalent to the following constrained least-squares problem:*

$$\min_{\mathbf{q}_i} \left\| \Lambda_{i+}^{-p/2} V_{i+}^T (\boldsymbol{\varepsilon}_i - \mathbf{q}_i) \right\|^2 \quad \textit{subject to} \quad V_{i0}^T (\boldsymbol{\varepsilon}_i - \mathbf{q}_i) = 0 \tag{4.18}$$

for $p = 1$ or 2.

Proof. Note that the null-space constraint in (4.15) is equivalent to that in (4.18). Assume first that $\mathbf{q}_i$ satisfies (4.15) with $p = 1$. Since $\mathbf{e} \perp \text{Null}(A_i)$, we can write $\mathbf{e} = V_{i+}\Lambda_{i+}^{-1/2}\mathbf{w}$, which yields

$$\min_{\mathbf{q}_i \in \mathcal{Z}_i} \max_{\mathbf{e} \perp \text{Null}(A_i)} M_{i,1}(\mathbf{q}_i, \mathbf{e}) = \min_{\mathbf{q}_i \in \mathcal{Z}_i} \max_{\mathbf{w}} \frac{\left\| (\boldsymbol{\varepsilon}_i - \mathbf{q}_i)^T V_{i+}\Lambda_{i+}^{-1/2}\mathbf{w} \right\|^2}{\|\mathbf{w}\|^2}$$
$$= \min_{\mathbf{q}_i \in \mathcal{Z}_i} \left\| \Lambda_{i+}^{-1/2} V_{i+}^T (\boldsymbol{\varepsilon}_i - \mathbf{q}_i) \right\|^2 .$$

Assume now that $\mathbf{q}_i$ satisfies (4.15) with $p = 2$. Writing $\mathbf{e} = V_{i+}\Lambda_{i+}^{-1}\mathbf{w}$, we then have

$$\min_{\mathbf{q}_i \in \mathcal{Z}_i} \max_{\mathbf{e} \perp \text{Null}(A_i)} M_{i,2}(\mathbf{q}_i, \mathbf{e}) = \min_{\mathbf{q}_i \in \mathcal{Z}_i} \max_{\mathbf{w}} \frac{\left\| (\boldsymbol{\varepsilon}_i - \mathbf{q}_i)^T V_{i+}\Lambda_{i+}^{-1}\mathbf{w} \right\|^2}{\|\mathbf{w}\|^2}$$
$$= \min_{\mathbf{q}_i \in \mathcal{Z}_i} \left\| \Lambda_{i+}^{-1} V_{i+}^T (\boldsymbol{\varepsilon}_i - \mathbf{q}_i) \right\|^2 . \qquad \square$$

Computing the interpolation weights $\mathbf{q}_i$ using (4.18) requires eigendecomposition (4.16), which is not the most efficient method. We introduce a simpler approach in the next subsection. However, we include this notion of fitting eigenvectors because it is useful for understanding the basic principles involved in selecting interpolation.

4.2. A more practical algorithm for computing interpolation. Here we describe a practical algorithm for determining when (4.14) or (4.15) has a (unique) solution for $i \in \mathcal{F}$, and for computing Q when a solution does exist. One important consequence of this characterization is its update property: Whenever the solution with the current interpolatory set does not exist, we can add points to $\mathcal{C}_i$ and test again for solvability without redoing all of the computation.

Assume first that grid point $i \in \mathcal{F}$ has a neighborhood, as depicted in Figure 4.1, consisting of n_i points in set $\mathcal{N}_i$, with n_f fine-grid points and n_c coarse-grid points in C_i. Next, order the unknowns and equations of matrix A_i so that unknown i is first, followed by the other fine-grid points, with the coarse-grid points last. The neighborhood matrix and its square can then be written as

$$A_i = \begin{bmatrix} A_{ff}^{(1)} & A_{fc}^{(1)} \\ A_{cf}^{(1)} & A_{cc}^{(1)} \end{bmatrix} \quad \text{and} \quad A_i^2 = \begin{bmatrix} A_{ff}^{(2)} & A_{fc}^{(2)} \\ A_{cf}^{(2)} & A_{cc}^{(2)} \end{bmatrix},$$

respectively, and $\boldsymbol{\varepsilon}_i$ becomes $\boldsymbol{\varepsilon}_1$.

In the remainder of this subsection, we drop the subscript i whenever the meaning is clear. Set $\mathcal{Z}_i$ restricted to the neighborhood becomes

$$\mathcal{Z} := \{\mathbf{e} \in \mathbb{R}^{n_i} \ : \ e_j = 0 \ \forall j \notin C_i\} .$$

We can then interpret (4.15) with $p = 1$ or 2 as the problem of determining a vector $\mathbf{q} \in \mathcal{Z}$ that minimizes $\max_{\mathbf{e} \notin \text{Null}(A_i)} M_{i,p}(\mathbf{q}, \mathbf{e})$ subject to the constraint

$$(\boldsymbol{\varepsilon}_1 - \mathbf{q})^T \mathbf{e} = 0 \qquad \forall\, \mathbf{e} \in \text{Null}(A_i) = \text{Null}(A_i^2).$$

That is, we require

$$\boldsymbol{\varepsilon}_1 - \mathbf{q} \in \text{Range}\,(A_i) = \text{Range}\left(A_i^2\right). \tag{4.19}$$

Our first concern is the existence of such a vector $\mathbf{q}$. For this, we let $\hat{\boldsymbol{\varepsilon}}_1 \in \mathbb{R}^{n_f}$ denote the first canonical basis vector of length n_f.

LEMMA 4.2. *There exists* $\mathbf{q} \in \mathcal{Z}$ *such that* $\boldsymbol{\varepsilon}_1 - \mathbf{q} \in \text{Range}\,(A_i^p)$ *if and only if*

$$\hat{\boldsymbol{\varepsilon}}_1 \in \text{Range}\left(A_{ff}^{(p)}\right),$$

with $p = 1$ *or* 2.

Proof. Assume that $\hat{\boldsymbol{\varepsilon}}_1 \in \text{Range}\left(A_{ff}^{(p)}\right)$ so that

$$\hat{\boldsymbol{\varepsilon}}_1 = A_{ff}^{(p)} \hat{\boldsymbol{\delta}}_1$$

for some $\hat{\boldsymbol{\delta}}_1 \in \mathbb{R}^{n_f}$. Then

$$A_i^p \boldsymbol{\delta} := \begin{bmatrix} A_{ff}^{(p)} & A_{fc}^{(p)} \\ A_{cf}^{(p)} & A_{cc}^{(p)} \end{bmatrix} \begin{pmatrix} \hat{\boldsymbol{\delta}}_1 \\ 0 \end{pmatrix} = \boldsymbol{\varepsilon}_1 - \mathbf{q} \ \in \text{Range}\,(A_i^p)\,,$$

and $\mathbf{q} \in \mathcal{Z}$.

Conversely, suppose there exists $\mathbf{q} \in \mathcal{Z}$ such that $\boldsymbol{\varepsilon}_1 - \mathbf{q} \in \text{Range}\,(A_i^p)$; that is, there exists $\boldsymbol{\delta}$ such that

$$\boldsymbol{\varepsilon}_1 - \mathbf{q} = A_i^p \boldsymbol{\delta}.$$

This, in turn, implies that

$$\hat{\boldsymbol{\varepsilon}}_1 = \left[A_{ff}^{(p)},\ A_{fc}^{(p)}\right] \boldsymbol{\delta} \ \in \text{Range}\left(\left[A_{ff}^{(p)},\ A_{fc}^{(p)}\right]\right).$$

The proof will be completed by demonstrating that

$$\text{Range}\left(\left[A_{ff}^{(p)},\ A_{fc}^{(p)}\right]\right) = \text{Range}\left(A_{ff}^{(p)}\right).$$

This is certainly true if $A_{ff}^{(p)}$ is nonsingular. Assume otherwise, and let $\hat{\boldsymbol{\delta}}$ be a nonzero vector in $\text{Null}(A_{ff}^{(p)})$. Then

$$\left\langle \begin{bmatrix} A_{ff}^{(p)} & A_{fc}^{(p)} \\ A_{cf}^{(p)} & A_{cc}^{(p)} \end{bmatrix} \begin{pmatrix} \hat{\boldsymbol{\delta}} \\ 0 \end{pmatrix},\ \begin{pmatrix} \hat{\boldsymbol{\delta}} \\ 0 \end{pmatrix} \right\rangle = 0.$$

Since A_i^p is symmetric positive semidefinite, then 0 is an extreme value of $\langle A_i^p \mathbf{e},\ \mathbf{e}\rangle$, which implies that the vector $(\hat{\boldsymbol{\delta}},\ 0)^T$ is an eigenvector of A_i^p with eigenvalue 0. In other words, $(\hat{\boldsymbol{\delta}},\ 0)^T \in \text{Null}(A_i^p)$, which implies that

$$\text{Null}(A_{ff}^{(p)}) = \text{Null}(A_{cf}^{(p)}),$$

which, in turn, implies that

$$\text{Range}\left(A_{ff}^{(p)}\right) = \text{Range}\left(\left(A_{cf}^{(p)}\right)^T\right) = \text{Range}\left(A_{fc}^{(p)}\right),$$

and the lemma is proved. ∎

Rewriting (4.19), we want $\boldsymbol{\delta} \in \mathbb{R}^{n_i}$ such that

$$\boldsymbol{\varepsilon}_1 - \mathbf{q} = A_i^p \boldsymbol{\delta}$$

for some $\mathbf{q} \in \mathcal{Z}$. By the proof of Lemma 4.2, the set of all such $\boldsymbol{\delta}$ is

$$Y^{(p)} := \left\{ \boldsymbol{\delta} \in \mathbb{R}^{n_i} \ : \ \left[A_{ff}^{(p)},\ A_{fc}^{(p)} \right] \boldsymbol{\delta} = \hat{\boldsymbol{\varepsilon}}_1 \right\}.$$

If $Y^{(p)}$ is empty, then the constraint in (4.15) cannot be satisfied and $K_{i,p} = \infty$. In this case, more points must be added to C_i for (4.15) to have a solution. If $Y^{(p)}$ is not empty, then any $\boldsymbol{\delta} \in Y^{(p)}$ can be written as $\boldsymbol{\delta} = \boldsymbol{\delta}^* + \boldsymbol{\gamma}$, where $\boldsymbol{\delta}^*$ is a particular element of $Y^{(p)}$ and $\boldsymbol{\gamma} \in \text{Null}([A_{ff}^{(p)},\ A_{fc}^{(p)}])$. From the proof of Lemma 4.2, we may choose $\boldsymbol{\delta}^* = (\hat{\boldsymbol{\delta}}_1,\ 0)^T$, where $A_{ff}^{(p)} \hat{\boldsymbol{\delta}}_1 = \hat{\boldsymbol{\varepsilon}}_1$. We now show that

$$\boldsymbol{\varepsilon}_1 - \mathbf{q}^* = A_i^p \boldsymbol{\delta}^*$$

yields the unique solution to (4.14) or (4.15).

THEOREM 4.3. *If $\hat{\boldsymbol{\varepsilon}}_1 \notin$ Range $(A_{ff}^{(p)})$, then $K_{i,p} = \infty$. If $\hat{\boldsymbol{\varepsilon}}_1 = A_{ff}^{(p)} \hat{\boldsymbol{\delta}}_1$, then the unique solution of* (4.14) *is given by*

$$\mathbf{q}^* = \begin{pmatrix} 0 \\ -A_{cf}^{(p)} \hat{\boldsymbol{\delta}}_1 \end{pmatrix} \in \mathcal{Z}, \tag{4.20}$$

and $K_{i,p} = \langle \hat{\boldsymbol{\varepsilon}}_1, \hat{\boldsymbol{\delta}}_1 \rangle$ for $p = 1$ or 2.

Proof. The first statement follows from Lemma 4.2. To prove the second, let $\boldsymbol{\delta}^* = (\hat{\boldsymbol{\delta}}_1,\ 0)^T$. Using the substitution

$$\boldsymbol{\varepsilon}_1 - \mathbf{q} = A^{(p)} \boldsymbol{\delta}$$

with $\boldsymbol{\delta} \in Y^{(p)}$, we can rewrite (4.14) as

$$\begin{aligned} \min_{\mathbf{q} \in \mathcal{Z}} \max_{\mathbf{e} \notin \text{Null}(A_i^p)} \frac{\langle (\boldsymbol{\varepsilon}_1 - \mathbf{q})^T \mathbf{e},\ (\boldsymbol{\varepsilon}_1 - \mathbf{q})^T \mathbf{e} \rangle}{\langle A_i^p \mathbf{e},\ \mathbf{e} \rangle} &= \min_{\boldsymbol{\delta} \in Y^{(p)}} \langle A_i^p \boldsymbol{\delta}, \boldsymbol{\delta} \rangle \\ &= \min_{\boldsymbol{\gamma} \in \text{Null}([A_{ff}^{(p)},\ A_{fc}^{(p)}])} \langle A_i^p (\boldsymbol{\delta}^* + \boldsymbol{\gamma}),\ (\boldsymbol{\delta}^* + \boldsymbol{\gamma}) \rangle . \end{aligned} \tag{4.21}$$

Any solution of (4.21) is characterized by $\boldsymbol{\gamma}^* \in \text{Null}([A_{ff}^{(p)},\ A_{fc}^{(p)}])$ such that

$$\langle A_i^p (\boldsymbol{\delta}^* + \boldsymbol{\gamma}^*),\ \boldsymbol{\gamma} \rangle = 0 \qquad \forall\, \boldsymbol{\gamma} \in \text{Null}([A_{ff}^{(p)},\ A_{fc}^{(p)}]); \tag{4.22}$$

that is,

$$A_i^p (\boldsymbol{\delta}^* + \boldsymbol{\gamma}^*) \in \text{Range} \left(\begin{bmatrix} A_{ff}^{(p)} \\ A_{cf}^{(p)} \end{bmatrix} \right). \tag{4.23}$$

But $\boldsymbol{\gamma}^* = 0$ satisfies (4.22) by construction of $\boldsymbol{\delta}^*$, which proves that (4.20) solves (4.14).

To prove uniqueness, suppose there are two such solutions to (4.22), say, $\boldsymbol{\delta}^*$ and $\boldsymbol{\beta}^*$. Then

$$A_i^p (\boldsymbol{\delta}^* - \boldsymbol{\beta}^*) = \begin{bmatrix} A_{ff}^{(p)} \\ A_{cf}^{(p)} \end{bmatrix} \hat{\mathbf{w}}$$

for some $\hat{\mathbf{w}} \in \mathbb{R}^{n_f}$. Since both $\boldsymbol{\delta}^*$ and $\boldsymbol{\beta}^*$ are in $Y^{(p)}$, we have $\hat{\mathbf{w}} \in \text{Null}(A_{ff}^{(p)})$. From Lemma 4.2, we have $\text{Null}(A_{ff}^{(p)}) = \text{Null}(A_{cf}^{(p)})$, which implies that $A_i^p(\boldsymbol{\delta}^* - \boldsymbol{\beta}^*) = 0$ and that $\mathbf{q}^*$ is unique.

Finally, substituting $\boldsymbol{\delta}^*$ into (4.21) yields

$$K_{i,p} = \langle A_i^p \boldsymbol{\delta}^*,\ \boldsymbol{\delta}^* \rangle = \left\langle \hat{\boldsymbol{\varepsilon}}_1,\ \hat{\boldsymbol{\delta}}_1 \right\rangle,$$

which completes the proof. ∎

A practical algorithm for determining Q is as follows:

For $p = 1$, set

$$A_{ff}^{(1)} = A_{ff}, \qquad A_{cf}^{(1)} = A_{cf}.$$

For $p = 2$, set

$$A_{ff}^{(2)} = A_{ff}^2 + A_{fc}A_{cf}, \qquad A_{cf}^{(2)} = A_{cf}A_{ff} + A_{cc}A_{cf}.$$

Perform a QR factorization on $A_{ff}^{(p)}$ using Householder reflections and column pivoting to detect rank deficiency. If

$$A_{ff}^{(p)} \hat{\boldsymbol{\delta}}_1 = \hat{\boldsymbol{\varepsilon}}_1$$

has a solution, then set

$$\mathbf{q}^* = \begin{pmatrix} 0 \\ -A_{cf}^{(p)} \hat{\boldsymbol{\delta}}_1 \end{pmatrix}$$

and $K_{i,p} = \langle \hat{\boldsymbol{\varepsilon}}_1,\ \hat{\boldsymbol{\delta}}_1 \rangle$; otherwise, set $K_{i,p} = \infty$.

4.3. Local-global measure. This subsection shows that if $M_{i,1}$ or $M_{i,2}$ is bounded for every $i \in \mathcal{F}$, then the global measure M_1 is also bounded.

THEOREM 4.4. *Let $p = 1$ or 2 and assume that the local approximation property*

$$M_{i,p}(Q, \mathbf{e}) \le K_{i,p} \quad \forall \mathbf{e} \in \mathbb{R}^n \tag{4.24}$$

holds for some $K_{i,p}$ and all $i \in \mathcal{F}$. Then global approximation property (3.4) *is also satisfied with*

$$K = \max_{\alpha \in \mathcal{T}} \sum_{i \in \mathcal{M}_\alpha \cap \mathcal{F}} K_{i,p} \left\| A_i \right\|^{p-1}. \tag{4.25}$$

Proof. We have

$$\begin{aligned}
\langle (I-Q)\mathbf{e},\ (I-Q)\mathbf{e} \rangle &= \sum_{i \in \mathcal{F}} \left\langle \boldsymbol{\varepsilon}_i \boldsymbol{\varepsilon}_i^T (I-Q)\mathbf{e},\ \boldsymbol{\varepsilon}_i \boldsymbol{\varepsilon}_i^T (I-Q)\mathbf{e} \right\rangle \\
&\le \sum_{i \in \mathcal{F}} K_{i,p} \left\langle A_i^p \mathbf{e},\ \mathbf{e} \right\rangle \\
&\le \sum_{i \in \mathcal{F}} K_{i,p} \left\| A_i \right\|^{p-1} \left\langle A_i \mathbf{e},\ \mathbf{e} \right\rangle \\
&= \sum_{\alpha \in \mathcal{T}} \langle A_\alpha \mathbf{e},\ \mathbf{e} \rangle \sum_{i \in \mathcal{M}_\alpha \cap \mathcal{F}} K_{i,p} \left\| A_i \right\|^{p-1} \\
&\le K \sum_{\alpha \in \mathcal{T}} \langle A_\alpha \mathbf{e},\ \mathbf{e} \rangle \\
&= K \left\langle A\mathbf{e},\ \mathbf{e} \right\rangle. \qquad \blacksquare
\end{aligned}$$

Straightforward application of the above techniques can be used to bound M_2 in terms of $M_{i,2}$. However, the resulting bounds on M_2 can be much larger than the maximum value of $M_{i,2}$. While this may not be sharp, it is simple to construct an example where M_2 is much larger than the largest $M_{i,2}$ and, hence, much larger than M_1. In this case, using M_2 to estimate convergence could lead to the erroneous conclusion that the resulting two-level method is slow to converge.

The local measure bounds, $K_{i,p}$, can be used as a diagnostic tool: Theorem 4.4 shows that they contribute to the bound K used to establish convergence in Theorem 3.2. While neither measure provides a sharp bound when the algorithm exhibits a small convergence factor, they can provide a warning: If $K_{i,p}$ is large for some i, it may be profitable to reexamine the choice of the coarse grid, perhaps adding more grid points to $\mathcal{C}$.

As an alternative to increasing the size of $\mathcal{C}$, we could respond to large values of $K_{i,p}$ locally by increasing the size of the neighborhood. Define the set $\mathcal{N}_i^{(k)}$ of *kth removed neighbors* recursively by letting $\mathcal{N}_i^{(1)} := \mathcal{N}_i$ and

$$\mathcal{N}_i^{(\ell+1)} := \cup_{j \in \mathcal{N}_i^{(\ell)}} \mathcal{N}_j. \tag{4.26}$$

Then interpolation could be allowed from the set $\mathcal{C}_i := \mathcal{N}_i^{(k)} \cap \mathcal{C}$, which are the coarse-grid points connected to point i by a path of length k in the graph of A. While this would yield more accurate interpolation, the complexity of $P^T AP$ would certainly increase.

5. Numerical results. We apply the element interpolation methods numerically to two illustrative examples: a Poisson equation discretized on stretched quadrilaterals and a plane-stress cantilever beam. We compare our numerical results with the *bounds* predicted by our theory and demonstrate the improved robustness of the new methods over AMG.

For each problem, we first present results for AMG to show that our usual approach breaks down. For standard AMG, which we now refer to as AMG1, parameter θ defining the cutoff for strong connections is set to 0.25. The interpolation formula is that found in [18] (see (5.10) and (6.4) there). For the stretched grid problem, two variants of AMG are presented that restore convergence (although these fixes will be shown to be ineffective in the elasticity problem). The first, called AMG2, uses a more restrictive definition of strong connections by taking $\theta = 0.50$. The second variant (AMG3) returns to $\theta = 0.25$ but uses iterative weight interpolation [10].

For comparison, we also include a method presented in [9, 8], since they report results for both anisotropic Poisson problems and 2D elasticity. In those articles two basic methods are described, each with a user-specified parameter θ_1. The one we include (referred to in [8] as method II, with $\theta_1 = 0$, and called the Chang–Wong–Fu (CWF) method here) appears to be the most robust of their methods overall. There are two main differences between our standard AMG algorithm and theirs. The first is that their strong connections are determined by absolute value, while we consider only connections of the "right" sign (i.e., the opposite sign from that of the diagonal entry) to be strong. (As with standard AMG, they take $\theta = 0.25$.) The second difference is the use of a modified interpolation formula. As with standard AMG, interpolation to a point i is derived by writing the corresponding residual equation in terms of the error, making some approximations for those terms defined at points not used in interpolation, and solving for e_i to obtain the weights. Unlike standard AMG, these approximations use weighted averages based on absolute values of the matrix

entries and incorporate some geometric ideas based on the assumption that the size of matrix entries diminishes with the distance between grid points. A seemingly minor modification is that, for weak (small) connections j with no connection to set the interpolation points, the approximation is made that $e_j = e_i$ if $\text{sign}(a_{ij}) = -\text{sign}(a_{ii})$ or $e_j = -e_i$ if $\text{sign}(a_{ij}) = \text{sign}(a_{ii})$. Variants of their methods include modifications to restriction and the definition of the coarse-grid operator. However, the algorithm we test reduces to the standard Galerkin formulation. That is, restriction is defined as the transpose of interpolation, and the coarse-grid operator is taken as the product of restriction, the fine-grid matrix, and interpolation. The coarsening method they describe is simply the Ruge–Stüben two-pass coarsening scheme [18] using the modified definition of strong connections. For testing their method, we use our AMG coarsening code with strong connections determined by absolute value.

The only difference here between AMGe and the AMG variants described above is that we use the element interpolation method in AMGe to construct the interpolation operators. Thus, the coarse grids are selected in the same way that they are in AMG. The possibility of using the AMGe measures to determine coarsening is a topic of current research. Three different definitions are considered for interpolation: AMG, local measure 1 (AMGe1), and local measure 2 (AMGe2).

In the multilevel algorithm, we construct "coarse element stiffness matrices" $A_{c,\alpha}$ as follows:

$$A_{c,\alpha} = P^T A_\alpha P. \tag{5.1}$$

To reduce computational complexity and storage costs, we combine coarse elements that operate on the same points by summing them. That is, we define

$$\mathcal{M}_{c,\alpha} := \left\{ j \ : \ \boldsymbol{\varepsilon}_j^T A_{c,\alpha} \boldsymbol{\varepsilon}_j \neq 0 \right\} \tag{5.2}$$

and, when $\mathcal{M}_{c,\alpha} = \mathcal{M}_{c,\beta}$, we combine $A_{c,\alpha}$ and $A_{c,\beta}$ to form a single coarse-element stiffness matrix.

To conform to the theory, the linear systems for the AMGe tests are scaled so that the diagonal is the identity. That is, we actually solve $\hat{A}\hat{\mathbf{u}} = \hat{\mathbf{f}}$, where $\hat{A} = D^{-1/2} A D^{-1/2}$, $\hat{\mathbf{u}} = D^{1/2}\mathbf{u}$, and $\hat{\mathbf{f}} = D^{-1/2}\mathbf{f}$. Our initial experiments use $V(0,1)$ cycles based on damped Jacobi with step size $s = 1/2$. In the examples below, $\|A\|$ is between 2.5 and 3.0 so that $\frac{1}{\|A\|} \leq s \leq \frac{2}{\|A\|}$. For AMG, we use the original unscaled matrix A.

Equation (3.9) in Theorem 3.2 yields a bound on the convergence factor given by

$$\rho \leq 1 - \frac{4 - \|A\|}{4K}, \tag{5.3}$$

where K is the bound on either M_1 or M_2. As we will see, this bound is very pessimistic. Replacing K from (4.25) by $K_p = \max_i K_{i,p}$ yields a somewhat more realistic but still pessimistic estimate for the convergence factor. These estimates are included in the numerical results below.

5.1. Stretched quadrilateral. Consider the stretched quadrilateral problem introduced in section 2, which consists of a Poisson equation on a rectangular grid discretized with $n_x \times n_y$ bilinear quadrilateral elements. The fine-grid elements have a 10:1 aspect ratio, yielding the stencil in (2.4). The boundary conditions are Dirichlet, which are eliminated from the matrix during discretization.

TABLE 5.1
Two-level and V-cycle asymptotic convergence factors for the stretched quadrilateral problem.

	Two-Level				Multilevel			
Size	AMG1	AMG2	AMG3	CWF	AMG1	AMG2	AMG3	CWF
64×64	0.80	0.12	0.09	0.91	0.81	0.14	0.09	0.91
128×128	0.80	0.12	0.10	0.92	0.81	0.14	0.10	0.91

In Table 5.1, we present results of tests with AMG and its variants for the stretched-grid Poisson problem. Both two-level and V-cycle results are given. In all cases, (1,1) V-cycles are used, with C/F-ordered Gauss–Seidel relaxation. For AMG1, convergence is much worse than the two-level factors of 0.06 and V-cycle factors of 0.10 that we would get on unstretched grids. AMG2 and AMG3 both improve convergence greatly, nearly restoring the results that would be obtained in the uniform case.

The AMG variants each produced a semicoarsened grid for the first coarsening. Away from boundaries, AMG1 used a six-point interpolation stencil of the form

$$P_{\text{AMG}} = \begin{bmatrix} 0.084 & 0.332 & 0.084 \\ & * & \\ 0.084 & 0.332 & 0.084 \end{bmatrix}. \tag{5.4}$$

Both AMG2 and AMG3 produced two-point interpolation away from boundaries, so that weights of 0.5 were obtained for the north and south points. Here, the difference between these two methods lies in the interpolation stencils near boundaries.

In geometric multigrid, this anisotropic situation is often treated by semicoarsening, that is, by choosing coarse-grid points along each vertical line. Interpolation is then performed only in the y direction. The typical interpolation weights used in geometric semicoarsening do not involve corner points, so smaller weights intuitively make more sense here. In fact, it can be shown that, for this problem, smaller corner weights (up to a point) generally produce better two-level results. Since performance is very sensitive to these weights, small changes in the algorithm can affect convergence greatly. For example, in the code AMG1R5 that is widely available to the public, in computing the interpolation operator, weak connections are treated in a manner similar to strong connections. For this problem, this modification gives corner weights of 0.0052 and results in two-level and V-cycle factors of 0.33 and 0.54, respectively.

Poor performance of the CWF method is primarily due to the grid chosen. Using their modified definition of strong connections, all off diagonals in stencil (2.4) are considered strong. This results in standard coarsening (coarsening by a factor of 2 in each grid direction), not semicoarsening. It is well known that such a coarse grid cannot be used effectively for this problem without additional modifications to the algorithm, such as line relaxation. It is interesting to note that, as the grid becomes "unstretched," the connections to the left and right become smaller, and proper semicoarsening results. If semicoarsening were used in conjunction with the CWF interpolation, two-level results of 0.33 would be obtained. This is still far from the results reported for anisotropic problems in [8]. In that paper, however, simple five-point finite difference stencils are used, so that M-matrices result, and connections to the east and west actually do become small as the grid is stretched. AMG and the

variants used here would also have no problem with such discretizations. It should be noted that the coarse grids we obtain may not be those that would be obtained with the algorithm as they have implemented it, and in fact appear to be coarser in most of their test problems, given the larger grid complexities they report. Nevertheless, the grids chosen satisfy the criteria they present.

The point here is not that particular fixes exist for this specific problem. There is no shortage of variations of the AMG algorithm: Each works well for some specific cases but can break down for others. Our goal is instead to obtain robust AMG methods that are much more difficult to break. This goal is the motive for the development of AMGe.

In the AMGe tests, the AMG coarsening algorithm used for all three methods again produces semicoarsened grids, and the interpolation formulas for AMGe1 and AMGe2 are as follows:

$$P_{\mathrm{AMGe1}} = \begin{bmatrix} 0.007 & 0.486 & 0.007 \\ & * & \\ 0.007 & 0.486 & 0.007 \end{bmatrix}, \tag{5.5}$$

$$P_{\mathrm{AMGe2}} = \begin{bmatrix} 0.003 & 0.494 & 0.003 \\ & * & \\ 0.003 & 0.494 & 0.003 \end{bmatrix}. \tag{5.6}$$

The stencils at boundaries are similar. Note that interpolation for these algorithms also involve corner points, since these are considered strong connections, but the associated weights for AMGe1 and AMGe2 are much smaller than for AMG. The large element aspect ratio effectively decouples each vertical line of grid points from the others.

The experimental results for AMG1, AMGe1, and AMGe2 are presented in Table 5.2. Two grid sizes, 64×64 and 128×128, are used. For better comparison with the theory, two changes were made to the solution method. First, we used (1,0) V-cycles, as opposed to the (1,1) V-cycles reported in the previous table. In addition, relaxation has been changed from C/F Gauss–Seidel to a Richardson iteration with a relaxation parameter $\omega = 0.5$. For each grid, we show asymptotic convergence factors for AMG1, AMGe1, and AMGe2. Factors are shown for both two-level and multilevel cases. For the two-level case, we show the bound on the convergence factor corresponding to using (4.25) in Theorem 3.2 for M_1. This is computed using $\|A\| = 2.97$ and $K_1 = 2.68$. As expected, the bound is very pessimistic. We also show the convergence factor (labeled "estimate") that would result from substituting $K_1 = \max_i K_{i,1} = 1.34$ and $K_2 = \max_i K_{i,2} = 2.0$ for (4.25) in Theorem 3.2. This provides a somewhat improved but still very pessimistic value for the convergence factor. This behavior is typical of most multigrid theory, where results often substantially exceed theoretical estimates.

The key observation to be made from the data in Table 5.2 is that both AMGe1 and AMGe2 produce substantial improvement over AMG for stretched quadrilaterals. For this problem, AMG2 and AMG3 described above would produce results similar to AMGe1 and AMGe2. Such techniques, however, tend to be somewhat ad hoc, and are not based on theoretical considerations. As such, we cannot determine in advance whether such treatments will be useful for a given problem. By contrast, we expect AMGe1 and AMGe2 to perform well in more general problems involving high aspect

TABLE 5.2
Asymptotic convergence factors, bound predicted by theory, and "improvement" of observed over predicted for the stretched quadrilateral problem.

	Two-Level			Multilevel		
Size	AMG1	AMGe1	AMGe2	AMG	AMGe1	AMGe2
64×64	0.82	0.27	0.27	0.84	0.32	0.27
128×128	0.82	0.28	0.28	0.84	0.31	0.28
Bound	0.97	0.90	–	–	–	–
Estimate		0.81	0.87	–	–	–

ratios, so they should find wide applicability for problems based on unstructured grids having thin domains or regions.

5.2. Plane-stress cantilever beam. Consider the 2D linear elasticity equations

$$u_{xx} + \frac{1-\nu}{2}u_{yy} + \frac{1+\nu}{2}v_{xy} = f_1,$$
$$v_{yy} + \frac{1-\nu}{2}v_{xx} + \frac{1+\nu}{2}u_{xy} = f_2,$$

where u and v are displacements in the x and y directions, respectively. We take $\nu = 4/7$ for the tests. The problem, depicted in Figure 5.1, has free boundaries, except on the left where $u = v = 0$. We discretize with bilinear finite elements on a uniform rectangular mesh with spacing h in both directions (square elements).

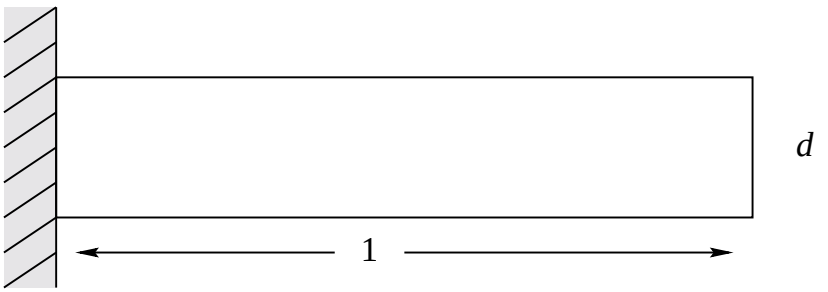


FIG. 5.1. *Plane-stress cantilever beam problem.*

Again, before testing the element-based methods, we present results for the AMG1, AMG2, AMG3, and CWF methods of the previous section. AMG1, AMG2, and AMG3 are applied in a separate fashion (the so-called unknown approach [18]) in which connections between u and v are completely ignored in the determination of strong connections and computation of interpolation weights. Such an approach has been shown to produce good results for the elasticity problem on the unit square when full Dirichlet boundary conditions are used, although it degrades with the number of free sides allowed (cf. [10]). As in [8], the CWF method does not differentiate between the two unknowns. (It should be noted that, in comparison with the AMG1R5 code in that paper, AMG was also applied in a "scalar" fashion. Not unexpectedly, it did not perform well since AMG1R5 was designed for scalar problems and there is really no local relationship between pointwise values of errors in u and errors in v, even when these errors are smooth.)

TABLE 5.3
Two-level and V-cycle asymptotic convergence factors for four methods for the plane-stress problem.

	Two-level				Multilevel			
d	AMG1	AMG2	AMG3	CWF	AMG1	AMG2	AMG3	CWF
1	0.33	0.31	0.31	0.73	0.58	0.62	0.58	0.96
3/4	0.33	0.31	0.29	0.79	0.72	0.65	0.64	0.95
1/2	0.34	0.32	0.29	0.89	0.80	0.76	0.81	0.96
1/4	0.58	0.57	0.51	0.98	0.97	0.95	0.98	0.98
1/8	0.90	0.90	0.87	0.99	0.99	0.98	0.99	0.99
1/16	0.98	0.98	0.98	0.99	0.98	0.98	0.99	0.99
1/32	0.98	0.99	0.98	0.99	0.99	0.99	0.99	0.98
1/64	0.99	0.99	0.99	0.99	0.98	0.98	0.98	0.98

Results are presented in Table 5.3. Several different thicknesses are used for the beam, ranging from a square cross section, $d = 1$, to a very thin beam, $d = 1/64$. As before, (1,1) V-cycles were used with C/F Gauss–Seidel relaxation. The factors shown were obtained after 100 cycles. (With such poor convergence factors, it generally takes many iterations to reach the asymptotic state.) Note that there is little difference between convergence factors for AMG1, AMG2, and AMG3, although AMG3 with iterative weight interpolation shows a slight advantage. For this problem, the $u - u$ and $v - v$ stencils each resemble the anisotropic stencils for the Poisson problem (with the first stretched in x and the second in y), but with grids that are less stretched. Thus, the "fixes" do not improve much on the basic algorithm, as they did for the stretched-grid Poisson problem. Note that two-level factors are nearly constant until d is reduced to below 1/2, at which convergence degrades sharply. V-cycle factors, while somewhat acceptable for large d, degrade even more quickly than two-level factors as the domain becomes thinner. With all three AMG variants, on the finest grid, u is semicoarsened in x while v is semicoarsened in y for all d used. Thus, for all $d > 1/64$, interpolation computed on the finest level does not change. This indicates strongly that the problem is not that interpolation accuracy for smooth components is affected, but that such smooth components must be interpolated *more* accurately as d is reduced. In fact, it can be shown that the energy norm relative to the Euclidean norm of the smoothest components decreases to zero with d, which by standard variational multigrid theory (see section 3) suggests that interpolation must become increasingly more accurate.

For this problem, the interpolation scheme in the CWF method gives very slow convergence. Even two-level factors are large for $d = 1$, indicating that the local interpolation accuracy is not sufficient. Since quite good results for the 2D elasticity problem were presented in [8], some explanation for the behavior of the method here is warranted. As noted earlier, the CWF method does not differentiate between u and v, and, for example, u at a point can interpolate from a combination of u and v values at surrounding points. With such interpolation, however, some minimal conditions must be met before reasonable convergence could be expected. With discretizations of scalar differential operators, away from Dirichlet boundaries, the matrix generally has zero row sums, so that the constant is in the local null space of the operator. Since the smoothest grid functions are locally constant, these must be interpolated exactly, which in turn requires interpolation weights at each point to sum to 1. Similarly, in the elasticity problem, away from fixed boundaries the $u - u$, $u - v$, $v - u$, and $v - v$

TABLE 5.4
Asymptotic convergence factors, bound predicted by theory, and "improvement" of observed over predicted for the plane-stress problem with $h = 1/64$.

	Two-level			Multilevel		
d	GMG	AMGe1	AMGe2	GMG	AMGe1	AMGe2
1	0.45	0.49	0.48	0.49	0.65	0.85
1/4	0.46	0.48	0.47	0.76	0.68	0.90
1/8	0.46	0.47	0.45	0.78	0.64	0.87
1/16	0.46	0.49	0.45	0.77	0.58	0.77
1/32	0.53	0.45	0.50	0.75	0.51	0.56
1/64	0.67	0.39	0.28	0.67	0.39	0.28
Bound	–	0.97	–	–	–	–
Estimate	–	0.87	0.97	–	–	–

connections at each point also sum to zero, so that separate independent constants in u and v are in the local null space of the operator. For these components to be interpolated exactly, the $u - u$ and $v - v$ interpolation weights both must sum to 1, while the $u-v$ and $v-u$ weights both must sum to zero. Loss of any of these conditions can severely degrade convergence. Now, one thing to note is that the tests in [8] use finite differences and full Dirichlet boundary conditions. In the finite difference case, the $u - v$ and $v - u$ connections are smaller relative to the maximum connection, and are considered weak, thus keeping interpolation completely separate for u and v, at least on the finest grid. In the finite element case here, connections from u to v and v to u are considered strong. By itself, this is not bad. In fact, when full Dirichlet conditions are used with the finite element discretization, the CWF method gives a two-level convergence factor of 0.33 for the $d = 1$ problem. Along free boundaries, however, it was found that different geometric assumptions applied to positive and negative cross connections resulted in a loss of one-row sum for the $u-u$ interpolation weights and zero-row sum for the $u - v$ and $v - u$ weights, resulting in the uniformly poor convergence found.

In the AMGe tests, we use the geometric coarsening strategy of doubling the element size in both directions until there is only one element in the y direction, then doubling the element size in the x direction only. For the multilevel results, we coarsen until $h_x = 2h_y$. Such a grid is not admissible with the other AMG algorithms presented, so a direct comparison of the effect of using the different interpolation formulas on the same grid is not practical here. (Actually, a modification of iterative weight interpolation could be used and would yield results comparable with those previously obtained.) However, it is instructive to include results of AMG where linear interpolation is actually used on these uniform grids. This method then becomes a geometric multigrid algorithm, which we call GMG.

Experimental results from three methods are shown in Table 5.4, using (0,1) V-cycles based on Jacobi sweeps with relaxation parameter $\omega = 0.5$. This should be kept in mind when comparing results with the AMG tests of Table 5.3, since (1,1) V-cycles were used there. The theoretical bounds and estimates suggest extremely slow convergence for AMGe1 and AMGe2 (when applied to AMG, they do not indicate that AMG will converge at all). In fact, however, both AMGe1 and AMGe2 achieve substantial improvement, especially for the two-level algorithm, where they greatly exceed predictions. The bound is based on $\|A\| = 2.50$ and $K_1 = 12.25$, while the predictions are based on $\max_i K_{i,1} = 2.84$ and $\max_i K_{i,2} = 8.31$.

Note that the GMG method, which is basically the best that the AMG methods using separate interpolation can aspire to, also shows the same type of degradation as the AMG methods, although less severe. That is, two-level results are stable until d becomes small, then worsen. V-cycle results degrade faster, although they stabilize due to the artificial limitation on the coarsest grid used. This limitation was not present in the AMG tests. In fact, coarser grids could have been used here, which would result in much worse convergence factors if we were to continue with the same interpolation and relaxation schemes. For further coarsening in the x direction, smoothing of pointwise relaxation would suffer as the grid aspect ratios worsen, so "group" relaxation (here equivalent to y-line relaxation) would be needed to maintain efficiency. We did not consider this group relaxation option or its implications on our theory because the focus here is on the interpolation process.

Two observations are significant: The two-level performance of AMGe1 and AMGe2 is generally independent of the beam thickness until $d = h_x$, where even greater improvement occurs; and the multilevel performance of AMGe1 and AMGe2 improves steadily as the beam becomes thinner. This is in direct contrast to the AMG and GMG results obtained, demonstrating the effectiveness of the new interpolation methods.

While this paper concentrates on the effect of the new interpolation method, it should be kept in mind that there are other techniques that may be applied to enhance performance of the algorithm. For instance, the multilevel experiments shown here focused on Jacobi relaxation and a (0,1) V-cycle. The relaxation method and its parameters can be chosen differently. For example, the multilevel AMGe1 case with $d = 1/4$ shows a convergence factor of 0.65 in Table 5.4. A Jacobi (1,1) V-cycle improves this factor to 0.58, while a (1,1) F-cycle (see [14]) attains a convergence factor of 0.31. Nearly identical results, 0.65 for V(0,1), 0.56 for V(1,1), and 0.33 for $F(1,1)$, are obtained if the Jacobi relaxation is replaced by nodal Gauss–Seidel with symmetric CF relaxation, which sweeps over the C points followed by the F points on the downward leg of the V-cycle, and over the F points followed by the C points on the upward leg. Another possibility is the use of a single multigrid V(1,1) cycle as a preconditioner for a conjugate gradient iteration. Applied to the plane-stress problem using the nodal relaxation described above, this yields convergence factors ranging from 0.16 to 0.26 per conjugate gradient iteration.

For both sets of experiments, AMGe interpolation achieves significant improvement over conventional AMG performance. We believe that further improvement is possible using more sophisticated coarse-grid selection. We observe that local measures $M_{i,1}$ and $M_{i,2}$ carry a great deal of information about the nature of the underlying problem and its discretization, and we should be able to exploit this information to determine more effective coarse grids.

6. Conclusions. For any multigrid method to work, errors that remain after relaxation must be well approximated by the range of interpolation. Since algebraic multigrid does not rely on geometric information, its fundamental challenge is to construct coarse grids and interpolation operators that approximate these errors. The core of this challenge is to determine errors that cannot be effectively reduced by local processing.

Two local measures were introduced here to quantify how well the coarsening processes determine algebraically smooth error, and they were used to construct new interpolation operators. Experimental data for two representative test problems confirm that these operators produce an AMGe algorithm whose convergence rates for

these cases are substantially better than standard AMG.

Current research focuses on using these measures in AMGe also to assess the ability of coarse-grid points to represent the necessary error components, that is, to determine *which* points are best suited to be on the coarse grid. Combined with the improved interpolation operator, this may lead to very efficient AMGe algorithms for a much wider range of problems than is currently available.

REFERENCES

[1] R. E. Alcouffe, A. Brandt, J. E. Dendy Jr., and J. W. Painter, *The multi-grid methods for the diffusion equation with strongly discontinuous coefficients*, SIAM J. Sci. Statist. Comput., 2 (1981), pp. 430–454.

[2] C. Baldwin, P. N. Brown, R. D. Falgout, J. Jones, and F. Graziani, *Iterative linear solvers in a 2d radiation-hydrodynamics code: Methods and performance*, J. Comput. Phys., 154 (1999), pp. 1–40.

[3] J. H. Bramble, J. E. Pasciak, J. Wang, and J. Xu, *Convergence estimates for multigrid algorithms without regularity assumptions*, Math. Comp., 57 (1991), pp. 23–45.

[4] A. Brandt, *Algebraic multigrid theory: The symmetric case*, Appl. Math. Comput., 19 (1986), pp. 23–56.

[5] A. Brandt, S. F. McCormick, and J. W. Ruge, *Algebraic Multigrid (AMG) for Automatic Multigrid Solutions with Application to Geodetic Computations*, Tech. report, Institute for Computational Studies, Fort Collins, CO, 1982.

[6] A. Brandt, S. F. McCormick, and J. W. Ruge, *Algebraic multigrid (AMG) for sparse matrix equations*, in Sparsity and Its Applications, D. J. Evans, ed., Cambridge University Press, Cambridge, UK, 1984, pp. 257–284.

[7] W. L. Briggs, V. E. Henson, and S. F. McCormick, *A Multigrid Tutorial*, 2nd ed., SIAM, Philadelphia, 2000.

[8] Q. Chang, Y. S. Wong, and H. Fu, *On the algebraic multigrid method*, J. Comput. Phys., 125 (1996), pp. 279–292.

[9] Q. Chang, Y. S. Wong, and Z. Li, *New interpolation formulas using geometric assumptions in the algebraic multigrid method*, Appl. Math. Comput., 50 (1992), pp. 223–254.

[10] A. J. Cleary, R. D. Falgout, V. E. Henson, J. E. Jones, T. A. Manteuffel, S. F. McCormick, G. N. Miranda, and J. W. Ruge, *Robustness and scalability of algebraic multigrid*, SIAM J. Sci. Comput., 21 (2000), pp. 1886–1908.

[11] W. Z. Huang, *Convergence of algebraic multigrid methods for symmetric positive definite matrices with weak diagonal dominance*, Appl. Math. Comput., 46 (1991), pp. 145–164.

[12] J. Mandel, M. Brezina, and P. Vaněk, *Energy optimization of algebraic multigrid bases*, Computing, 62 (1999), pp. 205–228.

[13] S. F. McCormick, *Multigrid methods for variational problems: Further results*, SIAM J. Numer. Anal., 21 (1984), pp. 255–263.

[14] S. F. McCormick, *Multigrid methods for variational problems: General theory for the V-cycle*, SIAM J. Numer. Anal., 22 (1985), pp. 634–643.

[15] S. F. McCormick and J. W. Ruge, *Multigrid methods for variational problems*, SIAM J. Numer. Anal., 19 (1982), pp. 924–929.

[16] J. Ruge, *Element interpolation for algebraic multigrid (AMG)*, presented at the 4th Copper Mountain Conference on Multigrid Methods, Copper Mountain, CO, 1989, unpublished.

[17] J. W. Ruge and K. Stüben, *Efficient solution of finite difference and finite element equations by algebraic multigrid (AMG)*, in Multigrid Methods for Integral and Differential Equations, D. J. Paddon and H. Holstein, eds., Oxford University Press, New York, 1985, pp. 169–212.

[18] J. W. Ruge and K. Stüben, *Algebraic multigrid*, in Multigrid Methods, S. F. McCormick, ed., SIAM, Philadelphia, 1987, pp. 73–130.

[19] K. Stüben, *Algebraic multigrid (AMG): Experiences and comparisons*, Appl. Math. Comput., 13 (1983), pp. 419–452.

[20] K. Stüben, U. Trottenberg, and K. Witsch, *Software development based on multigrid techniques*, in Proceedings of the IFIP Conference on PDE Software, Modules, Interfaces and Systems, B. Enquist and T. Smedsaas, eds., Söderköping, Sweden, 1983, North-Holland, Amsterdam, 1984, pp. 241–267.

[21] P. Vaněk, M. Brezina, and J. Mandel, *Convergence analysis of algebraic multigrid based on smoothed aggregation*, Numer. Math., to appear.

[22] P. Vaněk, M. Brezina, and R. Tezaur, *Two-grid method for linear elasticity on unstructured meshes*, SIAM J. Sci. Comput., 21 (1999), pp. 900–923.

[23] W. L. Wan, *An Energy-Minimizing Interpolation for Multigrid Methods*, UCLA CAM Report 97-18, Department of Mathematics, UCLA, Los Angeles, CA, 1997.

[24] W. L. Wan, T. F. Chan, and B. Smith, *An Energy-Minimizing Interpolation for Robust Multigrid Methods*, UCLA CAM Report 98-6, Department of Mathematics, UCLA, Los Angeles, CA, 1998.

SIAM J. SCI. COMPUT.
Vol. 22, No. 5, pp. 1593–1609
© 2000 Society for Industrial and Applied Mathematics

COLLOCATION METHODS FOR THE COMPUTATION OF PERIODIC SOLUTIONS OF DELAY DIFFERENTIAL EQUATIONS*

K. ENGELBORGHS†, T. LUZYANINA†‡, K. J. IN 'T HOUT§, AND D. ROOSE†

Abstract. In this paper we investigate collocation methods for the computation of periodic solutions of autonomous delay differential equations (DDEs). Periodic solutions are found by solving a periodic two-point boundary value problem, which is an infinite-dimensional problem for DDEs, in contrast to the case of ordinary differential equations. We investigate three collocation methods based on piecewise polynomials. We discuss computational issues and show numerical orders of convergence using an extensive number of tests. We compare our numerical results with known theoretical convergence results for initial value problems for DDEs. In particular, we show how superconvergence at the mesh points can be lost or recovered depending on the DDE model under consideration and on the choice of collocation discretization. We end with a brief discussion of adaptive mesh selection.

Key words. delay differential equations, periodic solutions, collocation methods

AMS subject classifications. 65J15, 65P05

PII. S1064827599363381

1. Introduction.

This paper deals with collocation methods for the computation of periodic solutions to autonomous *delay differential equations* (DDEs),

$$\frac{\mathrm{d}x(t)}{\mathrm{d}t} = f(x(t), x(t-\tau)), \tag{1.1}$$

with one (constant) delay $\tau > 0$ and $f : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^n$. For the sake of convenience we restrict ourselves to one-delay DDEs, but generalization to multiple delays is straightforward.

Let $C := C([-\tau, 0], \mathbb{R}^n)$ be the vector space of continuous functions mapping the interval $[-\tau, 0]$ into $\mathbb{R}^n$. For $s \in \mathbb{R}$, denote by $x_s \in C$ the segment of a solution x to (1.1) defined by

$$x_s(\theta) = x(s+\theta), \ \ \theta \in [-\tau, 0].$$

C is the state space for (1.1), that is, for any given s the solution segment x_s uniquely determines, via (1.1), the values $x(t)$ for all $t \geq s$. Consequently, any periodic solution

*Received by the editors November 5, 1999; accepted for publication (in revised form) July 11, 2000; published electronically November 17, 2000. This research presents results of research project TO 98/16 and fellowship F/98/68, funded by the Research Council of Katholieke Universiteit Leuven, of research project G.0270.00 funded by the Fund for Scientific Research–Flanders (Belgium) and of research project IUAP P4/02 funded by the program on Interuniversity Poles of Attraction, initiated by the Belgian State, Prime Minister's Office for Science, Technology and Culture. The scientific responsibility is assumed by its authors.

http://www.siam.org/journals/sisc/22-5/36338.html

†Computer Science Department, Katholieke Universiteit Leuven, Celestijnenlaan 200A, B-3001 Heverlee, Belgium (koen.engelborghs@cs.kuleuven.ac.be, tatyana.luzyanina@cs.kuleuven.ac.be, dirk.roose@cs.kuleuven.ac.be). The first author is a Research Assistant of the Fund for Scientific Research–Flanders.

‡Institute of Mathematical Problems in Biology, Russian Academy of Sciences, Pushchino, Moscow region, 142292, Russia.

§Mathematical Institute, Leiden University, P.O.Box 9512, 2300 RA Leiden, The Netherlands (hout@math.leidenuniv.nl). The research of the third author was supported by the Netherlands Organisation for Scientific Research.

to (1.1) can be found as the solution of the following *two-point boundary value problem* (BVP),

$$\left\{\begin{array}{l} \frac{\mathrm{d}x(t)}{\mathrm{d}t} = f(x(t), x(t-\tau)),\ t \in [0,T], \\ x_0 = x_T, \\ p(x,T) = 0, \end{array}\right. \tag{1.2}$$

where T denotes the (unknown) period and $p(x,T) = 0$ represents a suitable phase condition to remove translational invariancy.

Observe that the boundary condition $x_0 = x_T$ in (1.2) concerns a condition in an *infinite-dimensional* space, namely, C. This is clearly in contrast with the analogous problem for the case of ordinary differential equations (ODEs, i.e., (1.1) without a delay argument $x(t-\tau)$), where the boundary condition applies in a *finite-dimensional* space, $\mathbb{R}^n$.

In [22] the convergence of numerical simulations to stable periodic solutions was considered. In [27] a shooting approach is used, based on a combination of [16] and [26], for solving problem (1.2). This approach has been used to study a number of specific applications of DDEs [14, 13] and has been shown to be reasonably efficient.

The approach we follow in this paper is to solve the BVP (1.2) by the well-known collocation method. Collocation methods are popular methods for the numerical solution of BVPs for systems of ODEs. The widely used COLSYS/COLNEW [1, 6] and AUTO [12] software packages employ collocation with piecewise polynomials to approximate the solutions to BVPs for ODEs.

In this paper, our main aim is to study, by means of numerical experiments, a number of related collocation schemes for (1.2) with respect to their (observed) convergence orders and efficiency. In section 2 we distinguish between two kinds of BVPs for DDEs and discuss theoretical convergence results from the literature. In section 3 we outline the general setting relevant to collocation methods. In section 4 we review known convergence results concerning the numerical solution of IVPs for DDEs, which we will compare with the outcomes of our numerical experiments for the periodic BVP (1.2). This comparison is motivated by the fact that in the case of ODEs the same convergence orders are valid for similar discretizations of IVPs and BVPs, respectively. Next, in section 5 we explain the collocation variants that we investigate. In section 6 we describe the test models which we use, and we present our actual numerical results in section 7. Section 8 contains conclusions.

2. Two kinds of BVPs. A general two-point BVP for delay differential equations (1.1) may be formulated as (cf., e.g., [18, 24])

$$\left\{\begin{array}{l} \frac{\mathrm{d}x(t)}{\mathrm{d}t} = f(x(t), x(t-\tau)),\ t \in [0,T], \\ b(x_0, x_T) = 0, \end{array}\right. \tag{2.1}$$

where $b : C \times C \to C$ and $T > 0$ are given. However, in the current literature on the numerical solution of BVPs for DDEs (or more general functional differential equations), see, e.g., [5, 8, 3, 9, 30, 25] and [2, section 11.7], one invariably encounters BVPs for (1.1) of the form

$$\left\{\begin{array}{l} \frac{\mathrm{d}x(t)}{\mathrm{d}t} = f(x(t), x(t-\tau)),\ t \in [0,T], \\ x(\theta) = \phi(\theta),\ \theta \in [-\tau, 0), \\ b(x(0), x(T)) = 0. \end{array}\right. \tag{2.2}$$

Here, an initial function $\phi \in C$ is given, a discontinuity in the solution at $t = 0$ is allowed, and $b : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^n$ specifies a boundary condition in the finite-dimensional space $\mathbb{R}^n$. We note that both kinds of BVPs, (2.1) and (2.2), may contain additional conditions and corresponding free parameters (cf. $p(x, T) = 0$ and T in (1.2)). In [24] problem (2.1) is called Halanay's boundary value problem, whereas (2.2) is called a BVP with finite defect.

The existing literature on the numerical solution of BVPs for DDEs seems to be mainly targeted towards problems of the kind (2.2), which arise, among others, in control applications. In particular, a convergence result for piecewise polynomial collocation for the problem (2.2) is proven in [4]. We are not aware of theoretical convergence results on the numerical solution of problems of the kind (2.1) which cannot be reduced to the BVP (2.2) (and are not based on approximations by truncated Fourier series as in [11, 10, 33]). Note that, in general, periodic solutions of (1.1) cannot be found as solutions to BVPs of the form (2.2).

3. Preliminaries. Below, we give our basic notations and the general setting relevant to collocation methods.

Let Π be a *mesh*, i.e., a collection of *mesh points* $0 = t_0 < t_1 < \cdots < t_L = T$ which partition the interval $[0, T]$. Set $h_i := t_{i+1} - t_i$ for $i = 0, \ldots, L-1$. Denote by π_m the set of all (vector-valued) polynomials of degree not exceeding m. We will approximate a solution x to (1.1) on the interval $[0, T]$ by an element from the following space of piecewise polynomials:

$$S_m(\Pi) := \{u \in C([0, T], \mathbb{R}^n) : u|_{[t_i, t_{i+1}]} \in \pi_m,\ i = 0, \ldots, L-1\}.$$

Clearly, $\dim S_m(\Pi) = n \times (L \times m + 1)$. Note that $u \in S_m(\Pi)$ is continuous but not necessarily continuously differentiable (at the mesh points).

Let

$$X(\Pi) := \bigcup_{i=0}^{L-1} X_i$$

with

$$X_i := \{c_{i,l} := t_i + c_l h_i,\ l = 1, \ldots, m\}$$

be a given set of (so-called) *collocation points* in $[0, T]$ based on the given, fixed set of *collocation parameters* $\{c_l\}$ with $0 \le c_1 < c_2 < \cdots < c_m \le 1$. Then the idea of a *collocation method* for approximating a solution to the DDE (1.1) is to find a function $u : [-\tau, T] \to \mathbb{R}^n$, the so-called *collocation solution*, such that (a) its restriction to $[0, T]$ belongs to $S_m(\Pi)$, (b) for $i = 0, \ldots, L-1$ its restriction to $[t_i, t_{i+1}]$ satisfies system (1.1) on the (finite) set X_i, and (c) it fulfills the relevant initial or boundary value condition.

For $i = 0, \ldots, L-1$ the collocation solution can be represented on subinterval $[t_i, t_{i+1}]$ as

$$u(t) = \sum_{j=0}^{m} u(t_{i+\frac{j}{m}}) P_{i,j}(t), \tag{3.1}$$

where

$$P_{i,j}(t) = \prod_{r=0, r\neq j}^{m} \frac{t - t_{i+\frac{r}{m}}}{t_{i+\frac{j}{m}} - t_{i+\frac{r}{m}}},\ j = 0, \ldots, m \tag{3.2}$$

are Lagrange polynomials and

$$t_{i+\frac{j}{m}} := t_i + \frac{j}{m}h_i,\ j = 1, \ldots, m-1$$

are so-called *representation points* (between the mesh points $t_{i+\frac{0}{m}} = t_i$ and $t_{i+\frac{m}{m}} = t_{i+1}$). Thus the collocation solution u is completely determined on $[0, T]$ by the vectors $u_{i+\frac{j}{m}} := u(t_{i+\frac{j}{m}})$, $i = 0, \ldots, L-1$, $j = 0, \ldots, m-1$, and $u_L = u(t_L)$.

4. Initial value problems. In this section we consider the initial value problem (IVP)

$$\begin{cases} \frac{\mathrm{d}x(t)}{\mathrm{d}t} = f(x(t), x(t-\tau)),\ t \in [0, T], \\ x_0 = \phi, \end{cases} \tag{4.1}$$

where $\phi \in C$ is a given initial function.

Even if both f and ϕ are arbitrarily smooth, a solution x to (4.1) may have a discontinuity in its first derivative at $t = 0$. This is because, in general,

$$\left.\frac{\mathrm{d}\phi(\theta)}{\mathrm{d}\theta}\right|_{\theta=0^-} \neq f(\phi(0), \phi(-\tau)).$$

Due to the presence of the delayed term $x(t-\tau)$ in (4.1), the discontinuity at $t = 0$ is propagated forward in time. However, the discontinuity is smoothed as time increases, and a discontinuity in the $(k+1)$st derivative of x, in general, appears at $t = k\tau$, $k = 0, 1, 2, \ldots$. In a numerical procedure for IVPs (4.1) it is important to include some of these so-called breaking points $t = k\tau$, $k = 0, 1, 2, \ldots$, in the mesh to avoid a reduction in the order of accuracy.

The theory concerning the numerical solution of IVPs for DDEs is well developed. Below, several main results on the convergence behavior of collocation methods in the case of the IVP (4.1) are collected. In particular, these deal also with collocation methods that are based on the well-known *Gauss–Legendre* collocation parameters $\{c_l\}$, i.e., the roots of the mth degree Gauss–Legendre polynomial transformed to $[0, 1]$.

The collocation method as outlined in section 3 consists of solving successively for $i = 0, \ldots, L-1$ the following system of equations for u restricted to $[t_i, t_{i+1}]$:

$$\frac{\mathrm{d}u(c_{i,l})}{\mathrm{d}t} = \begin{cases} f(u(c_{i,l}), \phi(c_{i,l}-\tau)) \text{ when } c_{i,l} - \tau \leq 0, \\ f(u(c_{i,l}), u(c_{i,l}-\tau)) \text{ when } c_{i,l} - \tau > 0 \end{cases} \tag{4.2}$$

for $l = 1, \ldots, m$, where $u(0) = \phi(0)$. The derivative of u in (4.2) has to be interpreted such that if $c_{i,l} = t_i$ it denotes the right-hand derivative at t_i, whereas if $c_{i,l} = t_{i+1}$, it stands for the left-hand derivative at t_{i+1}.

For ease of presentation, we assume in the rest of this section that all breaking points in $[0, T]$ are included in the mesh Π and, further, that f, ϕ are both arbitrarily smooth. We set $h := \max_i(h_i)$ and let $\|\cdot\|$ denote a given, fixed norm on $\mathbb{R}^n$.

The first theorem is a standard convergence result.

THEOREM 4.1. *Let x be the exact solution to* (4.1), *and let u be a corresponding collocation solution as described in section* 3 *and above. Then the following estimate holds:*

$$\max_{t\in[0,T]} \|u(t) - x(t)\| = \mathcal{O}(h^m) \quad (h \downarrow 0). \tag{4.3}$$

If $\{c_l\}$ are the Gauss–Legendre collocation parameters, then the above estimate can be sharpened to

$$\max_{t\in[0,T]} \|u(t) - x(t)\| = \mathcal{O}(h^{m+1}) \quad (h \downarrow 0). \tag{4.4}$$

The following result deals with the special case where the mesh Π is such that for each $i < L$ with $t_i \geq \tau$ the subinterval $[t_i, t_{i+1}]$ is mapped by $t \mapsto t - \tau$ exactly onto a (previous) subinterval $[t_k, t_{k+1}]$ with $0 \leq k < i$. A mesh with this property is called a *constrained mesh*.

THEOREM 4.2. *Let x be the exact solution to* (4.1)*, and let u be a corresponding collocation solution, as described in section* 3 *and above, based on a constrained mesh Π. If $\{c_l\}$ are the Gauss–Legendre collocation parameters, then the following estimate is valid:*

$$\max_{i=0,\ldots,L} \|u(t_i) - x(t_i)\| = \mathcal{O}(h^{2m}) \quad (h \downarrow 0). \tag{4.5}$$

The above result can be found in [7, 17]. It means that *superconvergence* at the mesh points, a well-known phenomenon in the case of both IVPs and BVPs for ODEs, also occurs in the case of IVPs for DDEs under an appropriate choice of Π.

The result of Theorem 4.2 can be obtained for more general meshes Π when one considers application of alternative interpolation procedures in the past, instead of using the relevant local collocation polynomial. We state convergence theorems for two alternative types of interpolation procedures, where we leave the detailed (interpolation) formulas to section 5.1. We assume that the ratios h_{i+1}/h_i are uniformly bounded from below and above by fixed positive real numbers. We further assume that, near a breaking point, the support points for the two interpolation procedures below are all taken on one side of the breaking point.

Let $q \geq 1$ be a given fixed integer. The first theorem below deals with the case of polynomial interpolation with respect to the mesh points for approximating the delayed term $x(t - \tau)$ (cf. (5.9)).

THEOREM 4.3. *Let x be the exact solution to* (4.1)*, and let u be a corresponding collocation solution, where in* (4.2) *the quantity $u(c_{i,l} - \tau)$ is replaced by the value of the polynomial of degree q at $t = c_{i,l} - \tau$ that interpolates u at $q + 1$ mesh points t_j in the neighborhood of t. If $\{c_l\}$ are the Gauss–Legendre collocation parameters, then the following estimates hold:*

$$\max_{i=0,\ldots,L} \|u(t_i) - x(t_i)\| = \mathcal{O}(h^{\min(2m,q+1)}) \quad (h \downarrow 0), \tag{4.6}$$

$$\max_{t\in[0,T]} \|u(t) - x(t)\| = \mathcal{O}(h^{\min(m+1,q+1)}) \quad (h \downarrow 0). \tag{4.7}$$

Theorem 4.3 follows from common estimation arguments. If the above polynomial interpolation procedure is used with $q = 2m-1$, superconvergence at the mesh points is obtained for (in essence) arbitrary meshes.

The next theorem deals with the case of *equistage* interpolation [20, 21] for approximating the delayed term (cf. (5.10)). At present, a satisfactory convergence analysis for this procedure, relevant to arbitrary meshes, is not known. A first result was obtained in [21].

THEOREM 4.4. *Let x be the exact solution to* (4.1)*, and let u be a corresponding collocation solution, where in* (4.2) *the quantity $u(c_{i,l} - \tau)$ is replaced by the value of*

the equistage polynomial of degree q at $t = c_{i,l} - \tau$ that interpolates u at $q+1$ points $c_{j,l}$ in the neighborhood of t. If $\{c_l\}$ are the Gauss–Legendre collocation parameters, then (4.7) *holds, and* (4.6) *holds under a model assumption on* Π *(see* [21]*).*

Result (4.6) follows from a direct application of the convergence theorem proved in [21]. The result concerning (4.7) can be shown using well-known arguments.

We remark that if the mesh is constrained, then the collocation process described in Theorem 4.4 reduces to the original collocation process, as in this case equistage interpolation is not actually performed. Thus, in this case Theorems 4.1, 4.2 apply.

5. Computation of periodic solutions. In this section we explain in detail the collocation variants that we consider. We also discuss the structure of the linear system which needs to be solved in each Newton iteration and describe adaptive mesh selection.

5.1. Outline of the collocation variants. For our numerical method, it is convenient to consider instead of BVP (1.2) a transformed BVP that is obtained from (1.2) after scaling time by the factor T^{-1},

$$\left\{ \begin{array}{l} \frac{\mathrm{d}x(t)}{\mathrm{d}t} = Tf(x(t), x(t - \frac{\tau}{T})), \ \ t \in [0,1], \\ x_0 = x_1, \\ p(x,T) = 0. \end{array} \right. \tag{5.1}$$

Recall that T is unknown and will be determined during computations. In the following we assume that $\tau < T$. We set $\bar{\tau} = \tau T^{-1}$. Note that a periodic solution to (1.1) is arbitrarily smooth for all times (if f is arbitrarily smooth). Therefore no time points have to be included in the mesh a priori; cf. section 4.

A collocation solution u to the transformed BVP (5.1), corresponding to a mesh Π on $[0,1]$ and collocation parameters $\{c_l\}$, is determined (cf. section 3) in terms of the unknowns

$$u_{i+\frac{j}{m}} = u(t_{i+\frac{j}{m}}), \ i = 0, \ldots, L-1, \ j = 0, \ldots, m-1, \text{ and } u_L = u(t_L)$$

by the *collocation equations*

$$\frac{\mathrm{d}u(c_{i,l})}{\mathrm{d}t} = \left\{ \begin{array}{l} Tf(u(c_{i,l}), u(c_{i,l} - \bar{\tau} + 1)) \text{ when } c_{i,l} - \bar{\tau} < 0, \\ Tf(u(c_{i,l}), u(c_{i,l} - \bar{\tau})) \text{ when } c_{i,l} - \bar{\tau} \geq 0 \end{array} \right. \tag{5.2}$$

for $i = 0, \ldots, L-1$, $l = 1, \ldots, m$. Here, the same convention on the derivative of u holds as in the case of (4.2).

Clearly, for BVP (1.2) the initial function, x_0, is not known beforehand. However, the periodic boundary condition is linear, and in constructing (5.2) we have used this fact in order to eliminate directly the numerical unknowns corresponding to times $t < 0$. We did not eliminate u_0 (because its value is used to determine u in $[0, t_1]$), so we still need to require

$$u_0 = u_L, \tag{5.3}$$

together with the phase condition

$$p(u,T) = 0. \tag{5.4}$$

In order to solve the combined nonlinear system (5.2), (5.3), (5.4), we apply Newton iteration. Write $c = c_{i,l}$ and

$$\tilde{c} = \left\{ \begin{array}{ll} c - \bar{\tau} + 1 & \text{when } c - \bar{\tau} < 0, \\ c - \bar{\tau} & \text{when } c - \bar{\tau} \geq 0. \end{array} \right. \tag{5.5}$$

Let integer k be such that $t_k \le \tilde{c} < t_{k+1}$. Then (5.2) assumes the form

$$\sum_{j=0}^{m} u_{i+\frac{j}{m}} P'_{i,j}(c) = Tf\left(\sum_{j=0}^{m} u_{i+\frac{j}{m}} P_{i,j}(c), \sum_{j=0}^{m} u_{k+\frac{j}{m}} P_{k,j}(\tilde{c})\right), \tag{5.6}$$

where $P'_{i,j}$ denotes the derivative of $P_{i,j}$. After linearization of (5.6) with respect to u, T we obtain

$$\begin{aligned} &\textstyle\sum_{j=0}^{m} P'_{i,j}(c)\Delta u_{i+\frac{j}{m}} - f(u(c),u(\tilde{c}))\Delta T - TA_0(u(c),u(\tilde{c}))\sum_{j=0}^{m} P_{i,j}(c)\Delta u_{i+\frac{j}{m}} \\ &-TA_1(u(c),u(\tilde{c}))\left(\textstyle\sum_{j=0}^{m} P_{k,j}(\tilde{c})\Delta u_{k+\frac{j}{m}} + \left(\sum_{j=0}^{m} u_{k+\frac{j}{m}} P'_{k,j}(\tilde{c})\right)\frac{\tau}{T^2}\Delta T\right) \\ &\qquad = -\left(\textstyle\sum_{j=0}^{m} u_{i+\frac{j}{m}} P'_{i,j}(c) - Tf(u(c),u(\tilde{c}))\right), \end{aligned} \tag{5.7}$$

where

$$u(c) = \textstyle\sum_{j=0}^{m} u_{i+\frac{j}{m}} P_{i,j}(c),\ u(\tilde{c}) = \sum_{j=0}^{m} u_{k+\frac{j}{m}} P_{k,j}(\tilde{c})$$

and

$$A_0(\xi,\eta) = \frac{\partial f}{\partial \xi}(\xi,\eta),\ A_1(\xi,\eta) = \frac{\partial f}{\partial \eta}(\xi,\eta).$$

Note that (5.6) is, through $\tilde{c}$, in general nondifferentiable with respect to T whenever $t_k = \tilde{c}$. We have (arbitrarily) chosen the right-hand derivative when this occurs. The nondifferentiability can reduce the asymptotic quadratic convergence of the Newton iteration. However, since periodic solutions have continuous derivatives, we did not expect nor encountered problems in our test cases.

We also deal with two variants to the above collocation scheme. Here, the value of the collocation solution at $\tilde{c}$ ("in the past"),

$$u(\tilde{c}) = \sum_{j=0}^{m} u_{k+\frac{j}{m}} P_{k,j}(\tilde{c}), \tag{5.8}$$

is replaced, in (5.6), by the value of an interpolating polynomial of (fixed) degree $q = k_1 + k_2$ constructed through $q+1$ mesh points in the neighborhood of $\tilde{c}$,

$$u(\tilde{c}) \to \sum_{j=k-k_1}^{k+k_2} u(t_j) Q_{k,j}(\tilde{c}), \tag{5.9}$$

or, by the value of an interpolating polynomial defined according to equistage interpolation (see [20], [21]),

$$u(\tilde{c}) \to \sum_{j=k-k_1}^{k+k_2} u(c_{j,l}) R_{k,j,l}(\tilde{c}), \tag{5.10}$$

where

$$Q_{k,j}(t) = \prod_{r=k-k_1, r\neq j}^{k+k_2} \frac{t-t_r}{t_j - t_r}, \qquad R_{k,j,l}(t) = \prod_{r=k-k_1, r\neq j}^{k+k_2} \frac{t-c_{r,l}}{c_{j,l} - c_{r,l}}. \tag{5.11}$$

Integers k_1, k_2 are chosen such that the points t_j and $c_{j,l}$ lie centered around the value $\tilde{c} \in [t_k, t_{k+1}]$. Remark that, if the mesh points or collocation points used in (5.9), (5.10), (5.11) have indices outside $0,\dots,L$, they are appropriately substituted using "periodic continuation" of Π and u (as in (5.5)).

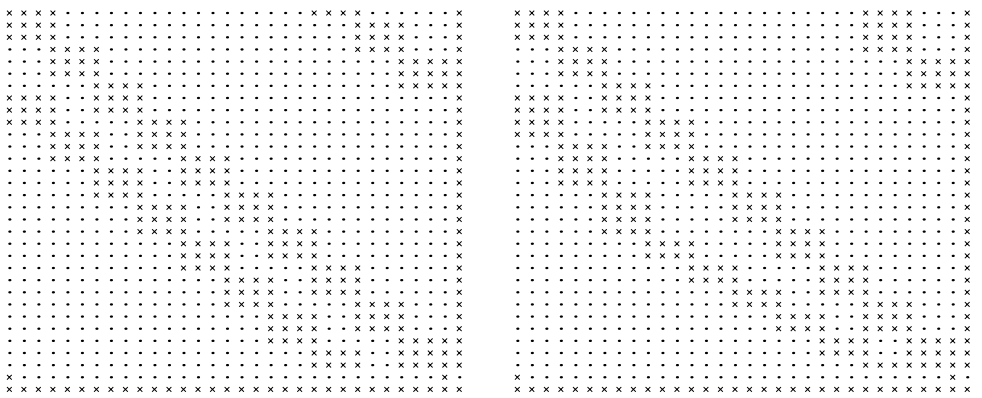


FIG. 1. *Structure of the matrix arising in the Newton iteration in the case of method* (5.3), (5.4), (5.6) *and model* (6.3) *using a uniform mesh (left) and a nonuniform mesh (right) with* $L = 10$ *and collocation polynomials of degree* $m = 3$ *(*$n = 1$*,* $\tau = 1$*,* $T \approx 4.0964$*). Zero elements are indicated by dots* $(\cdot)$ *and nonzero elements by crosses* $(\times)$*.*

5.2. Linear system structure. Figure 1 visualizes the structure of two matrices corresponding to two cases of the system of linear equations (5.7), (5.3), (5.4), which defines the Newton iteration. These matrices contain a (large) $mLn \times (mL+1)n$ block filled with two (circular) bands. This block is bordered by one column and $n+1$ rows. The extra column contains derivatives with respect to the period; n extra rows contain condition (5.3), and one extra row arises because of the phase condition (5.4).

The diagonal band is itself a concatenation of $mn \times (m+1)n$ blocks. The off-diagonal band is a consequence of the delay term, its circularity is due to the (eliminated) periodicity condition. When the mesh is equidistant (uniform), the off-diagonal band lies at a fixed distance from the diagonal band. This distance is approximately equal to $\bar{\tau}$ of the total matrix size. When the mesh is nonuniform, this distance varies (see Figure 1 (right)).

In the case of ODEs, the linear system can be solved with a band solver. Then the band size is proportional to the system size n and the (polynomial) degree m but it does not depend on the number of intervals L. For DDEs this is no longer possible. For moderate values of m, n, and L the linear system can be solved with a direct method as we did in our tests. Efficiency could be increased using, e.g., a chord-Newton method, in which case the Jacobian is not recomputed (and factored) in every iteration but remains fixed during a number of iterations.

Figure 2 shows the structure of the matrices corresponding to the linearized system of equations for the collocation variants defined by the replacements (5.9) and (5.10), respectively. As can be seen, both variants yield an increase in the size of the bandwidth of the off-diagonal band, approximately by a factor q.

We conclude this subsection by noticing that the technique of condensation of parameters (cf. [2]) cannot be applied in our case except for the collocation variant based on (5.9) because its interpolation procedure in the past uses approximations only at the mesh points.

5.3. Adaptive mesh selection. Adaptive mesh selection for BVPs for ODEs is achieved by equidistributing the integral

$$K = \int_0^1 |x^{(m+1)}(Ts)|^{\frac{1}{m+1}} ds$$

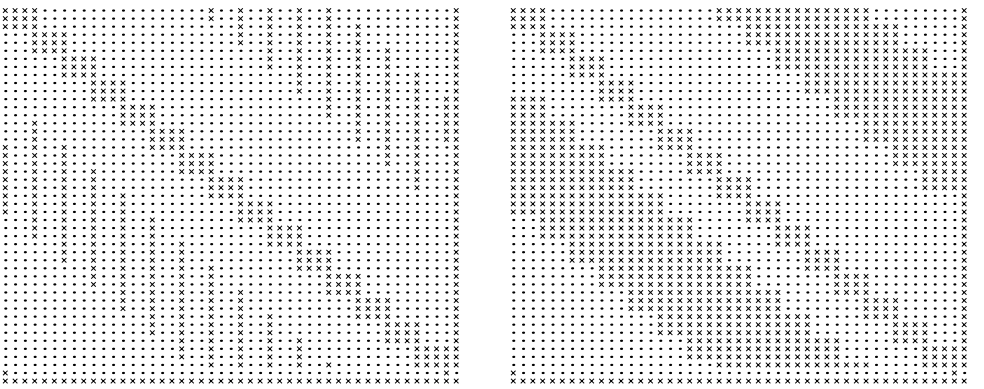


FIG. 2. *Structure of the matrices arising in the Newton iteration in the case of collocation variants based on* (5.9) *(left) and* (5.10) *(right) for model* (6.1) *using a uniform mesh with* $L = 15$*, collocation polynomials of degree* $m = 3$*, and* $q = 5$ *(*$n = 1$*,* $\tau = 2$*,* $T \approx 5.3918$*). Zero elements are indicated by dots* $(\cdot)$*, and nonzero elements by crosses* $(\times)$*.*

over the mesh intervals

$$\int_{t_i}^{t_{i+1}} |x^{(m+1)}(Ts)|^{\frac{1}{m+1}} ds = \frac{1}{L} K, \ i = 0, \ldots, L-1, \tag{5.12}$$

where $x(t)$ is scalar and $|\cdot|$ is the absolute value; see, e.g., [12, 2].

The new mesh points, t_i, satisfying (5.12) are determined using an approximation of $x^{(m+1)}$ obtained from the current mesh and collocation solution u.

Formula (5.12) is based on the error estimate

$$|x(tT) - u(t)| = C h_i^{m+1} \max_{s \in [t_i, t_{i+1}]} |x^{(m+1)}(Ts)| + \mathcal{O}(h^{m+2}), \ t \in [t_i, t_{i+1}], \tag{5.13}$$

when using Gauss–Legendre collocation points. Formula (5.13) is proven for BVPs for ODEs using a Green's function [31]. The use of a Green's function can be extended to BVPs of finite defect (2.2) (see [5] which includes a discussion of its changed smoothness properties) but the approach is not applicable to the "infinite-dimensional" BVP (2.1). Hence the validity of (5.13) remains an important open question in our situation.

Our numerical tests (see section 7.2), however, indicate that formula (5.13) remains valid for the periodic BVP (1.2). We subsequently implemented and tested the strategy (5.12) as implemented in AUTO on the models considered in this paper. These tests resulted in a desirable decrease in error for difficult profiles (compared to the use of a uniform mesh with the same number of mesh points) but in an increase in error for rather easy profiles (i.e., profiles without steep gradients). The latter phenomenon (which did not disappear even for very fine meshes) was, after a number of tests, removed by changing the discretization formula used for (5.12) in AUTO to the trapezium rule. The scheme then showed both a (small) decrease in error for easy profiles and an (even larger than before) decrease in error for difficult profiles. We conclude with the formulae for the latter variant.

Based on the old mesh points, t_i, and the piecewise constant $u^{(m)}$, let

$$d_i = \frac{u^{(m)}(\xi_{i+1}) - u^{(m)}(\xi_i)}{\frac{1}{2}(t_{i+1} - t_{i-1})}, \ i = 0, \ldots, L,$$

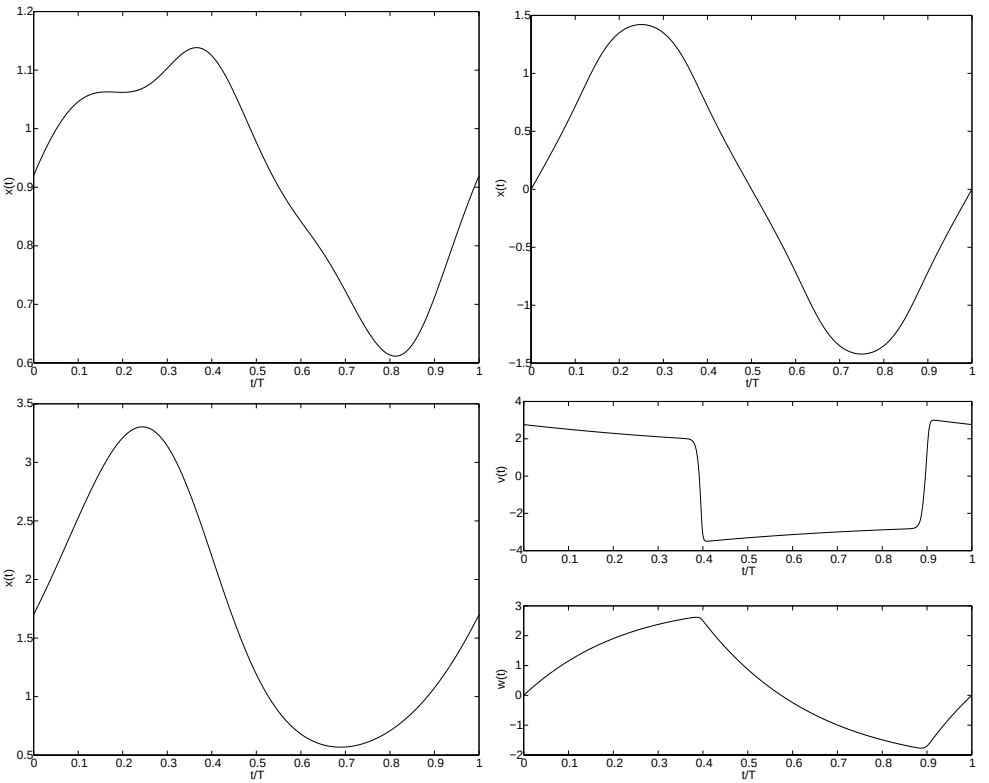


FIG. 3. *The periodic solutions under consideration: model* 1 *(upper left),* 2 *(upper right),* 3 *(lower left), and* 4 *(lower right).*

where $\xi_i \in (t_{i-1}, t_i)$, $\xi_{i+1} \in (t_i, t_{i+1})$ are chosen arbitrarily and where "periodic continuation" is used for d_0 and d_L. We approximate

$$S(t) = \int_0^t |x^{(m+1)}(Ts)|^{\frac{1}{m+1}} ds$$

by $\tilde{S}(t)$ at the old mesh points using $\tilde{S}(t_0) = 0$ and

$$\tilde{S}(t_{i+1}) = \tilde{S}(t_i) + (t_{i+1} - t_i)\frac{|d_i|^{\frac{1}{m+1}} + |d_{i+1}|^{\frac{1}{m+1}}}{2},\ i = 1, \ldots, L-1,$$

and assume $\tilde{S}$ is linear between mesh points. If u is not scalar, $\tilde{S}$ is computed for each component as above and then summed over all components. The new t_i's are now easily selected from the monotonically increasing $\tilde{S}$ by linear interpolation such that $t_0 = 0$ and $\tilde{S}(t_{i+1}) - \tilde{S}(t_i) = \Delta\tilde{S} = \frac{\tilde{S}(1)}{L},\ i = 1, \ldots, L-1$.

6. Test models. In this section we formulate the models we considered to test the collocation methods described in section 5. The periodic solution profiles (scaled to the interval $[0, 1]$) are displayed in Figure 3.

Model 1. The following equation is a model for the regeneration of white blood cells [15, 19] and is usually referred to as the Mackey–Glass equation,

$$\frac{dx}{dt} = ax(t) + b\frac{x(t-\tau)}{1 + x^c(t-\tau)}, \tag{6.1}$$

where a, b, c, τ are given parameters. For $a = -1$, $b = 1.74$, $c = 10$, $\tau = 2$ we computed an unstable periodic solution with period $T \approx 5.3918$.

Model 2. The scalar DDE

$$\frac{dx}{dt} = \alpha x(t-1)\frac{1+x^2(t-1)}{1+x^4(t-1)} \tag{6.2}$$

was studied analytically (see, e.g., [23, 28, 32]) and numerically [16, 11, 27]. It was proven that at $\alpha = \pi/2$ a family of periodic solutions bifurcates from the zero solution and that stable periodic solutions of period $T = 4$ exist whenever $\alpha > \pi/2$. Here we consider such a stable periodic solution at $\alpha = 2$.

Model 3. This is the well-known delayed logistic equation

$$\frac{dx}{dt} = (\lambda - x(t-1))x(t); \tag{6.3}$$

cf., e.g., [18, 22]. We computed a stable periodic solution at $\lambda = 1.7$ with $T \approx 4.0964$.

Model 4. The following system models recurrent neural feedback [29, 10],

$$\begin{cases} \frac{dv}{dt} = h(v(t)) - w(t) + I(v(t-\tau)), \\ \frac{dw}{dt} = \rho\,(v(t) + a - bw(t)) \end{cases} \tag{6.4}$$

with

$$\begin{cases} h(v) = v - v^3/3, \\ I(v) = \mu\,(v - v_0), \\ v_0 = \text{a real root of } h(v) - (v+a)/b. \end{cases}$$

This system exhibits a stable spiked periodic solution for parameter values $a = 0.7$, $b = 0.8$, $\rho = 0.08$, $\tau = 25$, $v_0 \approx -1.1994$, and $\mu = -2$ with $T \approx 50.7326$.

7. Results. In this section we discuss our numerical results. For a given mesh Π we compute (approximations of) the *continuous error* E_C and the *discrete error* E_I, defined by

$$E_C = \max_{t\in[0,1]} \|x(Tt) - u(t)\|, \; E_I = \max_{i=0,\ldots,L} \|x(Tt_i) - u(t_i)\|,$$

where x denotes the exact periodic solution, u denotes a collocation solution, and $\|\cdot\|$ is the Euclidean norm. To this purpose, we first computed a reference solution for x, using $L = 1000$ subintervals. Next, we compared the reference solution to u (for $L = 10, \ldots, 200$) at a very fine mesh in order to compute E_C, and at (only) the mesh points in order to compute E_I. As a phase condition, we fixed the value of one component of the solution at $t_0 = 0$. We show observed orders of convergence and the results of adaptive mesh selection.

7.1. Numerical order results. In this subsection the mesh points are always taken equidistant, i.e., $h_i = h$ for $i = 0, \ldots, L-1$.

First we concentrate on the collocation method (5.3), (5.4), (5.6). Figure 4 shows the evolution of E_C and E_I for collocation solutions with $m = 4$ as h goes to zero using equidistant collocation points (collocation parameters $\{c_l\} = \{\frac{1}{8}, \frac{3}{8}, \frac{5}{8}, \frac{7}{8}\}$) (left) and Gauss–Legendre collocation points (right). We have approximated the slope of the graph of the continuous error E_C by means of a least squares fit on (all) the results for $L = 100, \ldots, 200$. Table 1 summarizes the obtained convergence orders

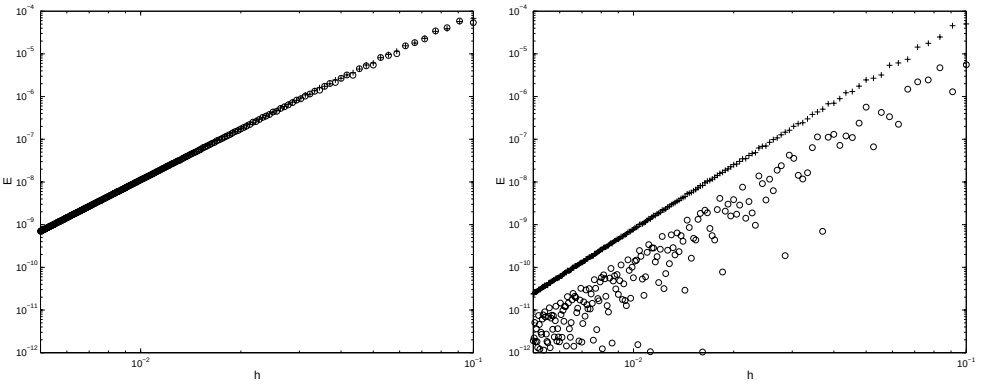


FIG. 4. *Evolution of $E_C(+)$ and E_I ($\circ$) in the case of Model* 1 *and the collocation method* (5.3), (5.4), (5.6) *with $m = 4$ using equidistant (left) and Gauss–Legendre (right) collocation points.*

TABLE 1

Approximations to the orders of convergence of E_C in the case of Models 1–3 *and the collocation method* (5.3), (5.4), (5.6) *for $L = 100, \ldots, 200$ using equidistant (left) and Gauss–Legendre (right) collocation points.*

m	Model 1	Model 2	Model 3	m	Model 1	Model 2	Model 3
2	2.016	2.042	2.004	2	2.992	3.004	3.003
3	4.008	4.009	3.980	3	4.003	3.999	4.001
4	4.003	4.017	4.033	4	5.003	4.990	4.927

of E_C for $m = 2, 3, 4$. Clearly, an $\mathcal{O}(h^m)$ versus $\mathcal{O}(h^{m+1})$ convergence behavior is apparent. For odd m, a numerical convergence behavior of $\mathcal{O}(h^{m+1})$ is also found in the case of equidistant collocation points. This feature seems to be a consequence of the specific choice of the collocation parameters and it disappears if they are not chosen symmetrically with respect to 1/2. The table clearly shows agreement with the result of Theorem 4.1 for IVPs for DDEs (and with results for BVPs of the form (2.2), see [5, 8]).

Superconvergence at mesh points is not present in the E_I curve of Figure 4 (right). According to Theorem 4.2 it is recovered in the case of IVPs for DDEs for constrained meshes. The special situation that mesh points get mapped into mesh points (and collocation points into collocation points) by the delay depends, for BVPs, both on the model and the mesh [2]. More specifically, it occurs when $k\tau = lT$ with k and l integer and (equidistant) mesh points with L a multiple of k. This is the case for Model 2 where $4\tau = T$ whenever L is a multiple of 4. During computations, this relation holds only approximately as T is computed with finite accuracy. Nevertheless we observe numerical superconvergence; see Figure 5 (left). We further observe that, when the period and the delay are not commensurate, E_I is generally smaller whenever L is such that $\frac{\tau}{T}L$ is near an integer number (that is, whenever L is a multiple of k and $\frac{\tau}{T} \approx \frac{l}{k}$ for some k, l); see Figure 5 (right).

We now consider the two variants of the collocation method (5.3), (5.4), (5.6) where, instead of using the collocation solution in the past, the interpolation formula (5.9) or (5.10) is applied. Figure 6 shows the evolution of E_C and E_I when $u(\tilde{c})$ is replaced by the value of an interpolating polynomial of degree $q = 2m - 1$ through $q + 1$ mesh points t_j according to (5.9) (left), or through $q + 1$ collocation points $c_{j,l}$ according to (5.10) (right). The obtained convergence orders for E_I in the case

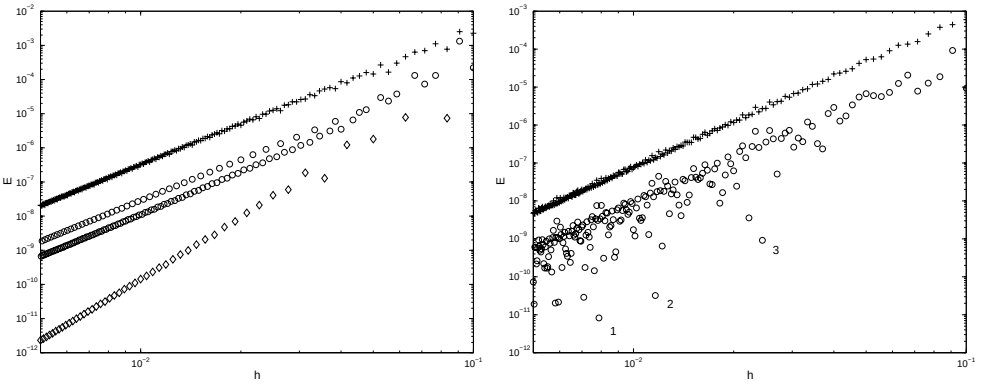


FIG. 5. *Evolution of $E_C(+)$ and E_I ($\circ$ and $\diamond$) for Models* 2 *(left) and* 3 *(right) based on the collocation method* (5.3), (5.4), (5.6) *with $m = 3$ and Gauss–Legendre collocation points. Left: superconvergence of E_I when L is a multiple of* 4 *(E_I indicated as a $\diamond$). Right: greater accuracy for E_I when $\frac{\tau}{T}L \approx 31.003$ (1), $\frac{\tau}{T}L \approx 20.994$ (2), $\frac{\tau}{T}L \approx 10.009$ (3).*

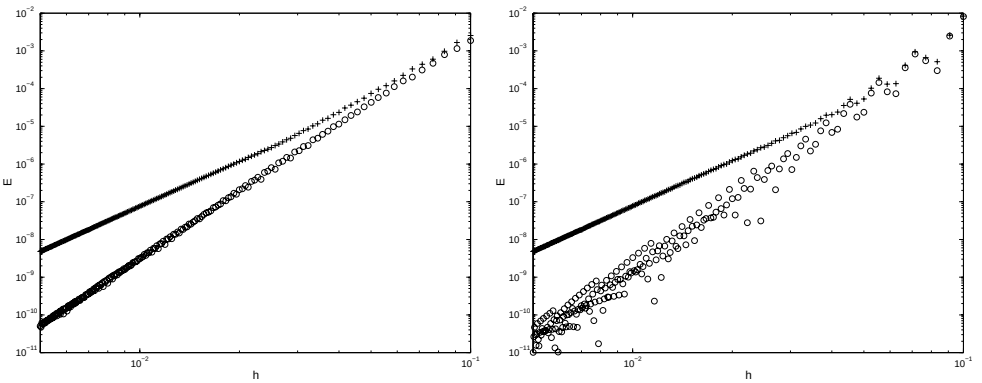


FIG. 6. *Evolution of $E_C(+)$ and E_I ($\circ$) for Model* 3 *based on collocation variants* (5.9) *(left) and* (5.10) *(right) with $m = 3$, $q = 2m - 1$, and Gauss–Legendre collocation points.*

of the variant based on (5.9) are summarized in Table 2. In both cases, we observe superconvergence behavior for E_I, which is in agreement with Theorems 4.3, 4.4. Further, if h is sufficiently small, both variants have essentially the same continuous error E_C as the original collocation method; cf. Figures 5 (right) and 6.

Though, for E_I, $\mathcal{O}(h^{2m})$ is the maximum order of convergence, we observed that, for the collocation variants, E_I can be further decreased for small h by choosing $q > 2m - 1$. This is illustrated in Figure 7 (compare left to right for points where L is not a multiple of 4).

7.2. Adaptive mesh selection. When the mesh is uniform, formula (5.13) implies that the error is locally proportional to the $(m+1)$-st derivative of the solution profile. This correspondence is clearly visible in Figure 8 and was observed in several other tests as well.

On profiles without steep gradients (Figure 3, Models 1–3) little benefit can be expected from an adapted mesh in comparison with a uniform mesh. However, as mentioned in section 5.3, AUTO's adaptive mesh selection can, at times, increase the

TABLE 2
Approximation to the orders of convergence of E_I for Models 1-3 *using collocation solutions based on collocation variant* (5.9) *with* $L = 50, \ldots, 100$ *and Gauss–Legendre collocation points.*

m	Model 1	Model 2	Model 3
2	4.009	3.991	4.020
3	5.900	5.736	5.980
4	7.776	7.379	7.814

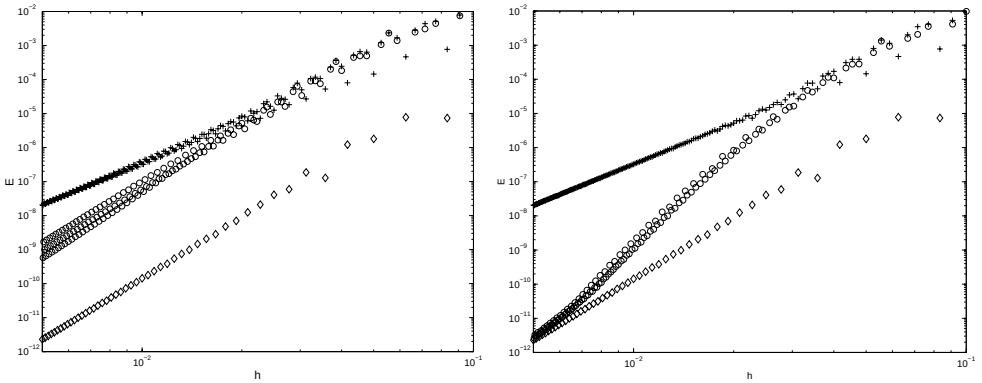


FIG. 7. *Evolution of $E_C(+)$ and E_I ($\diamond$ when L is a multiple of* 4, $\circ$ *otherwise*) *for Model* 2 *based on collocation variant* (5.10) *with* $m = 3$, $q = 2m - 1$ *(left), respectively,* $q = 2m + 3$ *(right) and Gauss–Legendre collocation points.*

error for the case of DDEs. This unfavorable feature does not disappear as h goes to zero; see Figure 9 (top). Instead, it was (almost completely) removed by slightly changing the scheme as described in section 5.3. We might also note that we did not observe this feature in a small number of tests on periodic solutions of ODEs.

The periodic solution of model 4 exhibits steep gradients and we successfully used adaptive mesh selection to decrease the errors significantly. Here too, the variant described in section 5.3 gives better results (see Figure 9 (bottom)). The evolution, for this example, of both E_C and E_I versus L is shown in Figure 10. Note that an increase of more than three orders of accuracy is gained by adapting the mesh compared to the accuracy using a uniform mesh with the same number of intervals and thus the same number of unknowns in the Newton procedure.

As a last remark, we note that when adaptive mesh selection is applied in the case of Model 4 and the collocation variants based on (5.9) or (5.10), errors decrease compared to uniform meshes but not as much as in Figure 10. Also, no superconvergence was visible within the range of intervals computed.

We conclude that our numerical results indicate for the collocation method (5.3), (5.4), (5.6) the validity of formula (5.13) and approach (5.12) combined with the need for a (small) change in their discretization used in a practical implementation.

8. Conclusions. Collocation, based on piecewise polynomials, is considerably successful in the numerical solution of (periodic) BVPs for systems of ODEs. It is, e.g., used in the software packages COLSYS/COLNEW [1, 6] and AUTO [12]. In this paper we investigated the use of collocation methods for computing periodic solutions to DDEs.

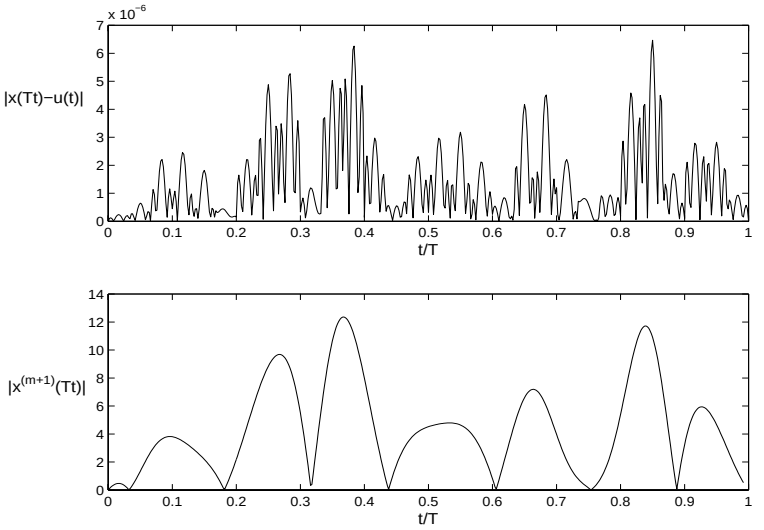


FIG. 8. *Correspondence of the error* $|x(Tt) - u(t)|$ *and the* $(m+1)$*th derivative,* $|x^{(m+1)}(Tt)|$*, of a collocation solution for Model* 1 *using* $m = 3$*,* $L = 30$*, a uniform mesh, and Gauss–Legendre collocation points.*

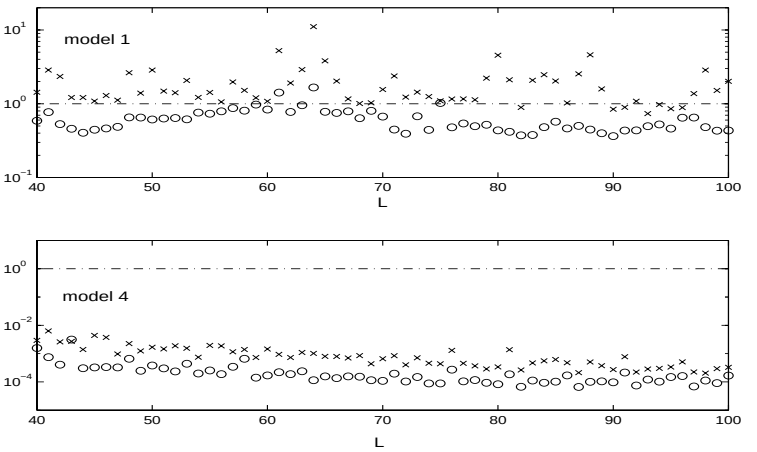


FIG. 9. *Ratio* $E_C^{\text{adapted}}(L)/E_C^{\text{uniform}}(L)$ *versus* L *using AUTO's adaptive mesh selection* $(\times)$*, respectively the variant described in section* 5.3*,* $(\circ)$ *for* $L = 40, \ldots, 100$*. Here,* $m = 3$ *and Gauss–Legendre collocation points are used to compute the collocation solutions for Model* 1 *(upper part) and Model* 4 *(lower part).*

We studied three collocation methods, two of which are based on alternative interpolation procedures for approximating the delayed term. The presence of the delayed term influences the structure of the linear system to be solved in the Newton iteration and thus affects the efficiency of the collocation method. Based on an extensive set of numerical tests we obtained information on the order of convergence and its dependence on the collocation discretization. We show how these results correspond with known convergence results for IVPs for DDEs. In particular, we show when superconvergence at mesh points is lost and/or recovered. We end with a brief discussion of adaptive mesh selection.

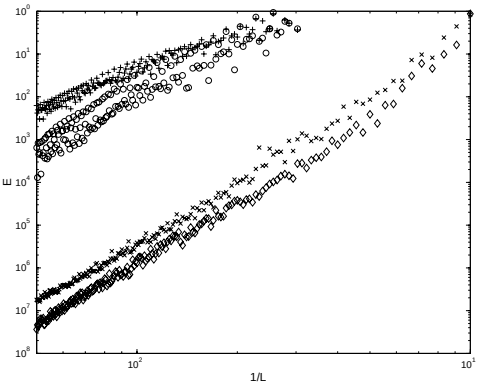


FIG. 10. *Evolution of* E_C $(+, \times)$ *and* $E_I(\circ, \diamond)$ *for Model* 4 *using equidistant* $(+, \circ)$*, respectively, adapted* $(\times, \diamond)$ *meshes with* $m = 3$ *and Gauss–Legendre collocation points.*

REFERENCES

[1] U. M. ASCHER, J. CHRISTIANSEN, AND R. D. RUSSELL, *Collocation software for boundary value ODEs*, ACM Trans. Math. Software, 7 (1981), pp. 209–222.

[2] U. M. ASCHER, R. M. M. MATTHEIJ, AND R. D. RUSSELL, *Numerical Solution of Boundary Value Problems for Ordinary Differential Equations*, Prentice-Hall, Englewood Cliffs, NJ, 1988.

[3] N. V. AZBELEV, V. P. MAKSIMOV, AND L. F. RAKHMATULLINA, *Introduction to the Theory of Functional Differential Equations*, Nauka, Moscow, 1991 (in Russian).

[4] G. BADER, *Numerische Behandlung von Randwertproblemen für Funktionaldifferentialgleichungen*, Technical Report 227, Universität Heidelberg, Heidelberg, Germany, 1983.

[5] G. BADER, *Solving boundary value problems for functional differential equations by collocation*, in Numerical Boundary Value ODEs, U. M. Ascher and R. D. Russell, eds., Progr. Sci. Comput. 5, Birkhäuser, Boston, 1985, pp. 227–243.

[6] G. BADER AND U. ASCHER, *A new basis implementation for a mixed order boundary value ODE solver*, SIAM J. Sci. Statist. Comput., 8 (1987), pp. 483–500.

[7] A. BELLEN, *One-step collocation for delay differential equations*, J. Comput. Appl. Math., 10 (1984), pp. 275–283.

[8] A. BELLEN, *A Runge-Kutta-Nystrom method for delay differential equations*, in Numerical Boundary Value ODEs, U. M. Ascher and R. D. Russell, eds., Progr. Sci. Comput. 5, Birkhäuser, Boston, 1985, pp. 271–283.

[9] A. BELLEN AND M. ZENNARO, *A collocation method for boundary value problems of differential equations with functional arguments*, Computing, 32 (1984), pp. 307–318.

[10] A. M. CASTELFRANCO AND H. W. STECH, *Periodic solutions in a model of recurrent neural feedback*, SIAM J. Appl. Math., 47 (1987), pp. 573–588.

[11] E. J. DOEDEL AND P. P. C. LEUNG, *A numerical technique for bifurcation problems in delay differential equations*, Congr. Numer., 34 (1982), pp. 225–237.

[12] E. J. DOEDEL, X. J. WANG, AND T. F. FAIRGRIEVE, *AUTO*94: *Software for continuation and bifurcation problems in ordinary differential equations*, Technical Report CRPC-95-2, Center for Research on Parallel Computing, California Institute of Technology, Pasadena, CA, 1995.

[13] K. ENGELBORGHS, V. LEMAIRE, J. BÉLAIR, AND D. ROOSE, *Numerical bifurcation analysis of delay differential equations arising from physiology modeling*, J. Math. Biol., to appear.

[14] K. ENGELBORGHS, D. ROOSE, AND T. LUZYANINA, *Bifurcation analysis of periodic solutions of neutral functional differential equations: A case study*, Internat. J. Bifur. Chaos Appl. Sci. Engrg., 8 (1998), pp. 1889–1905.

[15] L. GLASS AND M. C. MACKEY, *Oscillation and chaos in physiological control systems*, Science, 197 (1977), pp. 287–289.

[16] K. P. HADELER, *Effective computation of periodic orbits and bifurcation diagrams in delay equations*, Numer. Math., 34 (1980), pp. 457–467.

[17] E. Hairer, S. P. Nørsett, and G. Wanner, *Solving Ordinary Differential Equations*. I. *Nonstiff Problems*, 2nd ed., Springer Ser. Comput. Math. 8, Springer-Verlag, New York, 2nd ed., 1993.

[18] J. K. Hale, *Theory of Functional Differential Equations*, Appl. Math. Sci., Springer-Verlag, New York, 1977.

[19] J. K. Hale and N. Sternberg, *Onset of chaos in differential delay equations*, J. Comput. Phys., 77 (1988), pp. 221–239.

[20] K. J. in ´t Hout, *A new interpolation procedure for adapting Runge-Kutta methods to delay differential equations*, BIT, 32 (1992), pp. 634–649.

[21] K. J. in ´t Hout, *Convergence of Runge–Kutta methods for delay differential equations*, BIT, submitted.

[22] K. J. in ´t Hout and Ch. Lubich, *Periodic orbits of delay differential equations under discretization*, BIT, 38 (1998), pp. 72–91.

[23] J. L. Kaplan and J. A. Yorke, *Ordinary differential equations which yield periodic solutions of differential delay equations*, J. Math. Anal. Appl., 48 (1974), pp. 317–324.

[24] V. B. Kolmanovskii and A. Myshkis, *Introduction to the Theory and Application of Functional Differential Equations*, Math. Appl. 463, Kluwer Academic Publishers, Dordrecht, The Netherlands, 1999.

[25] X. Liu, *Periodic boundary value problems for differential equations with finite delay*, Dynam. Systems Appl., 3 (1994), pp. 357–368.

[26] K. Lust, D. Roose, A. Spence, and A. R. Champneys, *An adaptive Newton–Picard algorithm with subspace iteration for computing periodic solutions*, SIAM J. Sci. Comput., 19 (1998), pp. 1188–1209.

[27] T. Luzyanina, K. Engelborghs, K. Lust, and D. Roose, *Computation, continuation and bifurcation analysis of periodic solutions of delay differential equations*, Internat. J. Bifur. Chaos Appl. Sci. Engrg., 7 (1997), pp. 2547–2560.

[28] R. D. Nussbaum, *Periodic solutions of nonlinear autonomous functional differential equations*, in Functional Differential Equations and Approximation of Fixed Points, H.-O. Peitgen and H.-O. Walther, eds., Lecture Notes in Math. 730, Springer-Verlag, New York, 1978, pp. 283–325.

[29] R. E. Plant, *A FitzHugh differential-difference equation modeling recurrent neural feedback*, SIAM J. Appl. Math., 40 (1981), pp. 150–162.

[30] G. W. Reddien and C. C. Travis, *Approximation methods for boundary value problems of differential equations with functional arguments*, J. Math. Anal. Appl., 46 (1974), pp. 62–74.

[31] R. D. Russell and J. Christiansen, *Adaptive mesh selection strategies for solving boundary value problems*, SIAM J. Numer. Anal., 15 (1978), pp. 59–80.

[32] H.-O. Walther, *A theorem on the amplitudes of periodic solutions of differential delay equations with application to bifurcation*, J. Differential Equations, 39 (1978), pp. 369–404.

[33] A. Zapp, *Verzweigungseigenschaften bei nichtlinearen Differential gleichungen mit einer Zeitverzögerung*, Diplomarbeit, University of Cologne, Cologne, Germany, 1997.

SIAM J. SCI. COMPUT.
Vol. 22, No. 5, pp. 1610–1629
© 2000 Society for Industrial and Applied Mathematics

INDEX CONCEPTS FOR LINEAR MIXED SYSTEMS OF DIFFERENTIAL-ALGEBRAIC AND HYPERBOLIC-TYPE EQUATIONS*

MICHAEL GÜNTHER† AND YVONNE WAGNER‡

Abstract. For many technical systems, the use of a refined network modeling approach leads to hyperbolic-type initial-boundary value problems of partial differential-algebraic equations (PDAEs). The boundary conditions of these systems are governed by time-dependent differential-algebraic equations (DAEs) that couple the PDAE system with the network elements that are modeled by DAEs in time only. In order to classify these systems, we extend some index notions that have already been introduced to treat parabolic-type problems. A perturbation index is considered that reflects the sensitivity of the mixed system to slight perturbations in the right-hand side of the PDAEs, as well as in the input signals of the DAE systems and initial values. In order to make an a posteriori analysis of semidiscretization in space and time, we introduce additionally a space and a method of lines (MOL) index. Here one is especially interested in whether the semidiscretized systems properly reflect the properties of the underlying systems. We will show that these indices may detect an artificial sensitivity with respect to perturbations, e.g., if the semidiscretization does not consider the information transport along the characteristics.

Key words. refined network models, coupled systems of partial differential-algebraic equations and differential-algebraic equations, perturbation, space and method of line index, semidiscretization with respect to space and time

AMS subject classifications. 65L05, 35L99

PII. S1064827598349057

1. Introduction.

In technical simulation of time-dependent processes today's industrial software is based on a network approach [14]. In the case of so-called *compact* network elements only topology, and no spatial dimension, is considered; the network equations are given by systems of differential-algebraic equations (DAEs). However, if coupling and second order effects become more important, the spatial distribution can no longer be neglected; the network approach has to be extended to cope with so-called *distributed* elements that are mainly described by partial differential equations (PDEs) in space and time. For example, chemical engineering plants are simulated which consist of both compact and distributed network elements such as heat exchangers [23]. Parameter models can lead to PDEs with DAE systems as boundary conditions [13]. In electric network analysis, the network equations describing the compact network elements such as capacitors and resistors are completed by the telegrapher's equations for transmission lines to model the interconnections more accurately [8]. The use of elastic connections leads to a flexible part in the network equations of multibody system dynamics [22]. Whereas chambers in gas flow networks are described by DAE and ODE models, the connecting pipes are governed by the Euler equations [19].

In many of the applications mentioned above one has to deal with hyperbolic-type initial-boundary value problems of partial differential-algebraic equations (PDAEs). The boundary conditions of these systems are governed by time-dependent DAEs that couple the PDAE system with the network DAEs for the compact elements.

*Received by the editors December 11, 1998; accepted for publication (in revised form) August 8, 2000; published electronically December 5, 2000.
http://www.siam.org/journals/sisc/22-5/34905.html
†Universität (TH) Karlsruhe, Center for Scientific Computing and Mathematical Modelling (IWRMM), Engesserstr. 6, D-76128 Karlsruhe, Germany (guenther@iwrmm.math.uni-karlsruhe.de).
‡Goethestr. 2, D-83435 Bad Reichenhall, Germany (yvonne_wagner@lycos.de).

The aim of this paper is to classify such mixed systems by generalizing the index concept for DAE systems. The DAE index concept has turned out to give insight into the solution properties, as well as into the numerical problems to be expected when solving these systems, for example, how to obtain consistent initial data if there are hidden constraints [20]. For this purpose, Martinson and Barton [17] have generalized the differential time index to PDAE systems that are regarded as abstract Cauchy problems in any direction in the independent variable space. Generalizations of index concepts to linear PDAE systems have recently been proposed by Campbell and Marzalek [6], and Lucht, Strehmel, and Eichler-Liebenow [16], with a main focus on parabolic systems. Though these concepts do not take care of general boundary conditions that arise in our case, some of them can be adapted to the hyperbolic-type case with DAE boundary conditions.

There are different properties that are demanded (or wished) from an index: First, it would be interesting to know some structural properties of the system before its numerical solution, for example, the sensitivity of the analytical solution to small perturbations in the initial data and/or input signals. Second, estimates are required for the impact of semidiscretization in space or time on the index of the semidiscretized system: Does the semidiscretized system properly reflect the behavior of the underlying mixed system, for example, the information transport along the characteristics? Or do we detect an artificial smoothing or coarsening effect?

The paper is organized as follows. In section 2 we specify the system to be examined. We will restrict our investigations to linear hyperbolic-type PDAEs of first order in one space dimension. The boundary conditions are generally given as differential-algebraic equations and may be coupled with network equations for the compact elements.

In section 3 we will define appropriate index concepts, adjusting and extending the concepts introduced in [6, 16] to our hyperbolic case. In order to cope with the effects of semidiscretization in space and time, different indices have been proposed. The spatial index is transferred from the linear parabolic case; by a Laplace transformation the time variable is eliminated and a DAE-BVP is obtained. The time index which is based on a Fourier expansion of the solution is replaced by a MOL index derived by an upwind semidiscretization of the space variable using the characteristic formulation. Finally, a perturbation index is considered which reflects the sensitivity of the mixed system with respect to perturbations in the right-hand side of the PDAE equations, in the initial and boundary data. Energy estimates are used to obtain results for the perturbation index.

Section 4 treats two benchmark problems in detail: a PDAE model for a heat exchanger, and a mixed model for a single transmission line with voltage source and capacitor as terminating elements. We discuss the impact of semidiscretization in time or space on the index of the resulting approximate DAE (ADAE) system. The first example shows that the application of different Rothe methods, where — in contrast to the classical method of lines — semidiscretization is made with respect to time to obtain a boundary value problem in space alone, may lead to different indices for the semidiscretized system, higher than the perturbation index. For the latter example the perturbation and MOL index will allow insights into whether the properties of the system are correctly reflected by the ADAE system: if the information in the characteristic formulation is not adequately used, the semidiscretized system may show a spurious sensitivity with respect to perturbations.

2. Linear mixed systems of DAEs and hyperbolic-type PDAEs. In the following we will consider hyperbolic-type initial-boundary value problems of PDAEs often occurring in a refined network approach. The boundary conditions are governed by time-dependent DAEs that may couple the PDAE system with the network DAEs for those compact elements linked with the PDAE system.

We motivate different levels of abstraction by two examples. The first comes from interconnected electrical network simulation, the second is a mathematical model for a countercurrent heat exchanger.

The signal propagation in a transmission line can be characterized by the telegrapher's equations

$$Au_t + Bu_x + Cu = 0 \quad \text{with} \quad A = \begin{bmatrix} 0 & L \\ C & 0 \end{bmatrix},\ B = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix},\ C = \begin{bmatrix} 0 & R \\ G & 0 \end{bmatrix} \tag{2.1}$$

with $u(x,t) = [U(x,t), I(x,t)]^\top$ denoting line voltage $U(x,t)$ with respect to ground, and line current $I(x,t)$. The coefficients R, L, G, and C are the resistance, inductance, conductance, and capacitance per unit length. This first order system of PDEs is initialized by a set of initial values $u(x,0) = g(x)$, $x \in [0,1]$. If we do not neglect any effects, R, L, G and $C > 0$ holds. Thus (2.1) defines a conservation law with source term, a linear hyperbolic PDE system with two waves traveling to the left and right, respectively.

Depending on the terminating elements that link the transmission line to the electrical networks, we can distinguish two levels of mathematical models:

- *Level* 1*a: PDE system with purely time-dependent boundary conditions.*
  If we use two purely time-dependent sources that fix the boundary conditions, for example, voltage source $V(t)$ at the left and current source $J(t) \equiv 0$ (for an open end) at the right end of the transmission line, we get a hyperbolic initial-boundary value problem. The corresponding boundary conditions for (2.1) then read as follows:

  $$U(0,t) - V(t) = 0, \qquad I(1,t) = 0. \tag{2.2}$$

  A network description for this system is given in Figure 2.1.
- *Level* 1*b: PDE system with DAE boundary conditions.*
  However, if the right end is closed by a capacitor with capacitance C_1, the boundary conditions of system (2.1) are now given by DAEs in time

  $$U(0,t) - V(t) = 0, \qquad C_1 U_t(1,t) - I(1,t) = 0, \tag{2.3}$$

  with line voltage and current at both ends as the only unknowns. Consider now the case where the transmission line is coupled with other electrical networks. If, for example, the voltage source is driven by the node voltage of a JL cutset[1] [10] (see Figure 2.2) the boundary conditions are now DAE systems in time with line voltage and currents, as well as network variables z_1 and $z_2 := I_L$ of the coupled systems as unknowns:

  $$U(0,t) - z_1 = 0, \tag{2.4a}$$
  $$C_1 U_t(1,t) - I(1,t) = 0, \tag{2.4b}$$

[1]Let an electrical network be cut into two parts. If the network elements connected to cut nodes consist only of current sources (J) and inductors (L), the circuit is said to contain a JL cutset. Hence the left circuit of Figure 2.2 is the generic case for circuits containing JL cutsets.

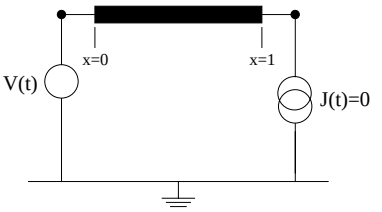


FIG. 2.1. *Transmission line driven by voltage source at the left end and open right end.*

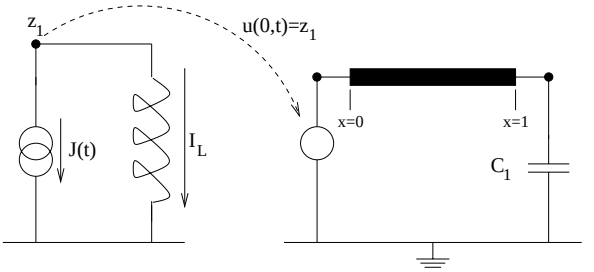


FIG. 2.2. *VC loop with transmission line. The voltage source is driven by JL cutset.*

$$z_2 + J(t) = 0, \tag{2.4c}$$

$$L_{ind}\dot{z}_2 - z_1 = 0. \tag{2.4d}$$

The next example describes a model for a countercurrent heat exchanger used in process simulation.

A plain scheme for a countercurrent heat exchanger is illustrated in Figure 2.3 where two fluids are flowing in the opposite direction and exchange heat through a metal layer. The mathematical model is derived using energy balances under simplifying assumptions:

$$u_t + Bu_x + Cu = 0\,, \qquad u(x,0) = g(x)\,, \tag{2.5}$$

where $u(x,t) = (T_1, T_2, T_m)^\top$, $x \in [0, L]$, $t \in [0, T]$, and

$$B = \begin{pmatrix} v_1 & 0 & 0 \\ 0 & -v_2 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \qquad C = \begin{pmatrix} \frac{v_1\alpha_1 A_1}{Ln_1 C_{p_1}} & 0 & -\frac{v_1\alpha_1 A_1}{Ln_1 C_{p_1}} \\ 0 & \frac{v_2\alpha_2 A_2}{Ln_2 C_{p_2}} & -\frac{v_2\alpha_2 A_2}{Ln_2 C_{p_2}} \\ -\frac{\alpha_1 A_1}{mC_{p_m}} & -\frac{\alpha_2 A_2}{mC_{p_m}} & \sum_{i=1}^{2} \frac{\alpha_i A_i}{mC_{p_m}} \end{pmatrix}$$

with fluid temperature $T_i(x,t)$, metal temperature $T_m(x,t)$, fluid velocity v_i, heat transfer area A_i, heat transfer coefficients α_i, length of the heat exchanger L, mass of the heat exchanger m, molar quantity flux n_i, heat capacity of the metal C_{p_m}, and molar heat capacity of the fluid C_{p_i}, $i = 1, 2$. For a more detailed description, see [23].

As before, different levels can be distinguished for the system (2.5) with respect to boundary conditions:

- *Level 2a: PDAE system with purely time-dependent boundary conditions.* System (2.5) together with boundary conditions

$$u^1(0,t) = s_1(t), \quad u^2(1,t) = s_2(t) \tag{2.6}$$

defines an initial-boundary value problem of PDAEs.

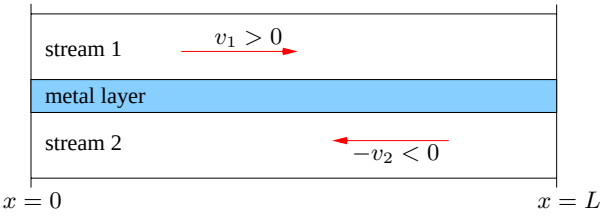


FIG. 2.3. *Countercurrent heat exchanger.*

- *Level 2b: PDAE system with DAE boundary conditions.*
  Correspondingly, system (2.5) together with DAE boundary conditions, e.g.,

$$\begin{aligned} u^1(0,t) &= s_1(t)\,, \\ \dot{z}(t) + u^2(1,t) &= s_2(t)\,, \\ z(t) &= s_3(t)\,, \end{aligned} \tag{2.7}$$

  yields an initial-boundary value problem of PDAEs with DAEs as boundary conditions that may be linked with the network variables z of the coupled networks.

With these examples in mind, we now define a general first order linear PDAE system of dimension n_u as

$$\mathcal{F}(u, u_t, u_x, f) := Au_t + Bu_x + Cu - f = 0, \quad x \in [0,1], \quad t \in [0,T], \tag{2.8a}$$

with consistent initial values

$$u(x,0) = g(x). \tag{2.8b}$$

Note that we define the system on closed intervals of definition, forcing initial and boundary data to fulfill the PDAE equations analogous to the DAE case. If at least one of the matrices A and B is singular, system (2.8a) is called partial differential-algebraic. In order to guarantee the hyperbolicity of this system of first order, we further assume A regular and $A^{-1}B$ real diagonalizable. Otherwise it would be possible to obtain parabolic or elliptic parts. For example, consider (2.1) with $L = G = 0$. Then there are two cases to distinguish: For $R = 0$ the system can be transformed to the elliptic Laplace equation $I_{xx} = 0$, and for $R \neq 0$ we get the parabolic heat equation $RCu_t - u_{xx} = 0$. As a consequence, different boundary and/or initial values would have to be prescribed. These effects are avoided by the above assumption. Thus, the DAE part consists of $n_u - m$, $m = \dim \ker B$, physical boundary conditions of Dirichlet type for the PDAE system, and n_z network equations for the compact parts of the system of the following mixed type:

$$\begin{aligned} &\mathcal{R}(u(0,t), u_t(0,t), u(1,t), u_t(1,t), z(t), \dot{z}(t), s) \\ &:= R_1 \begin{bmatrix} u_t(0,t) \\ u_t(1,t) \\ \dot{z}(t) \end{bmatrix} + R_2 \begin{bmatrix} u(0,t) \\ u(1,t) \\ z(t) \end{bmatrix} - s(t) = 0 \end{aligned} \tag{2.8c}$$

with the vector of network variables $z \in \mathbb{R}^{n_z}$, $z(0) = z_0$ and the matrices $R_1, R_2 \in \mathbb{R}^{(n_u - m + n_z) \times (2n_u + n_z)}$. We assume that the DAE part fulfills the rank condition $\operatorname{rank}(\lambda R_1 + R_2) = n_z + n_u - m$ for at least one $\lambda \in \mathbb{C}$ and that the boundary conditions are consistent with the initial values.

System (2.8a) can be rewritten into characteristic form

$$w_t + \Lambda w_x + \widetilde{C} w = \widetilde{f}, \tag{2.9a}$$

where $w = X^{-1}u$, $\widetilde{C} := X^{-1}A^{-1}CX$, $\widetilde{f} := X^{-1}A^{-1}f$ with the matrix X of eigenvectors and the matrix $\Lambda = \operatorname{diag}(\lambda_1, \ldots, \lambda_{n_u})$ of eigenvalues of $A^{-1}B$. The initial conditions (2.8b) now read

$$w(x,0) = X^{-1}g(x), \tag{2.9b}$$

and the boundary conditions

$$\widetilde{R}_1 \begin{bmatrix} w_t(0,t) \\ w_t(1,t) \\ \dot{z}(t) \end{bmatrix} + \widetilde{R}_2 \begin{bmatrix} w(0,t) \\ w(1,t) \\ z(t) \end{bmatrix} - s(t) = 0 \tag{2.9c}$$

with $\widetilde{R}_i = R_i \cdot \operatorname{diag}(X, X, I_{n_z})$, $i = 1, 2$. In the following we shall see that it is often important to consider the characteristic formulation.

Example 1 (transmission line). The telegrapher's equations (2.1) rewritten in characteristic form (2.9) are given by

$$w(x,t) = X^{-1}u(x,t) \quad \text{with} \quad X = \begin{bmatrix} -\sqrt{\frac{L}{C}} & \sqrt{\frac{L}{C}} \\ 1 & 1 \end{bmatrix} \quad \text{and} \tag{2.10a}$$

$$\Lambda = \begin{bmatrix} \frac{-1}{\sqrt{LC}} & 0 \\ 0 & \frac{1}{\sqrt{LC}} \end{bmatrix}, \quad \widetilde{C} = \frac{1}{2}\begin{bmatrix} \frac{R}{L} + \frac{G}{C} & \frac{R}{L} - \frac{G}{C} \\ \frac{R}{L} - \frac{G}{C} & \frac{R}{L} + \frac{G}{C} \end{bmatrix}, \quad \widetilde{f} = 0. \tag{2.10b}$$

The boundary conditions (2.3) of level 1b, for example, now read

$$\underbrace{\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & -C_1\sqrt{L/C} & C_1\sqrt{L/C} \end{bmatrix}}_{\widetilde{R_1}} \cdot \begin{bmatrix} w_t(0,t) \\ w_t(1,t) \end{bmatrix}$$

$$+ \underbrace{\begin{bmatrix} -\sqrt{L/C} & \sqrt{L/C} & 0 & 0 \\ 0 & 0 & -1 & -1 \end{bmatrix}}_{\widetilde{R_2}} \cdot \begin{bmatrix} w(0,t) \\ w(1,t) \end{bmatrix} = \begin{bmatrix} V(t) \\ 0 \end{bmatrix}. \tag{2.10c}$$

Example 2 (countercurrent heat exchanger). The mathematical model for the countercurrent heat exchanger (2.5) is already formulated in characteristic form, with boundary conditions of either level 2a or 2b.

Since we analyze linear systems, smooth solutions can be obtained provided that the inhomogenity $f(x,t)$, consistent initial data $g(x)$, and input signals $s(t)$ are smooth enough, according to the structure of system (2.8a) and the type of boundary conditions (2.8c). The index concepts are introduced in the following section to classify these properties.

3. Index concepts. Before introducing index concepts for linear mixed systems of DAEs and hyperbolic-type PDAEs, we first review the index concept for linear systems of DAEs of the form

$$A\dot{u} + Cu = f(t), \tag{3.1}$$

where $A, C \in \mathbb{R}^{n\times n}$, $\dot{u}, u \in \mathbb{R}^n$, $f : I \to \mathbb{R}^n$, and I is a subinterval of $\mathbb{R}$. See, for example, [3, 14] for more details.

3.1. Linear DAEs. If the matrix pencil $\{A, C\}$ is regular, i.e., $\det(\lambda A + C) \not\equiv 0$ for at least one $\lambda \in \mathbb{C}$, it can be transformed into Kronecker normal form

$$\{PAQ, PCQ\} = \left\{ \begin{bmatrix} I & 0 \\ 0 & N \end{bmatrix}, \begin{bmatrix} M & 0 \\ 0 & I \end{bmatrix} \right\}$$

with regular matrices $P, Q \in \mathbb{R}^{n\times n}$, $M \in \mathbb{R}^{k\times k}$, $k \leq n$, and a nilpotent matrix $N \in \mathbb{R}^{(n-k)\times(n-k)}$. N belongs to the eigenvalue 0 of A and is of nilpotency ν_a, i.e., ν_a is the smallest number such that $N^{\nu_a} = 0$, but $N^{\nu_a - 1} \neq 0$. With the variable transformation

$$\begin{bmatrix} y \\ z \end{bmatrix} = Q^{-1}u, \qquad \begin{bmatrix} \eta(t) \\ \chi(t) \end{bmatrix} = Pf(t)$$

into differential variables $y \in \mathbb{R}^k$ and algebraic variables $z \in \mathbb{R}^{n-k}$, the transformation of the matrix pencil corresponds to a decoupling of (3.1) into an explicit ODE and a nilpotent part:

$$\dot{y} = \eta(t) - My, \tag{3.2a}$$

$$N\dot{z} = \chi(t) - z. \tag{3.2b}$$

One has to perform $\nu_a - 1$ differentiations to solve for the nilpotent part:

$$z = \sum_{i=0}^{\nu_a - 1} (-1)^i N^i \chi^{(i)}(t), \tag{3.3}$$

and a last differentiation yields an explicit ODE for z. The solution behavior of system (3.1) differs from standard ODE theory in the following ways:

- The solution has to fulfill an algebraic constraint, since $z(0)$ is fixed by χ and its higher derivatives at the initial time point. Especially, consistent initial values have to be found.
- The system is sensitive with respect to perturbations. Take as example a signal noise, modeled by the input signal $\delta(t)$ that is added to $\chi(t)$: although δ may be very small, its higher derivatives may be arbitrarily large. A severe amplification of perturbations may occur for problems with nilpotency $\nu_a \geq 2$.

These analytical results suggest that no severe numerical problems arise for $\nu_a \leq 1$: the algebraic constraint is explicitly given; hence implicit numerical integration schemes for stiff systems, which contain a solver for nonlinear equations, are suitable to treat these problems. Additionally, no amplification of round-off errors is to be expected since the system is not sensitive with respect to perturbations.

However, severe numerical problems arise for systems with nilpotency $\nu_a \geq 2$: there are hidden algebraic constraints, which may be resolved by an unstable differentiation process. Regarding perturbations from round-off errors, the numerical solution will include terms of order $\delta/h^{\nu_a - 1}$, where h is the small time step.

Since the value of ν_a defines the behavior of the system (3.1), both in a theoretical and a numerical respect, ν_a is called the *algebraic index* of system (3.1). Additionally, the observations made above motivate two different points of view.

Differential index: To obtain an explicit differential system instead of the linear-implicit system (3.1), we had to differentiate the nilpotent part (3.2b). Since numerical differentiation is an unstable procedure, the number of differentiation steps needed to get an explicit ODE system is a measure for the numerical problems to be expected

when solving systems of type (3.1). Hence the minimum number of differentiations required is called the *differential index* ν_d of the linear-implicit linear system (3.1).

Perturbation index: We have seen that derivatives of the perturbation enter the solution of (3.1). This observation leads to a new kind of index, which measures the sensitivity of the solution with respect to perturbations in the equations: The linear-implicit system (3.1) has *perturbation index* ν_p if derivatives of perturbations up to degree ν_p enter the derivative of the solution.

All index notions are equivalent in the case of linear DAE systems: $\nu_a = \nu_p = \nu_d$. Hence we just speak of *the* DAE index for a linear DAE system throughout this paper.

Remark 1. In the nonlinear case, both definitions for the differential and perturbation index can be generalized in a straightforward way; see [7, 11] for technical details. However, the differential and perturbation index may differ arbitrarily for nonlinear systems [4].

In the following sections, we will apply and extend the index concept introduced in [6, 16] for parabolic-type PDAEs to linear mixed systems of DAEs and hyperbolic-type PDAEs. However, the time index, which needs periodic boundary conditions, is replaced by a MOL index based on an upwind semidiscretization. The *spatial index* and *MOL index* will measure the sensitivity of the system with respect to slight perturbations in the initial data and input signals, respectively. These index concepts are a measure for the sensitivity of the ADAE system after semidiscretization with respect to space and time. The *perturbation index* aims at the sensitivity of the system in both space and time.

3.2. A spatial index. The theory for the spatial index can be directly transferred from the approach for parabolic systems. We apply the Laplace transformation as described in [16] to (2.8a). If a componentwise growth condition

$$|u(x,t)| \le Ke^{ct}, \quad K, c \in \mathbb{R}, \qquad K > 0,$$

for u is fulfilled, u can be transformed into

$$U_\xi(x) = \int_0^\infty e^{-\xi t} u(x,t)\, dt, \quad \xi \in \mathbb{C}, \quad \operatorname{Re}\xi > c.$$

If the growth conditions holds for u_t, f, s, and z, too, this transformation leads to

$$BU'_\xi(x) + (A\xi + C)U_\xi(x) = F_\xi(x) + Ag(x)$$

for (2.8a) and to

$$[R_1\xi + R_2] \begin{bmatrix} U_\xi(0) \\ U_\xi(1) \\ Z_\xi \end{bmatrix} = S_\xi + R_1 \begin{bmatrix} g(0) \\ g(1) \\ z_0 \end{bmatrix}$$

for the boundary conditions (2.8c), where $F_\xi(x), S_\xi, Z_\xi$ are the Laplace-transforms of $f(x,t), s(t), z(t)$, respectively.

For the ODE case, B regular, the spatial index ν_S is defined to be zero. Else, if B is singular, we get a BVP-DAE system where we define the spatial index ν_S as the algebraic index of the matrix pencil $\{B, A\xi + C\}$. Here, we have to bear in mind that in the previous chapter we defined the algebraic index for a real-valued matrix pencil, whereas $\{B, A\xi + C\}$ is a polynomial matrix pencil. However, the index can be defined analogously for matrix elements of any field. In our case, as we assumed A

regular, it is not possible to obtain a spatial index > 1. Looking at the characteristic formulation (2.9a) and considering the matrix pencil $\{\Lambda, I\xi + \widetilde{C}\}$, we immediately get $\nu_S = 0$ for Λ regular and $\nu_S = 1$ for Λ singular for $\operatorname{Re}\xi$ sufficiently large.

Example 3 (transmission line). For the transmission line system (2.1) we have a regular matrix Λ in (2.10a), and hence space index $\nu_S = 0$ for either boundary conditions (2.3) or (2.4).

Example 4 (countercurrent heat exchanger). Consider the two stream countercurrent heat exchanger (2.5) with boundary conditions (2.6) of level 2a. We apply the Laplace transformation $u \longrightarrow U_\xi$ to (2.5) and obtain

$$BU'_\xi(x) + (\xi I + C)U_\xi(x) = g(x).$$

The boundary conditions transform equivalently into

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} U_\xi(0) + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} U_\xi(1) = \begin{bmatrix} \int_0^\infty e^{-\xi t} s_1(t)dt \\ \int_0^\infty e^{-\xi t} s_2(t)dt \end{bmatrix}.$$

The resulting DAE-BVP has index 1, since B is a singular diagonal matrix and $\xi I + C$ is regular with positive entries on the diagonal. This is what we would expect considering the underlying PDE. This system can be solved for U_ξ, and the final solution u is obtained by back transformation.

3.3. A MOL index. In [6, 16] definitions for a modal and time index are given in the context of parabolic PDAEs of type

$$Au_t + Du_{xx} + Cu = f \tag{3.4}$$

with homogeneous boundary conditions. It is possible to expand u in orthogonal eigenfunctions $\sin k\pi x$ of the operator d^2/dx^2,

$$u(x,t) = \sum_{k=1}^{\infty} u_k(t) \sin k\pi x,$$

fulfilling the same boundary conditions as the above system. With the Galerkin principle, the $u_k(t)$ decouple and the DAEs for $u_k(t)$ can be solved independently. This approach is not applicable if an additional first order term Bu_x arises in (3.4).

However, in system (2.8) we have to face this case. Additionally, we cannot assume homogeneous, or more generally, periodic boundary conditions for first order hyperbolic systems. Hence we propose a MOL index rather than a time index to measure the sensitivity of system (2.8) with respect to time: the following index definition is not based on an expansion of the PDAE solution in orthogonal eigenfunctions, but on a semidiscretization with respect to space that is applied to the characteristic formulation (2.9) of the system.

In order to reflect properly the information flow in the system, we use first order approximations of w_x in the different upwind directions. We use a uniform grid $\Omega_h = \{x_k : \ x_k = kh, \ k = 0, 1, \ldots, N, \ h = 1/N\}$. Introducing the discrete vector $W^h(t) = [w_0^h(t)^\top, w_1^h(t)^\top, \ldots, w_N^h(t)^\top]^\top$ with $w_k^h(t) \approx w(t, x_k)$, $k = 0, 1, \ldots, N$, we can define the following ADAE system in the unknown W:

$$\dot{w}_j^{h,i} + \sum_{k=1}^{n_u} c_{ik} w_j^{h,k} + \left\{ \begin{array}{ll} \frac{\lambda_i}{h}\left[w_j^{h,i} - w_{j-1}^{h,i}\right] & \text{if } \lambda_i > 0 \\ \frac{\lambda_i}{h}\left[w_{j+1}^{h,i} - w_j^{h,i}\right] & \text{if } \lambda_i < 0 \\ 0 & \text{if } \lambda_i = 0 \end{array} \right\} = f^i(x_j, t) \tag{3.5a}$$

for $i = 1, \ldots, n_u$, $j = 1, \ldots, N-1$. The numerical boundary conditions at the outflows are given by

$$\dot{w}_N^{h,i} + \sum_{k=1}^{n_u} c_{ik} w_N^{h,k} + \frac{\lambda_i}{h}\left[w_N^{h,i} - w_{N-1}^{h,i}\right] = f^i(x_N, t) \tag{3.5b}$$

for $\lambda_i > 0$ and

$$\dot{w}_0^{h,i} + \sum_{k=1}^{n_u} c_{ik} w_0^{h,k} + \frac{\lambda_i}{h}\left[w_1^{h,i} - w_0^{h,i}\right] = f^i(x_0, t) \tag{3.5c}$$

for $\lambda_i < 0$. The system is completed by the physical boundary conditions (2.9c)

$$\widetilde{R}_1 \begin{bmatrix} \dot{w}_0^h(t) \\ \dot{w}_N^h(t) \\ \dot{z}^h(t) \end{bmatrix} + \widetilde{R}_2 \begin{bmatrix} w_0^h(t) \\ w_N^h(t) \\ z^h(t) \end{bmatrix} - s(t) = 0. \tag{3.5d}$$

Note that $z^h(t)$ is also indexed by h since we get different approximate solutions $z^h(t)$ for different h due to the coupling with $W^h(t)$.

DEFINITION 3.1. *System* (3.5) *defines a family* $\{F_N\}_{N\in\mathbb{N}_0}$ *of linear DAE systems with solution* $[W^h(t), z^h(t)]^\top$ *with index* ν_N. *We shall take the MOL index* ν_T *of the PDAE* (2.8) *to be* ν_N *for* N *sufficiently large, provided* ν_N *does not change for all* $N > N_0$ *with some* $N_0 \in \mathbb{N}_0$.

Example 5 (transmission line). Using the results of Example 1, we obtain with

$$M = (m)_{ij} = \delta_{i+1,j} - \delta_{i,j}, \quad i = 1, \ldots, N, \quad j = 1, \ldots, N+1,$$

and $c_1 = R/L + G/C$, $c_2 = R/L - G/C$ the DAE system

$$\begin{bmatrix} \dot{w}_0^{h,1} \\ \vdots \\ \dot{w}_{N-1}^{h,1} \end{bmatrix} - \frac{1}{h\sqrt{LC}} M \begin{bmatrix} w_0^{h,1} \\ \vdots \\ w_N^{h,1} \end{bmatrix} + \frac{1}{2}\left(c_1 \begin{bmatrix} w_0^{h,1} \\ \vdots \\ w_{N-1}^{h,1} \end{bmatrix} + c_2 \begin{bmatrix} w_0^{h,2} \\ \vdots \\ w_{N-1}^{h,2} \end{bmatrix} \right) = 0, \tag{3.6a}$$

$$\begin{bmatrix} \dot{w}_1^{h,2} \\ \vdots \\ \dot{w}_N^{h,2} \end{bmatrix} + \frac{1}{h\sqrt{LC}} M \begin{bmatrix} w_0^{h,2} \\ \vdots \\ w_N^{h,2} \end{bmatrix} + \frac{1}{2}\left(c_2 \begin{bmatrix} w_1^{h,1} \\ \vdots \\ w_N^{h,1} \end{bmatrix} + c_1 \begin{bmatrix} w_1^{h,2} \\ \vdots \\ w_N^{h,2} \end{bmatrix} \right) = 0, \tag{3.6b}$$

$$\sqrt{L/C}\left[w_0^{h,2}(t) - w_0^{h,1}(t)\right] = V(t) \tag{3.6c}$$

$$C_1\sqrt{L/C}\left[\dot{w}_N^{h,2}(t) - \dot{w}_N^{h,1}(t)\right] - \left[w_N^{h,1}(t) + w_N^{h,2}(t)\right] = 0 \tag{3.6d}$$

in $W^h(t)$. Obviously, this system has index one independent of h, which yields the MOL index $\nu_T = 1$. Correspondingly, the system with boundary conditions (2.4) has MOL index $\nu_T = 2$.

Example 6 (countercurrent heat exchanger). Consider again the model for the countercurrent heat exchanger (2.5) with boundary conditions (2.6) and (2.7), respectively, of levels 2a and 2b. The system is already in characteristic form. The

ADAE (3.5a)–(3.5c) now reads

$$\dot{u}_j^{h,1} + \frac{v_1}{h}\left[u_j^{h,1} - u_{j-1}^{h,1}\right] + \frac{v_1\alpha_1 A_1}{Ln_1 C_{p_1}}\left[u_j^{h,1} - u_j^{h,3}\right] = 0, \quad j = 1, \ldots, N,$$

$$\dot{u}_j^{h,2} - \frac{v_2}{h}\left[u_{j+1}^{h,2} - u_j^{h,2}\right] + \frac{v_2\alpha_2 A_2}{Ln_2 C_{p_2}}\left[u_j^{h,2} - u_j^{h,3}\right] = 0, \quad j = 0, \ldots, N-1,$$

$$\dot{u}_j^{h,3} + \frac{\alpha_1 A_1}{mC_{p_m}}\left[u_j^{h,3} - u_j^{h,1}\right] + \frac{\alpha_2 A_2}{mC_{p_m}}\left[u_j^{h,3} - u_j^{h,2}\right] = 0, \quad j = 0, \ldots, N,$$

together with either

$$u_0^{h,1} = s_1(t), \qquad u_N^{h,2} = s_2(t)$$

for the boundary conditions (2.6) of level 2a or

$$u_0^{h,1} = s_1(t), \quad \dot{z}^h + u_N^{h,2} = s_2(t), \quad z^h = s_3(t)$$

for boundary conditions (2.7) of level 2b. Thus we have $\nu_T = 1$ in the first and $\nu_T = 2$ in the latter case.

3.4. The perturbation index. Let $[u, z]^\top$ be the (reference) solution of the PDAE system (2.8a)–(2.8c) with consistent initial values $u(x, 0) = g(x)$ and $z(0) = z_0$. To investigate the sensitivity of this solution with respect to initial values, input signals, and inhomogenities, we apply perturbations $\delta(x, t)$ and $\mu(t)$ to the right-hand sides of (2.8a) and (2.8c). The corresponding perturbed solution is denoted by $[\hat{u}, \hat{z}]^\top$ with consistent perturbed initial values $\hat{u}(x, 0) = \hat{g}(x)$ and $\hat{z}(0) = \hat{z}_0$. The aim is to obtain estimates for the sensitivity of the solution. As we are especially interested in the influence of perturbations in the time-dependent boundary conditions, we will consider the PDAE as an evolution problem.

We consider first the telegrapher's equations (2.1) which are coupled with a network without inductors and voltage sources: the DAE boundary conditions (2.8c) then read

$$\widetilde{C}\dot{z} + \widetilde{G}z + P\begin{bmatrix} I(0,t) \\ -I(1,t) \end{bmatrix} = V(t) \tag{3.8}$$

with a positive-definite, symmetric capacitance matrix $\widetilde{C}$, a conductance matrix $\widetilde{G}$, an incidence matrix P that assembles the current contributions from the transmission lines at the respective nodes, and the coupling condition

$$\begin{bmatrix} U(0,t) \\ U(1,t) \end{bmatrix} = P^\top z$$

for the boundary node voltages. Due to the linearity of the systems, $[\bar{u}, \bar{z}]^\top := [\hat{u} - u, \hat{z} - z]^\top$ is now the solution of (2.1), (3.8) with δ and μ as inhomogenities for the initial values $\bar{u}(x, 0) = \bar{g}(x) := \hat{g}(x) - g(x)$ and $\bar{z}(0) = \bar{z}_0 = \hat{z}_0 - z_0$.

To get an estimate for $\bar{u}$, we consider the inner product of the PDE with $[\bar{I}, \bar{U}]^\top$. Integration over $(0, 1)$ with respect to x and integration by parts yield

$$\bar{U}(\cdot,t)\bar{I}(\cdot,t)\Big|_0^1 + \frac{1}{2}\frac{d}{dt}\Big[(L\bar{I},\bar{I}) + (C\bar{U},\bar{U})\Big] + (R\bar{I},\bar{I}) + (G\bar{U},\bar{U}) + (\bar{I},\delta_1) + (\bar{U},\delta_2) = 0$$

with $(\cdot,\cdot)$ denoting the inner product in $L^2(0,1)$. Using the coupling condition and $P^\top P = I$, the first term can be replaced by

$$\frac{1}{2}\frac{d}{dt}\left[\bar{z}^\top \widetilde{C}\bar{z}\right] + \bar{z}^\top \widetilde{G}\bar{z} - \bar{z}^\top \mu(t).$$

Integrating over $(0,t)$, we get with $C, L > 0$ and $\widetilde{C}$ positive definite the estimate

$$\rho(t) := \|\bar{U}(\cdot,t)\|_{L^2}^2 + \|\bar{I}(\cdot,t)\|_{L^2}^2 + \|\bar{z}(t)\|_2^2 \tag{3.9}$$

$$\leq \text{ const } \left(\rho(0) + \int_0^t \left(\|\mu(\tau)\|_2^2 + \|\delta(\cdot,\tau)\|_{L^2}^2 + \rho(\tau)\right) d\tau\right). \tag{3.10}$$

Here $\|\cdot\|_2$ denotes the Euclidian norm on $\mathbb{R}^n$, and $\|u(\cdot,t)\|_{L^2}^2 := (u(\cdot,t),u(\cdot,t))$ the norm associated to the inner product. By Gronwall's lemma we end up with

$$\max_{t\in[0,T]} \|u(\cdot,t) - \hat{u}(\cdot,t)\|_{L^2} + \max_{t\in[0,T]} \|z(t) - \hat{z}(t)\|_2 \tag{3.11}$$

$$\leq \text{ const} \left(\|g - \hat{g}\|_{L^2} + \|z_0 - \hat{z}_0\|_2 + \max_{t\in[0,T]} \left(\|\mu(t)\|_2 + \|\delta(\cdot,t)\|_{L^2}\right)\right).$$

Generalizing this estimate, a perturbation index can be defined similar to the definition in [6] as follows.

DEFINITION 3.2. *If there exists for all $t \in [0,T]$ an estimate for the reference solution $[u,z]^\top$ with initial values $[g(x),z_0]^\top$ of system* (2.8a)–(2.8c) *of the type*

$$||u - \hat{u}||_{C([0,T],L^2)} + ||z - \hat{z}||_{C([0,T])} \tag{3.12}$$

$$\leq \Gamma \left\{ ||g - \hat{g}||_{H^{q_2}} + ||z_0 - \hat{z}_0||_2 + \left(\|\mu\|_{C^{p_2}([0,T])} + \|\delta\|_{C^{p_1}([0,T],H^{q_1})}\right)\right\},$$

where $[\hat{u},\hat{z}]^\top$ denotes the corresponding perturbed solution with perturbations $[\delta(x,t),\mu(t)]^\top$, then the perturbation index ν_P is defined by

$$\nu_P = 1 + \min\left\{\max(p_1+q_1, p_2, q_2)\middle|\ (3.12) \text{ holds for } (p_1,q_1,p_2,q_2)\right\}.$$

The used norms are given by

$$||\delta||_{C^{p_1}([0,T],H^{q_1})} := \max_{t\in[0,T]} \sqrt{\int_0^1 \sum_{i=0}^{p_1}\sum_{k=0}^{q_1} \left\| \frac{\partial^{i+k}\delta(x,t)}{\partial t^i \partial x^k}\right\|_2^2 dx},$$

$$||g||_{H^{q_2}} := \sqrt{\int_0^1 \sum_{k=0}^{q_2} \left\| \frac{d^k g(x)}{dx^k}\right\|_2^2 dx},$$

$$||\mu||_{C^{p_2}([0,T])} := \max_{t\in[0,T]} \sqrt{\sum_{i=0}^{p_2} \left\| \frac{d^i \mu(t)}{dt^i}\right\|_2^2}.$$

Remark 2.

- Γ is a nonzero constant independent of δ, μ, and $g - \hat{g}$. The constant depends only on A, B, C, R_1, R_2, and T.

- The perturbations and the respective consistent initial data have to be smooth enough to guarantee the existence of the norms on the right-hand side of (3.12).

Even for a hyperbolic system with purely time-dependent boundary conditions we can get perturbation index greater than one, where the analytical solution depends on derivatives of both initial values and input signals.

Example 7. Consider again telegrapher's equations of Example 1 in characteristic form with $L = C = 1$, $R = G = 0$, boundary conditions

$$\begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} w_t(0,t) \\ w_t(1,t) \end{bmatrix} + \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} w(0,t) \\ w(1,t) \end{bmatrix} = s(t),$$

and consistent initial values $w(x,0) = g(x)$. The analytical solution in $[0,1]^2$ reads

$$w^1(x,t) = \begin{cases} g_1(x+t), & 0 \le x+t \le 1, \\ s_1(x+t-1) - g_1(x+t-1), & \\ -\dot{s}_2(x+t-1) - \dot{g}_2(2-(x+t)), & 1 \le x+t \le 2, \end{cases}$$

$$w^2(x,t) = \begin{cases} g_2(x-t), & t \le x \le 1, \\ s_2(t-x) - g_2(1-t+x), & x \le t \le 1, \end{cases}$$

which yields an estimate of type (3.12) with $q_2 = p_2 = 1$ for the trivial solution as a reference, when neglecting disturbances in the inhomogenity.

For systems with purely time-dependent boundary conditions we state in the following lemma an estimate depending on the inhomogenity f and the initial conditions g, which extends the energy estimate results for Cauchy problems of symmetric hyperbolic systems [21].

LEMMA 3.3. *Consider system* (2.9a)

$$u_t + \Lambda u_x + Cu = f(x,t), \quad x \in [0,1], \quad t \in [0,T] \tag{3.13}$$

with $\Lambda = diag(\lambda_1, \dots, \lambda_r, 0, \dots, 0)$, $\lambda_i > 0$ *for* $1 \le i \le l$ *and* $\lambda_i < 0$ *for* $l+1 \le i \le r$, *initial condition* $u(x,0) = g(x)$ *and boundary conditions of type* (2.8c). *One obtains the following estimate for the solution* $u \in C([0,T], L^2)$ *for the system* (3.13), (2.8c)

$$||u(\cdot,t)||_{C([0,T],L^2)} \le \Gamma \left(||g||_{L^2} + ||f||_{C([0,T],L^2)} + ||\chi||_{C([0,T])} \right),$$

with $\chi = (\chi_1, \dots, \chi_r)^\top$, Γ *constant and the physical boundaries defined by*

$$u_i(0,t) = \chi_i(t), \quad i = 1, \dots, l, \text{ respectively, } u_i(1,t) = \chi_i(t), \quad i = l+1, \dots, r. \tag{3.14}$$

Proof. Integrating the inner product of (3.13) with u over the region $R_t = \{(\tau, x) |\, 0 \le \tau \le t,\, x \in [0,1]\}$ in the t-x-plane gives

$$\begin{aligned} \|u(\cdot,t)\|_{L^2}^2 = &\int_{R_t} (-2u^\top Cu + 2u^\top f) dx\, d\tau + ||g||_{L^2}^2 \\ &+ \int_0^t [u(0,\tau)^\top \Lambda u(0,\tau) - u(1,\tau)^\top \Lambda u(1,\tau)] d\tau\,. \end{aligned}$$

As C is constant, we find $-2u^\top Cu \le Ku^\top u$ with a constant $K > 0$; additionally, the expression with the inhomogenity f is estimated by

$$2u^\top f \le u^\top u + f^\top f\,.$$

In order to insert the boundary conditions, Λ is split:

$$u(0,\tau)^\top \Lambda u(0,\tau) = \sum_{k=1}^{l} \lambda_k \chi_k(\tau)^2 + \sum_{k=l+1}^{r} \lambda_k u_k(0,\tau)^2\,, \qquad \text{respectively,}$$

$$u(1,\tau)^\top \Lambda u(1,\tau) = \sum_{k=1}^{l} \lambda_k u_k(1,\tau)^2 + \sum_{k=l+1}^{r} \lambda_k \chi_k(\tau)^2\,.$$

Due to the different signs the following inequality holds with $M_1 := K + 1$:

$$\|u(\cdot,t)\|_{L^2}^2 \le M_1 \int_0^t \|u(\cdot,\tau)\|_{L^2}^2 d\tau + ||g||_{L^2}^2 + T \max_{0\le\tau\le t} \Big(\max_k |\lambda_k| \cdot ||\chi(\tau)||_2^2 + ||f(\cdot,\tau)||_{L^2}^2 \Big).$$

With $M_2 := \max\{1, T, T \max_{k=1,\dots,r} |\lambda_k|\}$ Gronwall's lemma yields

$$\|u(\cdot,t)\|_{L^2}^2 \le \left(||g||_{L^2}^2 + \max_{0\le\tau\le t} \Big(||\chi(\tau)||_2^2 + ||f(\cdot,\tau)||_{L^2}^2 \Big) \right) \cdot M_2 e^{M_1 T}, \tag{3.15}$$

and we obtain with $\Gamma := \sqrt{M_2 e^{M_1 T}}$ the desired inequality of the above lemma. ∎

COROLLARY 1. *If only physical boundaries $\chi_k(t)$, $k = 1,\dots,r$, appear, then the perturbation index ν_P is one higher than the degree of the highest derivative of the boundary data fulfilling an estimate*

$$||u(\cdot,t) - \hat{u}(\cdot,t)||_{L^2} \le \Gamma \left(||g - \hat{g}||_{L^2} + ||\delta||_{C([0,T];L^2)} + ||\mu||_{C^q([0,T])} \right)$$

with the norms of Lemma 3.3*, i.e., $\nu_P = q+1$. ν_P then corresponds to the DAE index of the boundary conditions.*

Example 8 (transmission line). Consider now the telegrapher's equations for a single transmission line given in (2.1) with both boundary conditions (2.3) and (2.4) of level 1b. Neglecting the anyway regularizing source term, line resistance R and conductance G, the system perturbed with $\delta(x,t)$ at the right-hand sides of (2.1) and $\mu(t)$ at the right-hand sides of (2.4) and (2.3), respectively, can be solved analytically for $t \in [0,1]$; see the appendix. For simplicity, we have fixed $C = L = 1$, and thus normalized the wave speed to 1. For system (2.1), (2.4) we get an estimate of type (3.12) with $(p_1, q_1, p_2, q_2) = (0,0,1,0)$, and the perturbation index is 2. The system is only sensitive with respect to perturbations in the input signals. For system (2.1), (2.3) the index reduces to 1.

Example 9 (countercurrent heat exchanger). Analyzing again the model for the countercurrent heat exchanger (2.5), we get with Corollary 1 that for purely time-dependent boundary conditions (2.6) the perturbation index is one, whereas the index-2 boundary conditions (2.7) yield a perturbation index two.

4. Benchmark examples. The index concepts derived in section 3 are now applied to the two previously introduced benchmark examples from chemical and electrical engineering.

On the one hand, the countercurrent heat exchanger model is already in characteristic form. Not surprisingly, semidiscretization with respect to space leads to ADAE systems with an index indicated by the perturbation and MOL index. Thus we will focus on the connection between spatial index and the ADAE systems obtained after semidiscretization with respect to time.

On the other hand, the higher sensitivity with respect to input signals in the transmission line model cannot be resolved by a horizontal MOL, which always will

lead to ODE systems. Therefore we will concentrate on the connection between MOL index and the ADAE systems obtained after semidiscretization with respect to space.

Then an interesting point in both cases will be the comparison between the perturbation index and the different indices mentioned above.

4.1. Countercurrent heat exchanger. First consider system (2.5) with boundary conditions (2.6) of level 2a. The perturbation and MOL index of this system are both equal to one, as shown in Examples 6 and 9.

The following subsection compares the spatial index and the DAE index of the approximative system obtained after application of some Rothe method, where — in contrast to the classical method of lines — semidiscretization is made with respect to time to obtain a boundary value problem in space alone. As the perturbation index measures the sensitivity of the original system with respect to boundary and initial values, the ADAE should have the same property.

Using the implicit Euler scheme as time discretization with $N+1$ grid points $t_j = j\tau$ in the horizontal MOL, we also have to solve a DAE of index one ($u_j^\tau(x)$ denotes the numerical approximation for $u(x,t_j)$):

$$\tau \begin{bmatrix} 0 & & & \\ & B & & \\ & & \ddots & \\ & & & B \end{bmatrix} \begin{bmatrix} {u_0^\tau}'(x) \\ {u_1^\tau}'(x) \\ \vdots \\ {u_N^\tau}'(x) \end{bmatrix} + \begin{bmatrix} I & & & & \\ -I & I+\tau\cdot C & & & \\ & -I & \ddots & & \\ & & \ddots & \ddots & \\ & & & -I & I+\tau\cdot C \end{bmatrix} \begin{bmatrix} u_0^\tau(x) \\ u_1^\tau(x) \\ \vdots \\ u_N^\tau(x) \end{bmatrix} = \begin{bmatrix} g(x) \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

with boundary values

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} u_j^\tau(0) + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} u_j^\tau(1) = \begin{bmatrix} s_1(j\tau) \\ s_2(j\tau) \end{bmatrix}.$$

For the implicit Euler scheme, the DAE index of the approximative system coincides with the spatial and perturbation index.

However, considering the explicit Euler scheme

$$\tau \begin{bmatrix} 0 & & & \\ B & 0 & & \\ & \ddots & \ddots & \\ & & B & 0 \end{bmatrix} \begin{bmatrix} {u_0^\tau}'(x) \\ {u_1^\tau}'(x) \\ \vdots \\ {u_N^\tau}'(x) \end{bmatrix} + \begin{bmatrix} I & & & \\ -I+\tau\cdot C & \ddots & & \\ & \ddots & I & \\ & & -I+\tau\cdot C & I \end{bmatrix} \begin{bmatrix} u_0^\tau(x) \\ u_1^\tau(x) \\ \vdots \\ u_N^\tau(x) \end{bmatrix} = \begin{bmatrix} g(x) \\ 0 \\ \vdots \\ 0 \end{bmatrix},$$

we obtain index $N+1$ because of the influence of the initial conditions. Thus, the index of the ADAE system in x does not always coincide with the spatial and perturbation

index. Such a behavior has to be avoided. Anyway, the explicit Euler method is not suitable for a time discretization, as the boundary values cannot be chosen freely.

Generally, the problems due to DAE boundary values are not well reflected using any Rothe method, i.e., the index would be the same regardless of applying (2.6) or (2.7).

4.2. VC loop with transmission line. Our next benchmark example arises from network theory. We consider the telegrapher's equations for a single transmission line given in (2.1) with both boundary conditions (2.3) and (2.4) (level 1b).

If the transmission line is skipped, we know from network theory that the DAE system has index three and two if the voltage source of the VC loop is driven by a JL cutset and a time-dependent voltage source, respectively; see [10].

In Examples 5 and 8 we have seen that both perturbation and MOL index for the corresponding systems with transmission lines is two and one, respectively, — the inclusion of transmission lines has a regularizing effect.

However, index three and two, respectively, may be obtained for the ADAE after semidiscretizations with respect to space, if the information in the characteristics is not adequately used. This will be shown in the following.

4.2.1. Semidiscretization with respect to space: Characteristic form. In a first step, we consider on a uniform grid Ω^h different semidiscretizations with respect to space which are based on system (2.10) in characteristic form. With the notation of Example 5, central differences lead to

$$\dot{\bar{w}}^h + (\Lambda \otimes \widetilde{M})w^h + (\widetilde{C} \otimes I_{N-1})\bar{w}^h = 0 \quad \text{with} \tag{4.1a}$$

$$\widetilde{M} = (\widetilde{m})_{ij} = \frac{1}{2h} \cdot (\delta_{i+2,j} - \delta_{i,j}), \quad i = 1, \dots, N-1, \quad j = 1, \dots, N+1,$$

and the Kronecker product $F \otimes G$ of two matrices $F \in \mathbb{R}^{n\times r}$ and $G \in \mathbb{R}^{s\times m}$ given by $F \otimes G := (f_{ij} \cdot G) \in \mathbb{R}^{ns\times rm}$. Using the abbreviations $\bar{w}^h = [\bar{w}^{h,1}, \bar{w}^{h,2}]^\top$, $\bar{w}^{h,i} = [w_1^{h,i}, \dots, w_{N-1}^{h,i}]^\top$ for $i = 1, 2$, the vector of approximate unknowns w^h is given by $w^h = [w_0^{h,1}, \bar{w}^{h,1}, w_N^{h,1}, w_0^{h,2}, \bar{w}^{h,2}, w_N^{h,2}]^\top$. The system is completed by the physical boundary conditions (3.6c), (3.6d) and by appropriate numerical boundary conditions: for example, upwind at the boundaries yields

$$\dot{w}_0^{h,1} - \frac{1}{\sqrt{LC}h}\left(w_1^{h,1} - w_0^{h,1}\right) + \frac{1}{2}\left(c_1 w_0^{h,1} + c_2 w_0^{h,2}\right) = 0, \tag{4.1b}$$

$$\dot{w}_N^{h,2} + \frac{1}{\sqrt{LC}h}\left(w_N^{h,2} - w_{N-1}^{h,2}\right) + \frac{1}{2}\left(c_2 w_N^{h,1} + c_1 w_N^{h,2}\right) = 0. \tag{4.1c}$$

Obviously, the index is one. One could also think of linear finite element approximations $\widetilde{w}^h(x,t)$ to $w^h(x,t)$ that are constructed according to the characteristics by fixing the solution at the respective inflow boundary:

$$\widetilde{w}^{h,1}(x,t) = \frac{1}{2}\left(I^h(1,t) - \frac{1}{\sqrt{L/C}}U^h(1,t)\right)\varphi_N + \sum_{k=0}^{N-1} w_k^{h,1}(t)\varphi_k(x) \tag{4.2a}$$

$$\widetilde{w}^{h,2}(x,t) = \frac{1}{2}\left(I^h(0,t) + \frac{1}{\sqrt{L/C}}U^h(0,t)\right)\varphi_0 + \sum_{k=1}^{N} w_k^{h,2}(t)\varphi_k(x). \tag{4.2b}$$

Then Galerkin applied to (2.9), (2.10a), (2.10b), together with the physical boundary conditions

$$U^h(0,t) - V(t) = 0, \qquad C_1 \cdot U_t^h(1,t) - I^h(1,t) = 0 \tag{4.3}$$

at the transmission line ends, and the coupling conditions

$$\widetilde{w}^{2,h}(1,t) = \frac{1}{2}\left(I^h(1,t) + \frac{U^h(1,t)}{\sqrt{L/C}}\right), \ \widetilde{w}^{1,h}(0,t) = \frac{1}{2}\left(I^h(0,t) - \frac{U^h(0,t)}{\sqrt{L/C}}\right) \tag{4.4}$$

at the remaining outflow boundaries, defines a linear DAE system in the unknowns $w_0^{1,h}, w_1^{1h} \ldots, w_{N-1}^{1,h}, w_1^{2,h}, w_2^{2,h}, \ldots, w_N^{2,h}, U^h(0,t), U^h(1,t), I^h(0,t), I^h(1,t)$. One can show that this system has index one independent of h.

4.2.2. Semidiscretization with respect to space: Noncharacteristic form. The situation is quite different if semidiscretization with respect to space is applied to the original formulation (2.1), (2.3). Central differences now yield index two for some exceptional values of the discretization parameter h, but index one is assured for h being small enough. This behavior is well known for ADAE systems arising from MOLs applied to PDAEs [1, 5].

However, things may be even worse: let us investigate the use of so-called RLGC companion network models [2], which corresponds to the use of central differences with staggered grids applied to (2.1), (2.3):

$$L\begin{bmatrix} \dot{I}_{1/2}^h \\ \vdots \\ \dot{I}_{N-1/2}^h \end{bmatrix} + R\begin{bmatrix} I_{1/2}^h \\ \vdots \\ I_{N-1/2}^h \end{bmatrix} + \frac{1}{h}\begin{bmatrix} -1 & 1 & & \\ & \ddots & \ddots & \\ & & -1 & 1 \end{bmatrix}\begin{bmatrix} U_0^h \\ \vdots \\ U_N^h \end{bmatrix} = 0, \tag{4.5a}$$

$$C\begin{bmatrix} \dot{U}_0^h \\ \vdots \\ \dot{U}_N^h \end{bmatrix} + G\begin{bmatrix} U_0^h \\ \vdots \\ U_N^h \end{bmatrix} + \frac{1}{h}\begin{bmatrix} -2 & 2 & & & & \\ & -1 & 1 & & & \\ & & \ddots & \ddots & & \\ & & & -1 & 1 & \\ & & & & -2 & 2 \end{bmatrix}\begin{bmatrix} I_0^h \\ I_{1/2}^h \\ \vdots \\ I_{N-1/2}^h \\ I_N \end{bmatrix} = 0, \tag{4.5b}$$

with $U_l^h(t)$, $I_l^h(t)$ numerical approximations for $U(x,t)$, $I(x,t)$ at the points (lh,t).

Together with the boundary conditions

$$U_0^h - V(t) = 0, \tag{4.5c}$$

$$C_1\dot{U}_N^h - I_N^h = 0 \tag{4.5d}$$

the ADAE (4.5) has index two for all $C, L, C_1 > 0$ and $R, G \geq 0$ independent of h: boundary condition (4.5d) is equivalent to

$$I_N^h = \frac{1}{C/C_1 + 2/h}\left(\frac{2}{h}I_{N-1/2}^h - GU_N^h\right),$$

which defines $\dot{I}_N^h$ after one differentiation in terms of $I_{N-1/2}^h, I_N^h, U_{N-1}^h, U_N^h$. However, one differentiation of (4.5c) is necessary to obtain

$$I_0^h = I_{1/2}^h + \frac{h}{2}\left(C\dot{V}(t) + GV(t)\right),$$

and a second differentiation is necessary to get an explicit ODE for I_0. Thus I_0^h, the current through the voltage source, is an index-2 variable, as for a VC loop without transmission lines. It can be shown that linear finite elements applied to the original formulation (2.1), (2.3) have the same deregularizing effect: they also yield index-2 ADAE systems for all discretization parameters [9].

Inspecting the VC loop, we have seen that a model refinement based on including transmission lines effects may regularize DAE network equations of higher index. Using semidiscretization with respect to space, the MOL approach converts this mixed system of PDEs and DAEs into an ADAE system in time only. To reflect the physical properties of the original PDAE network model, the ADAE system should neither be more nor less sensitive, and the regularizing effect should be kept. However, semidiscretization may deregularize a regularizing PDE model, if the method of lines approach is not consistent with the information flow along characteristics.

5. Conclusion. We have extended some of the index concepts for parabolic PDAEs with homogeneous boundary conditions to hyperbolic-type initial-boundary value problems of linear PDAEs, where the boundary conditions are governed by linear time-dependent DAEs. To investigate the impact of semidiscretization in space and time, we introduced a space and a MOL index for an a posteriori analysis. However, the DAE index of the approximative system obtained after a particular semidiscretization may be different. The properties of the underlying mixed system were described by a perturbation index, which reflects the sensitivity of the original system with respect to slight perturbations in the right-hand side of the PDAE, as well as in the input signals and initial values.

By inspecting interconnected electrical circuits and a heat exchanger as examples arising in applications, we have seen that these index notions give insight into the properties of the mixed PDAE/DAE system and the DAE systems obtained after semidiscretization: the spatial and MOL index give hints for the sensitivity of the ADAEs with respect to input signals and data. These properties should be consistent with the behavior of the original system described by the perturbation index. The ADAEs should be neither more nor less sensitive, though the latter might facilitate the numerical solution of the ADAE system; however, the obtained solution could be physically incorrect. On the other hand, a spurious sensitivity of the ADAE system detected by an index larger than the MOL index might indicate that the semidiscretization with respect to space is not appropriate.

The analysis of mixed DAEs and hyperbolic-type PDAEs using index concepts provided helpful information. In order to give reliable recipes for the numerical treatment of coupled DAE and hyperbolic-type PDAE systems, further investigations should aim at a more closed theory for the different index notions and extend the analysis to nonlinear systems.

Appendix. Perturbed VC loop with transmission lines. Fixing $R = G = 0$ and $C = L = 1$, systems (2.1), (2.4) and (2.1), (2.3) perturbed with $\delta(x,t)$ at the right-hand sides of (2.1), and $\mu(t)$ at the right-hand sides of (2.4) and (2.3), respectively, can be solved analytically for $t \in [0,1]$.

For system (2.1), (2.4) we get

$$\text{(A.1a)} \qquad \begin{bmatrix} U(x,t) \\ I(x,t) \end{bmatrix} = \underbrace{\begin{bmatrix} -1 & 1 \\ 1 & 1 \end{bmatrix}}_{T} \cdot \begin{bmatrix} w^1(x,t) \\ w^2(x,t) \end{bmatrix}$$

with the characteristic variables w^1 and w^2.

1. $1 \le x+t \le 2$:

$$w^1(x,t) = w_0^2(2-(x+t)) - \exp\left(\frac{1-(x+t)}{C_1}\right)(w_0^1(1) - w_0^2(1))$$
$$- \int_0^{(x+t)-1} \exp\left(\frac{(1+\tau)-(x+t)}{C_1}\right)\left[\frac{\mu_2(\tau)}{C_1} + \frac{2}{C_1}\left(w_0^2(1-\tau)\right.\right.$$
$$\left.\left. + \int_0^{\tau} \widetilde{\delta}_2(\xi + (1-\tau), \xi)\, d\xi\right)\right] d\tau + \int_0^{(x+t)-1} \widetilde{\delta}_2(\tau + 2 - (x+t), \tau)\, d\tau$$
$$+ \int_0^{1-x} \widetilde{\delta}_1(1-\tau, \tau + (x+t) - 1)\, d\tau. \tag{A.1b}$$

2. $0 \le x+t \le 1$:

$$w^1(x,t) = w_0^1(x+t) + \int_x^{x+t} \widetilde{\delta}_1(\xi, (x+t) - \xi)\, d\xi. \tag{A.1c}$$

3. $0 \le x-t \le 1$:

$$w^2(x,t) = w_0^2(x-t) + \int_0^{t} \widetilde{\delta}_2(\xi + (x-t), \xi)\, d\xi. \tag{A.1d}$$

4. $0 \le t-x \le 1$:

$$w^2(x,t) = \mu_4(t-x) + \mu_1(t-x) + L_{ind}(\dot{\mu}_3(t-x) - \dot{J}(t-x)) + w_0^1(t-x)$$
$$+ \int_0^{t-x} \widetilde{\delta}_1(\xi, (t-x) - \xi)\, d\xi + \int_{t-x}^{t} \widetilde{\delta}_2(\tau - (t-x), \tau)\, d\tau. \tag{A.1e}$$

The network variables are given by

$$z = L_{ind}(\dot{\mu}_3(t) - \dot{J}(t)) + \mu_4(t), \qquad I_L = \mu_3(t) - J(t). \tag{A.1f}$$

Here we have used the transformed source term $\widetilde{\delta}(x,t) = T^{-1}\delta(x,t)$ and the transformed initial values

$$\begin{bmatrix} w_0^1(x) \\ w_0^2(x) \end{bmatrix} = T^{-1} \begin{bmatrix} U(x,0) \\ I(x,0) \end{bmatrix}.$$

For system (2.1), (2.3) equation (A.1f) can be skipped, and (A.1e) is transformed to

$$w^2(x,t) = V(t-x) + \mu_1(t-x) + w_0^1(t-x)$$
$$+ \int_0^{t-x} \widetilde{\delta}_1(\xi, (t-x) - \xi)\, d\xi + \int_{t-x}^{t} \widetilde{\delta}_2(\tau - (t-x), \tau)\, d\tau.$$

Acknowledgments. The authors thank P. Rentrop and B. Simeon for valuable suggestions, as well as M. Arnold and S. Campbell for helpful discussions. We are indebted to the referees for helpful comments leading to an improved version of the paper.

REFERENCES

[1] M. Arnold, *A note on the uniform perturbation index*, Rostock. Math. Kolloq., 52 (1998), pp. 33–46.

[2] H. G. Brachtendorf, G. Welsch, and R. Laur, *Ein Simulationsmodell für verlustbehaftete Leitungen*, in 2.ITG-Diskussionssitzung "Neue Anwendungen Theoretischer Konzepte in der Elektrotechnik - mit Gedenksitzung zum 50. Todestag von Wilhelm Cauer," W. Mathis and P. Noll, eds., VDE-Verlag, Berlin, 1996, pp. 115–119.

[3] K. E. Brenan, S. L. Campbell, and L. R. Petzold, *Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations*, Classics Appl. Math. 14, SIAM, Philadelphia, 1995.

[4] S. L. Campbell and C. W. Gear, *The index of general nonlinear DAEs*, Numer. Math., 72 (1995), pp. 173–196.

[5] S. L. Campbell and W. Marszalek, *ODE/DAE integrators and MOL problems*, Z. Angew. Math. Mech. 76, Suppl. 1 (1996), pp. 251–254.

[6] S. L. Campbell and W. Marszalek, *The index of an infinite dimensional implicit system*, Math. Comput. Model. Dyn. Syst., 5 (1999), pp. 18–42.

[7] C. W. Gear, *Differential-algebraic equations, indices, and integral algebraic equations*, SIAM J. Numer. Anal., 27 (1990), pp. 1527–1534.

[8] M. Günther, *A joint DAE/PDE model for interconnected electrical networks*, Math. Comput. Model. Dyn. Syst., 6 (2000), pp. 114–128.

[9] M. Günther, *Semidiscretization may act like a deregularization*, Math. Comput. in Simulation, to appear.

[10] M. Günther and U. Feldmann, *CAD based electric circuit modeling in industry. Part* II: *Impact of circuit configurations and parameters*, Surv. Math. Indust., 8 (1999), pp. 131–157.

[11] E. Hairer, Ch. Lubich, and M. Roche, *The Numerical Solution of Differential-Algebraic Systems by Runge-Kutta Methods*, Springer-Verlag, Berlin, 1990.

[12] E. Hairer and G. Wanner, *Solving Ordinary Differential Equations* II. *Stiff and Differential-Algebraic Problems*, Springer-Verlag, Berlin, 1996.

[13] J. Heydweiller, R. Sincovec, and J. Fan, *Dynamic simulation of chemical processes described by distributed and lumped parameter models*, Comput. Chem. Engrg., 1 (1977), pp. 125–131.

[14] M. Hoschek, P. Rentrop, and Y. Wagner, *Network approach and differential-algebraic systems in technical application*, Surv. Math. Indust., 9 (1999), pp. 49–76.

[15] F. John, *Partial Differential Equations*, Springer-Verlag, New York, 1986.

[16] W. Lucht, K. Strehmel, and C. Eichler-Liebenow, *Indexes and special discretization methods for linear partial differential algebraic equations*, BIT, 39 (1999), pp. 484–512.

[17] W. S. Martinson and P. I. Barton, *A differentiation index for partial differential-algebraic equations*, SIAM J. Sci. Comput., 21 (2000), pp. 2295–2315.

[18] G. Massobrio and P. Antognetti, *Semiconductor Device Modeling with SPICE*, 2nd ed., McGraw-Hill, New York, 1993.

[19] T. Neumeyer and P. Rentrop, *The DAE aspect of the charge cycle in a combustion engine*, Appl. Numer. Math., 25 (1997), pp. 287–295.

[20] C. C. Pantelides, *The consistent initialization of differential-algebraic systems*, SIAM J. Sci. Statist. Comput., 9 (1988), pp. 213–231.

[21] A. Quarteroni and A. Valli, *Numerical Approximation of Partial Differential Equations*, Springer-Verlag, Berlin, Heidelberg, 1994.

[22] B. Simeon, *Modelling a flexible slider crank mechanism by a mixed system of DAEs and PDEs*, Math. Model. Systems, 2 (1996), pp. 1–18.

[23] Y. Wagner, *A moving grid algorithm for a heat exchanger with phase changes*, Appl. Numer. Math., 28 (1998), pp. 477–491.

SIAM J. SCI. COMPUT.
Vol. 22, No. 5, pp. 1630–1654
© 2000 Society for Industrial and Applied Mathematics

IMMERSED INTERFACE METHODS FOR NEUMANN AND RELATED PROBLEMS IN TWO AND THREE DIMENSIONS*

AARON L. FOGELSON† AND JAMES P. KEENER†

Abstract. We develop and apply a finite-difference method to discretize the Laplacian operator with Neumann boundary conditions on an irregular domain using a regular Cartesian grid. The method is an extension of the immersed interface method developed by LeVeque and Li [*SIAM J. Numer. Anal.*, 31 (1994) 1019–1044]. With careful selection of stencils, the method is second order accurate and produces a matrix that is stable (diagonally semidominant). The method is illustrated on several two-dimensional problems and one three-dimensional problem.

Key words. Neumann problems, immersed interface, Cartesian grid, stability

AMS subject classifications. 65M06, 65M12, 65N06, 65N12

PII. S1064827597327541

1. Introduction.

In this paper we discuss Cartesian-grid finite-difference methods for discretizing the Laplacian with Neumann boundary conditions specified on one or more curves or surfaces. The methods use special but compact stencils (i.e., 3 by 3 in two dimensions, 3 by 3 by 3 in three dimensions); they are second order accurate; and we give sufficient conditions that guarantee their stability. Even when these sufficient conditions are not met, the methods perform stably in our applications. We give a systematic procedure for selecting the stencils which can easily be implemented using look-up tables.

Cartesian-grid finite-difference methods are traditionally thought to be inappropriate for handling Neumann conditions on general curves or surfaces (see, e.g., [16]). The difficulty is this: on the one hand, first order one-sided difference approximations to the normal derivative combined with a standard five-point (in two dimensions) scheme for the Laplacian at interior points yields a stable but first order method [3]. On the other hand, traditional second order one-sided approximations to the normal derivative yield improved accuracy but at the cost of possible instability and a wide difference stencil [16]. The popularity of finite-element methods [4] and boundary-fitted grid methods [17] for such problems stems largely from their perceived superiority in handling this kind of boundary condition. But finite-element methods and boundary-fitted grid methods involve substantially more complicated data structures than do Cartesian-grid methods. The complicated data structures increase the complexity and cost of many parts of the solution process, especially the linear algebra, and complicate as well graphical display of the results. For these reasons, it is desirable to seek methods based on Cartesian grids that are easier to use.

The starting point for the derivation of the methods discussed here is the immersed interface method introduced by LeVeque and Li [7] for handling diffusion problems with discontinuous diffusion coefficients (see also [8, 11, 9, 10]). For our problems, the

*Received by the editors September 22, 1997; accepted for publication (in revised form) September 16, 1999; published electronically December 5, 2000.
http://www.siam.org/journals/sisc/22-5/32754.html

†Department of Mathematics, University of Utah, Salt Lake City, UT 84112 (fogelson@math.utah.edu, keener@math.utah.edu). The work of the first author was supported in part by NSF grants DMS-9307643 and DMS-9805518. The work of the second author was supported in part by NSF grant DMS-9626334.

physical domain is embedded in a larger rectangular domain, the partial differential equation(s) are extended to the rectangular domain, and a simple Cartesian grid is placed over the full domain. A standard discretization of the Laplacian is used at "regular" points of the grid for which the grid point and its immediate neighbors all lie on the same side of the boundary curves or surfaces. Special stencils are used at the remaining "irregular" points. In [7], LeVeque and Li use continuity and jump conditions at the interface across which the diffusion coefficient is discontinuous in order to simplify the truncation error that results when points on both sides of the interface are used to construct a discrete Laplacian. The equations for the stencil coefficients follow from this simplified expression. The present work is distinguished from that of LeVeque and Li by its use of Neumann boundary conditions instead of interface conditions, and by its careful exploration of the optimal choice of stencil to achieve stability as well as second order accuracy. We give detailed specifications of how to best choose a stencil in both two and three dimensions. These specifications are easily coded in terms of look-up tables making the method only slightly more expensive than standard schemes on rectangular grids with rectilinear boundaries aligned with the grid.

In a series of papers, Mayo and coworkers [12, 13, 14, 15] present an alternative approach to solving certain elliptic problems in irregular regions. It involves solving an integral equation on the domain boundary and using a standard discretization of the Laplacian on a Cartesian grid placed over a rectangle in which the irregular region is embedded. Later we briefly describe this approach, indicate how it is different from the method of this paper, and discuss some of the relative advantages and disadvantages of the two approaches.

The contents of the remainder of this paper are as follows: In section 2 we give a detailed derivation of the system of equations that the coefficients of the approximate two-dimensional Laplacian must satisfy at irregular points of the grid to achieve overall second order accuracy. In section 3, we discuss additional considerations in choosing the stencil to ensure stability, and arrive at a precise specification of how to make this choice. Section 4 concerns derivation of the coefficient equations for the three-dimensional Laplacian and stencil selections. In section 5, we apply the two-dimensional formulas to the heat equation in a number of contexts and numerically verify the stability and second order accuracy of the method. In section 6, we illustrate the three-dimensional method by solving the Poisson equation in a region bounded between an ellipsoid and a sphere.

As mentioned above, this project began as an effort to extend the ideas of LeVeque and Li's immersed interface method to handle Neumann conditions on irregular boundaries, and the derivation of our stencil equations is carried out in that spirit. But as we show in section 3, stable second order stencils often can be chosen using only grid points interior to the original physical domain. Hence, the method can be used as a purely interior-point method without reference to the grid points in the extended domain. Alternatively, it can be used in the spirit of the original derivation making use of grid points throughout the rectangular domain. Both approaches are illustrated below and we discuss why one approach or the other might be preferred in a given situation.

2. Derivation of the methods. We are interested in solving Poisson's equation

$$-\beta \Delta v(\mathbf{x}) \;=\; f(\mathbf{x}) \tag{2.1}$$

or the heat equation

$$v_t(\mathbf{x},t) = \beta\Delta v(\mathbf{x},t) + f(\mathbf{x},t) \tag{2.2}$$

in a two-dimensional region Ω^+. We assume v satisfies Neumann boundary condition

$$\beta\frac{\partial v}{\partial n} \;=\; g \tag{2.3}$$

at points of $\Gamma = \partial\Omega^+$. We adopt the convention throughout this paper that $\mathbf{n}$ is the unit normal vector which points *out* of the region Ω^+, so that $g > 0$ corresponds to a flux into Ω^+. We assume that Γ is at least piecewise smooth. We embed Ω^+ in a rectangular region Ω and define $\Omega^- = \Omega \setminus \overline{\Omega^+}$. We extend v to all of Ω by requiring that in Ω^-, v satisfy

$$-\beta\Delta v \;=\; f(\mathbf{x}) \tag{2.4}$$

in the case of Poisson's equation or

$$v_t = \beta\Delta v + f(\mathbf{x},t) \tag{2.5}$$

in the case of the heat equation. The extended source function $f(\mathbf{x})$ is assumed to be continuous in Ω. (If an analytic expression for f on the full rectangle is not available, then the extension of f to $(x,y) \in \Omega^-$ can be defined, for example, as the value of f at the point of Γ closest to (x,y) times a rapidly decaying function of the distance of (x,y) from Γ.) We require that v be continuous along Γ and that (2.3) be satisfied at points $(x,y) \in \Gamma$ when the derivatives are interpreted as one-sided ones involving values of v only on $\Omega^+ \cup \Gamma$. These conditions can be written as

$$[v] \;\equiv\; v^+ \;-\; v^- = 0 \tag{2.6}$$

and

$$\beta\frac{\partial v}{\partial n}^+ \;=\; g. \tag{2.7}$$

Here, and throughout this paper, the notation f^+ and f^- indicate limiting values of f at points of Γ with the limits taken entirely within Ω^+ or Ω^-, respectively. We also impose conditions on v or its derivatives on the boundary of the embedding rectangle Ω in order that v be uniquely determined (at least to within an additive constant) in all of Ω. We note that because of the continuity of v and the source term across Γ, the Laplacian of v is also continuous across Γ:

$$[\Delta v] \;\equiv\; (\Delta v)^+ \;-\; (\Delta v)^- = 0. \tag{2.8}$$

In this paper we describe a method for solving each of the above problems in two or three spatial dimensions. For now, we consider the problems in a *two-dimensional* region Ω. To determine an approximate solution to our problem, we place a uniform rectangular mesh with meshspace h over the region Ω and we seek to approximate (2.1) and (2.4) or (2.2) and (2.5) at each point of the mesh. We use a Crank–Nicolson time discretization in (2.2) and (2.5). We approximate the Laplacian in (2.1), (2.2), (2.4), or (2.5) by the standard five-point discrete Laplacian

$$\Delta^h u_{j_1,j_2} \;\equiv\; h^{-2}\left\{u_{j_1,j_2-1} + u_{j_1-1,j_2} + u_{j_1+1,j_2} + u_{j_1,j_2+1} - 4u_{j_1,j_2}\right\} \tag{2.9}$$

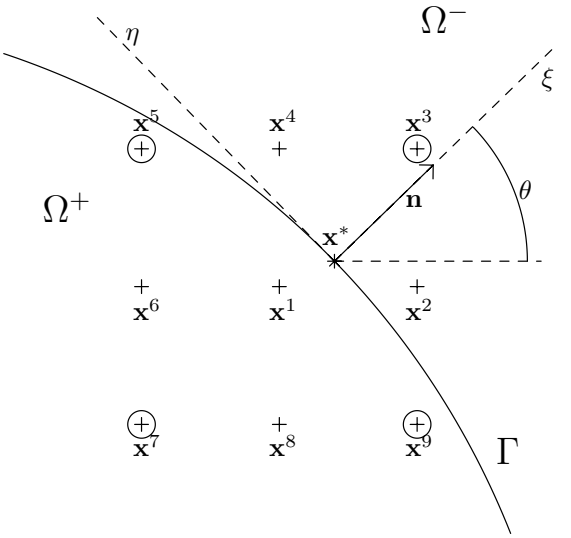


FIG. 2.1. *Layout near irregular point: Labeling of potential stencil points and definition of* (ξ, η) *coordinate system.*

at points $(j_1 h, j_2 h)$ for which all the points involved in this definition are on the same side of the "interface" Γ. Such points are called "regular points." This approximation to the Laplacian is, of course, second order.

Points of the mesh which are close to the interface and for which points of the standard stencil lie on both sides of the interface are called "irregular points." Our immediate goal is to derive a suitable approximation to the Laplacian at each irregular point. Because the size of the irregular region is roughly a dimension less than that of the domain, it suffices that the approximate Laplacian at such points be first order.

Consider an irregular point $\mathbf{x}^P = (j_1 h, j_2 h) \in \Omega^+$ and imagine that the interface Γ cuts through the mesh as indicated in Figure 2.1. (Point $\mathbf{x}^P$ is shown as $\mathbf{x}^1$ in the figure.) Let $\mathbf{x}^*$ be a point on the curve Γ close to $\mathbf{x}^P$ and such that the curve is smooth at $\mathbf{x}^*$. Since v is smooth in Ω^+, $\Delta v(\mathbf{x}^P) = \Delta v(\mathbf{x}^*)^+ + O(h)$, so it suffices to approximate $\Delta v(\mathbf{x}^*)^+$ to within $O(h)$. We approximate $\Delta v(\mathbf{x}^*)^+$ by a linear combination of values of v at the nine mesh points marked by a plus sign in Figure 2.1. Let K denote the set of indices of these nine points, $K^+ \subseteq K$ be the indices of those points which are in $\Omega^+ \cup \Gamma$, and $K^- \subseteq K$ be the indices of those points which are in Ω^-. Then we seek to determine a set of coefficients γ_k for $k \in K$ and a value C, so that $\sum_{k \in K} \gamma_k v(\mathbf{x}_k) - C$ is an $O(h)$ approximation to $\Delta v(\mathbf{x}^*)^+$. As is shown below, we generally need to have six nonzero γ_k in order to achieve an $O(h)$ approximation. In accordance with the size of the coefficients in the usual discretization (2.9) of the Laplacian, we expect that the magnitude of the nonzero γ_k is $O(h^{-2})$. The truncation error in using this approximation is

$$T^* = \Delta v(\mathbf{x}^*)^- - \sum_{k \in K^+} \gamma_k v(\mathbf{x}_k) - \sum_{k \in K^-} \gamma_k v(\mathbf{x}_k) + C. \tag{2.10}$$

For each $k \in K^+$, we expand $v(\mathbf{x}_k)$ in a Taylor series about $\mathbf{x}^*$ using limits of function

values from Ω^+. For each $k \in K^-$ we find a similar expansion using limits of function values from Ω^-. Before doing these expansions, it is useful to introduce a local change of coordinates near the point $\mathbf{x}^*$. As shown in Figure 2.1, θ is the angle between the positive x-direction and the normal $\mathbf{n}$ to the curve Γ pointing out of the region Ω^+. We introduce new orthogonal coordinates (ξ, η) by the equations

$$x - x^* = \xi \cos\theta - \eta \sin\theta, \tag{2.11}$$

$$y - y^* = \xi \sin\theta + \eta \cos\theta. \tag{2.12}$$

Note that $(\xi, \eta) = (0, 0)$ corresponds to the point $\mathbf{x}^*$. The curve Γ is assumed to be described locally near $\mathbf{x}^*$ by the relation $\xi = \chi(\eta)$. Note that $\chi(0) = 0$ and $\chi'(0) = 0$. We also write $\mathbf{x}_k = (\xi_k, \eta_k)$ to express the mesh points in terms of the new coordinates. In these new coordinates, we find Taylor series expansions for $k \in K^+$,

$$v(\mathbf{x}_k) = v^+ + v^+_\xi \xi_k + v^+_\eta \eta_k + \frac{1}{2} v^+_{\xi\xi} {\xi_k}^2 + v^+_{\xi\eta} \xi_k \eta_k + v^+_{\eta\eta} {\eta_k}^2 + O(h^3), \tag{2.13}$$

and for $k \in K^-$,

$$v(\mathbf{x}_k) = v^- + v^-_\xi \xi_k + v^-_\eta \eta_k + \frac{1}{2} v^-_{\xi\xi} {\xi_k}^2 + v^-_{\xi\eta} \xi_k \eta_k + v^-_{\eta\eta} {\eta_k}^2 + O(h^3). \tag{2.14}$$

We have truncated these expansions after the second order terms because this suffices if our expectation that $\gamma_k = O(h^{-2})$ is correct. When we substitute (2.13) and (2.14) into the expression (2.10) for T^*, we get an expression for T^* in terms of the twelve quantities $v^\pm$, $v^\pm_\xi$, $v^\pm_\eta$, $v^\pm_{\xi\xi}$, $v^\pm_{\xi\eta}$, and $v^\pm_{\eta\eta}$. We use the continuity conditions on v across Γ along with the given boundary conditions on Γ to express the Ω^+ quantities v^+, v^+_ξ, v^+_η, $v^+_{\xi\xi}$, $v^+_{\xi\eta}$, and $v^+_{\eta\eta}$ in terms of the Ω^- quantities v^-, v^-_ξ, v^-_η, $v^-_{\xi\xi}$, $v^-_{\xi\eta}$, and $v^-_{\eta\eta}$, and thus obtain an expression for T^* entirely in terms of Ω^- quantities. To this end, we express the interface conditions (2.6), (2.7), and (2.8) in terms of the local coordinate description of Γ:

$$v(\chi(\eta), \eta)^+ = v(\chi(\eta), \eta)^-, \tag{2.15}$$

$$v_\xi(\chi(\eta), \eta)^+ - \chi'(\eta) v_\eta(\chi(\eta), \eta)^+ = \frac{(1 + {\chi'}^2)^{1/2}}{\beta} g, \tag{2.16}$$

and

$$v_{\xi\xi}(\chi(\eta), \eta)^+ + v_{\eta\eta}(\chi(\eta), \eta)^+ = v_{\xi\xi}(\chi(\eta), \eta)^- + v_{\eta\eta}(\chi(\eta), \eta)^-. \tag{2.17}$$

In converting (2.7) to (2.16), we have used the fact that the outward unit normal to Γ is

$$\frac{1}{(1 + {\chi'}^2)^{1/2}} \begin{pmatrix} 1 \\ -\chi'(\eta) \end{pmatrix}.$$

We obtain additional relations between Ω^+ and Ω^- quantities by differentiating relations (2.15) and (2.16) with respect to η. Differentiating (2.15) with respect to η gives

$$(v_\xi \chi' + v_\eta)^+ = (v_\xi \chi' + v_\eta)^-. \tag{2.18}$$

Differentiating this again with respect to η yields

$$(v_{\xi\xi}\chi'^2 + 2v_{\xi\eta}\chi' + v_\xi\chi'' + v_{\eta\eta})^+ = (v_{\xi\xi}\chi'^2 + 2v_{\xi\eta}\chi' + v_\xi\chi'' + v_{\eta\eta})^-. \tag{2.19}$$

Differentiating (2.16) with respect to η gives

$$v^+_{\xi\xi}\chi' + v^+_{\xi\eta} - v^+_{\xi\eta}(\chi')^2 - v^+_{\eta\eta}\chi' - v^+_\eta\chi'' = \frac{(1 + \chi'^2)^{1/2}}{\beta}g' + \frac{\chi'\chi''}{\beta(1 + \chi'^2)^{1/2}}g. \tag{2.20}$$

We are interested in these relations at the point $\mathbf{x}^* = (\chi(0), 0)$. Setting $\eta = 0$ in (2.15)–(2.20), and recalling that $\chi(0) = 0$ and $\chi'(0) = 0$, we obtain the six relations

$$v^+ = v^-, \tag{2.21}$$

$$v^+_\xi = \frac{g}{\beta}, \tag{2.22}$$

$$v^+_\eta = v^-_\eta, \tag{2.23}$$

$$v^+_{\xi\xi} + v^+_{\eta\eta} = v^-_{\xi\xi} + v^-_{\eta\eta}, \tag{2.24}$$

$$v^+_{\xi\eta} - \chi''v^+_\eta = \frac{g'}{\beta}, \tag{2.25}$$

$$v^+_\xi\chi'' + v^+_{\eta\eta} = v^-_\xi\chi'' + v^-_{\eta\eta}. \tag{2.26}$$

Here, all quantities are evaluated at $\eta = 0$.

It is helpful to express relations (2.21)–(2.26) in a more compact form. We define V and $G \in \mathbb{R}^6$ by

$$V = (v,\ v_\xi,\ v_\eta,\ v_{\xi\xi}\ v_{\xi\eta}, v_{\eta\eta})^T \tag{2.27}$$

and

$$G = \left(0,\ \frac{g}{\beta},\ 0,\ 0,\ \frac{g'}{\beta},\ 0\right)^T. \tag{2.28}$$

Then (2.21)–(2.26) can be written as

$$A^+\,V^+ = A^-\,V^- + G, \tag{2.29}$$

where the 6×6 matrices A^+ and A^- are defined by

$$A^+ = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & -\chi'' & 0 & 1 & 0 \\ 0 & \chi'' & 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.30}$$

and

$$A^- = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \chi'' & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{2.31}$$

Since the matrix A^+ is nonsingular for any value of the curvature χ'', we can write

$$V^+ = (A^+)^{-1}A^-V^- + (A^+)^{-1}G. \tag{2.32}$$

We can now use this relation to express the truncation error defined by (2.10) in terms only of Ω^- quantities. In terms of the local (ξ, η) coordinate system, the truncation error at $\mathbf{x}^*$ is

$$T^* = v^+_{\xi\xi} + v^+_{\eta\eta} - \sum_{k\in K^+} \gamma_k v(\mathbf{x}_k) - \sum_{k\in K^-} \gamma_k v(\mathbf{x}_k) + C. \tag{2.33}$$

We let

$$Z_k = \left(1,\ \xi_k,\ \eta_k,\ \frac{1}{2}\xi_k^2,\ \xi_k\eta_k, \frac{1}{2}\eta_k^2\right)^T, \tag{2.34}$$

and

$$W = (0,\ 0,\ 0,\ 1,\ 0, 1)^T. \tag{2.35}$$

Then, using the Taylor series expansions (2.13) and (2.14) in (2.33), we get

$$T^* = W^TV^+ - \sum_{k\in K^+} \gamma_k Z_k^T V^+ - \sum_{k\in K^-} \gamma_k Z_k^T V^- + C + O(h^3)\max|\gamma_k|. \tag{2.36}$$

Since we anticipate that $|\gamma_k| = O(h^{-2})$, we expect that this last term is $O(h)$. Using (2.17) and (2.32), we can express T^* entirely in terms of Ω^- function values:

$$\begin{aligned} T^* = &\left\{ W^TV^- - \sum_{k\in K^+} \gamma_k Z_k^T \left[(A^+)^{-1}A^-V^-\right] - \sum_{k\in K^-} \gamma_k Z_k^T V^- \right\} \\ &+ \left(C - \sum_{k\in K^+} \gamma_k Z_k^T (A^+)^{-1}G \right) + O(h^3)\max|\gamma_k|. \end{aligned} \tag{2.37}$$

We want to choose the coefficients $\{\gamma_k\}$ so that the bracketed term in (2.37) vanishes for all vectors $V^- \in \mathbb{R}^6$ and to then choose C so that the term within ( ) vanishes. With these choices, $T^* = O(h)$. Clearly it suffices that the bracketed term vanish for each of the six standard basis vectors for $\mathbb{R}^6$. This gives us a system of six linear equations for the unknown γ_k, and so, in general, we need six nonzero γ_k, and therefore a six-point stencil, to obtain an $O(h)$ approximation. The six equations are

$$0 = \sum_{k\in K^+} \gamma_k + \sum_{k\in K^-} \gamma_k, \tag{2.38}$$

$$0 = \sum_{k\in K^+} \gamma_k \left\{ \frac{1}{2}\chi''({\eta_k}^2 - {\xi_k}^2) \right\} + \sum_{k\in K^-} \gamma_k \xi_k, \tag{2.39}$$

$$0 = \sum_{k\in K^+} \gamma_k \{\eta_k + \chi'' \xi_k \eta_k\} + \sum_{k\in K^-} \gamma_k \eta_k, \tag{2.40}$$

$$2 = \sum_{k\in K^+} \gamma_k {\xi_k}^2 + \sum_{k\in K^-} \gamma_k {\xi_k}^2, \tag{2.41}$$

$$0 = \sum_{k\in K^-} \gamma_k \xi_k \eta_k, \tag{2.42}$$

$$2 = \sum_{k\in K^+} \gamma_k {\eta_k}^2 + \sum_{k\in K^-} \gamma_k {\eta_k}^2. \tag{2.43}$$

In addition,

$$C = \sum_{k\in K^+} \gamma_k \{a_k + b_k\}, \tag{2.44}$$

where

$$a_k = \frac{1}{\beta}\, g\left[\xi_k - \frac{1}{2}\chi''({\eta_k}^2 - {\xi_k}^2)\right] \tag{2.45}$$

and

$$b_k = \frac{1}{\beta}\, g' \xi_k \eta_k. \tag{2.46}$$

This same approach can be used to handle Robin boundary conditions

$$\beta \frac{\partial v^+}{\partial n} + \alpha v^+ = g \qquad (\beta \neq 0) \tag{2.47}$$

instead of the Neumann conditions (2.7). In this case, the jump conditions can again be written in the form of (2.29). The matrix A^- and the vector G are the same as before, and the matrix A^+ is

$$A^+ = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ \frac{\alpha}{\beta} & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ \frac{\alpha}{\beta} & 0 & -\chi'' & 0 & 1 & 0 \\ 0 & \chi'' & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{2.48}$$

Since A^+ is again nonsingular, one can express V^+ in terms of V^- and therefore express the truncation error (2.36) entirely in terms of V^-. One obtains a system of six equations for the γ_k analogous to the system (2.38)–(2.43) above.

3. Stencil selection. We now face the issue of how to select the "best" stencil. While a criterion for "best" is not completely settled, here we seek a stencil whose coefficients satisfy the above equations and also satisfy the Gerschgorin sufficient conditions for stability. A matrix with entries a_{ij} satisfies the Gerschgorin stability conditions if

$$\sum_{j,j\neq i} |a_{ij}| \leq |a_{ii}| \tag{3.1}$$

and $a_{ii} \leq 0$. If this is the case, the eigenvalues of the matrix are guaranteed to lie in the left half of the complex plane, assuring only nonpositive real parts. It also follows in this case that the scheme satisfies a discrete maximum principle. In this problem we know from (2.38) that the row sum is zero, so the stability condition is that all noncentral γ's (i.e., other than the central point of the grid) must be positive. Thus, in what follows, we seek stencils for which the above equations hold and the noncentral γ's are positive.

Before we proceed further, we note that the grid points are numbered as

$$\begin{matrix} 5 & 4 & 3 \\ 6 & 1 & 2 \\ 7 & 8 & 9 \end{matrix} \tag{3.2}$$

so that the central point has label 1, the upper right corner is 3, and so on (see Figure 2.1). The Gerschgorin stability conditions can then be expressed

$$\gamma_1 < 0 \quad \text{and} \quad \gamma_{\mathrm{k}} \geq 0 \quad \text{for} \quad \mathrm{k} \neq 1. \tag{3.3}$$

Using the numbering just introduced, a six-point stencil is uniquely identified by five numbers (since the central point 1 is always assumed to be part of the stencil). For the remainder of this discussion, we define $\mathbf{x}^* = (x, y)$ to be the *closest* point on the boundary to the center of the stencil which we take to be at $(0, 0)$. We consider (x, y) in the unit square $0 \leq x \leq 1, 0 \leq y \leq 1$. (In essence we have set $h = 1$ for this analysis.)

We now make our first observation.

THEOREM 3.1. *With exactly one exterior point in the stencil, the system of equations* (2.38)–(2.43) *is singular.*

To show this, note that if we multiply (2.41) by $\frac{1}{2}\chi''$ and (2.43) by $-\frac{1}{2}\chi''$ and add both of these to (2.39), we find

$$0 \;=\; \sum_{k\in K^-} \gamma_k \left\{ \xi_k + \frac{1}{2}\chi''({\eta_k}^2 - {\xi_k}^2) \right\}, \tag{3.4}$$

which we use to replace (2.39). Now observe that both (2.41) and (3.4) have nonzero terms only for exterior grid points, and so, if there is only one exterior grid point, there are two equations with five zero coefficients, and the system of equations is singular.

The consequence of this is that if there is only one exterior point, one should use a standard five-point scheme, either the usual scheme 2468 (if 3, 5, 7, or 9 is missing) or the diagonal scheme 3579 (if 2, 4, 6, or 8 is missing).

Henceforth we assume that there are at least two exterior grid points, and to be specific we take them to be points 2 and 3. Now we make our second observation.

THEOREM 3.2. *With two exterior grid points and four interior grid points in the stencil, the coefficients for the exterior points are zero.*

This follows easily by observing that with two exterior grid points, (2.42) and (3.4) are a two by two system of linear equations which has only the trivial solution.

The problem is now simplified to looking at four-point interior schemes (which we call, simply, four-point schemes, and identify with three numbers, say, 568). There are twenty possible four-point interior schemes, giving a bewildering array of possibilities. However, we can immediately eliminate four of these schemes from further consideration, as follows.

THEOREM 3.3. *The four-point schemes with three adjacent points* (*schemes* 456, 567, 678, 789) *do not meet the Gerschgorin stability criterion for* χ'' *below some positive number.*

The proof of this follows from a direct calculation. For example, for the 678 scheme, one can calculate that

$$\gamma_7 = \frac{-2(x^2+y^2)^2 - (x+y)(x-y)^2 + 2\chi''(x^2+y^2)^{3/2}(2x^2+x+y+2y^2)}{xy(2x^2+x+y+2y^2) - \chi''(x^2+y^2)^{3/2}(1+2y)(1+2x)}. \tag{3.5}$$

It follows that for this scheme, γ_7 is negative in the range $0 < x \le 1,\ 0 < y \le 1$ if $\chi'' \le \frac{\sqrt{2}}{3}$. Similar statements can be made for the other schemes.

The primary consequence of this is that there are *no* acceptable four-point schemes when there are five grid points which are exterior to the domain. If there are five exterior points, then χ'' is likely to be negative. If, in fact, the boundary arc is described locally by the parabolic arc $\xi = \frac{1}{2}\chi''\eta^2$, then to have five exterior points χ'' must be negative. In this situation, therefore, if there are five exterior grid points, all possible four point schemes fail the Gerschgorin criterion. Thus, for the remainder of this discussion, we restrict our attention to a domain for which there are never more than four grid points that are exterior to the domain. With this further restriction we now have *only* sixteen schemes to evaluate. Later we discuss what to do if, in fact, a situation arises in which there are five exterior grid points.

We begin our search by looking at the region in which a particular scheme satisfies the Gerschgorin criterion. In this regard, the best of the sixteen schemes is the four-point scheme 579. For this scheme we calculate that

$$\gamma_5 = \frac{(x^2+y^2)^2 + 2x^2y - \chi''(x^2+y+y^2)(x^2+y^2)^{3/2}}{(x+y)D}, \tag{3.6}$$

$$\gamma_7 = \frac{2xy - \chi''(x^2+y^2)^{3/2}}{D}, \tag{3.7}$$

$$\gamma_9 = \frac{(x^2+y^2)^2 + 2xy^2 - \chi''(x^2+x+y^2)(x^2+y^2)^{3/2}}{(x+y)D}, \tag{3.8}$$

where $D = (x^2+y^2)(x+y) + 2xy - \chi''(x^2+y^2)^{3/2}(x+y+1)$. It is apparent that these three coefficients are positive if $\chi'' < 0$, suggesting that this is a good candidate for a useful scheme. Indeed it is a good scheme, but with limited usefulness. This limitation is because this scheme is admissible only when points 5 and 9 are both interior to the domain, a consideration that we address below.

A similar restriction applies to the second "obvious" good guess, 468. Here,

$$\gamma_6 = \frac{2(x^2-y^2) - 2\chi''(x^2+y^2)^{3/2}}{2x(x^2+y^2) + x^2 - y^2 - \chi''(x^2+y^2)^{3/2}(2x+1)} \tag{3.9}$$

with similar formulas for γ_4 and γ_8. Clearly, with $\chi'' = 0$, $\gamma_6 > 0$ provided $x > y$. In fact, all three coefficients are positive provided $x > y$ and $\chi'' \geq 0$. However, for any value of χ'', there are subregions of the square $0 < x < 1,\ 0 < y < 1$ for which the Gerschgorin criterion fails, so that this scheme is also not universally useful.

The disadvantage of the 579 and 468 schemes is that they are useful only if no more than three grid points are exterior to the domain. Schemes which are potentially more useful are those with a smaller footprint, including 457, 467, 568, 578, 679, and 689. (The symmetric partner of 457, which is 279, is not considered here because the point 2 is assumed to be exterior to the domain.) All of these schemes allow for four grid points exterior to the domain. One quickly determines that of these, 457 and 467 are essentially useless, as they satisfy the Gerschgorin criterion in only a small region of the unit square.

The schemes 568 and 578 have identical regions of the unit square in which the Gerschgorin criterion holds. The region is determined by $\gamma_6 > 0$, where γ_6 is given by

$$\gamma_6 = \frac{-2(x^2+y^2)^2 + x^3 + 5x^2y - xy^2 - y^3 + \chi''(x^2+y^2)^{3/2}(2x^2 - x - 1 - y + 2y^2)}{x(x+y)(2x^2+x+2y^2) - \chi''(x^2+y^2)^{3/2}(2x+1)(2x+1+2y)}. \tag{3.10}$$

The schemes 679 and 689 are the symmetric partners of 568 and 578 and so have symmetric regions in which they satisfy the Gerschgorin criterion.

At this point, we have identified a number of schemes that satisfy the Gerschgorin criterion in certain subregions of the unit square, but this does not help to decide if these schemes are useful without additional information about the location of the grid points relative to the boundary of the domain. Thus far, the number χ'' has been used solely as a parameter, but it also contains information about the boundary arc. If we now assume that the boundary arc is well approximated by the parabolic arc $\xi = \frac{1}{2}\chi''\eta^2$, then we can subdivide the unit square into regions in which each footprint is permitted. The boundary of these subregions are the 5 curves in the unit square for which the boundary arc $\xi = \frac{1}{2}\chi''\eta^2$ exactly intersects one of the grid points 2, 4, 5, 8, or 9. For example, the curve for gridpoint 4 is given by

$$2(x^2 - y + y^2)\sqrt{x^2+y^2} + \chi'' x^2 = 0, \tag{3.11}$$

and similar expressions are easily obtained for the other grid points. A plot of these five curves is shown in Figure 3.1(a) for $\chi'' = -0.3$, and Figure 3.1(b) shows similar plots for $\chi'' = 0.3$. The exact boundaries of the different regions shown change for different values of χ'', but the qualitative nature of the regions is as shown here. For example, to the right of the curve 4, the grid point 4 is exterior to the domain so a four-point stencil containing point 4 cannot be used.

Now that the unit square is divided into regions for which different footprints are possible, we must decide which of the possible schemes also satisfy the Gerschgorin criterion.

Figure 3.2 shows the further subdivisions of the unit square on which different schemes satisfy the Gerschgorin stability criterion. These regions are denoted by the number of possible schemes. For example, in the regions denoted 5a or 5b, there are 5 schemes that are permitted. In Table 3.1, the acceptable schemes for each of the subregions are listed, both for the case $\chi'' \leq 0$ and $\chi'' > 0$. The columns to the right of the double vertical line are acceptable only when $\chi'' > 0$. Note that schemes 459 and 489 have only an admissible range for $\chi'' > 0$. Region 0 is the region in which

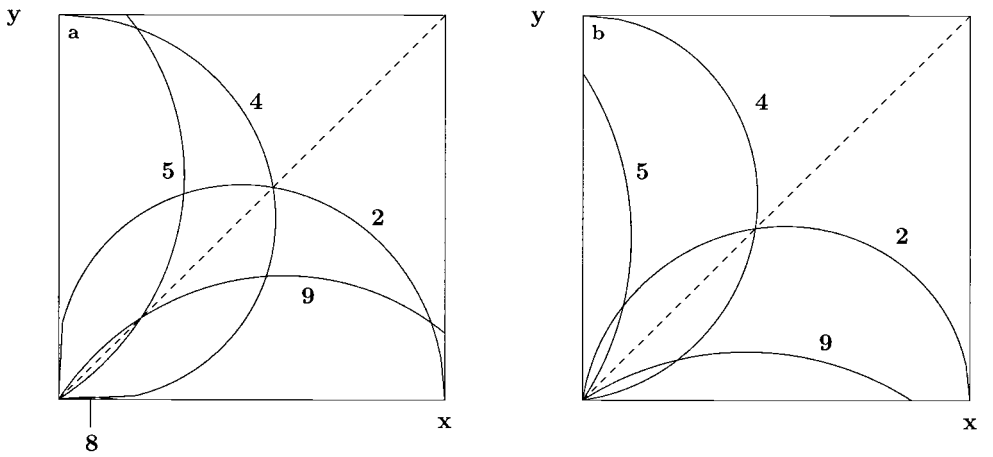


FIG. 3.1. *Geometric curves for cases* (a) $\chi'' = -0.3$ *and* (b) $\chi'' = 0.3$. *For each* $I \in \{2, 4, 5, 8, 9\}$, *curve* I *bounds the subregion of the unit square with the property that if the closest boundary point* (x, y) *is in that subregion, and if the boundary is described locally by the parabolic arc* $\xi = \frac{1}{2}\chi''\eta^2$, *then stencil point* I *is an interior point.*

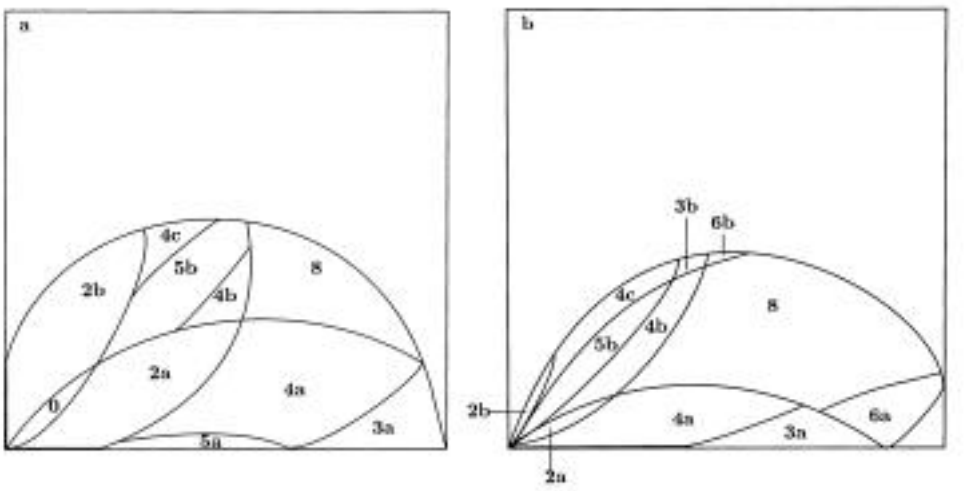


FIG. 3.2. *Gerschgorin regions and covering schemes for* (a) $\chi'' = -0.3$ *and* (b) $\chi'' = 0.3$. *The region labels correspond to those in Table* 3.1. *In the unlabeled regions, a standard five-point scheme is acceptable.*

there are five exterior grid points and for which, therefore, there are no admissible four-point schemes. This region exists only if $\chi'' < 0$.

The challenge is to find the smallest set of admissible schemes that covers the domain. Scanning through the table, one sees that for $\chi'' \leq 0$, three schemes are necessary for this purpose. For example, the three schemes 568, 689, and 468 cover the domain. This is not a unique covering set, as 578 and 568 can be freely interchanged, as can 679 and 689. This covering set is valid for $\chi'' \leq 0$ but not for $\chi'' > 0$. In Table 3.2 are listed the minimal covering sets as a function of χ''.

TABLE 3.1
Acceptable schemes for each subregion of Figure 3.2. *Regions to the right of double vertical line occur only if* $\chi'' > 0$.

Scheme	2a	2b	3a	4a	4b	4c	5a	5b	8	3b	6a	6b
457							x					
458			x								x	
459											x	
467							x					
468			x	x			x		x		x	
469									x		x	x
478			x	x					x		x	
479									x		x	x
489												x
568	x			x	x		x	x	x			
569						x			x	x		x
578	x			x	x		x	x	x			
579					x	x		x	x	x		x
589						x				x		x
679		x				x		x				
689		x				x		x				

TABLE 3.2
Minimal covering sets.

$\chi'' \leq 0$	$0 < \chi'' < .5$	$\chi'' > .5$
568	568	
689	689	
468	468	468
	579	579

Provided there are no more than four exterior points, an easy computational strategy for determining a scheme that satisfies the Gerschgorin stability criterion is evident. First, if $\chi'' \leq 0$, try the scheme 568 (as that covers the most area), and if this fails try 468 if point 4 is in the domain interior, or 689 if point 9 is in the domain interior. If $\chi'' > 0$, first, if point 9 is in the domain interior, try 579 as this covers the most area. If this fails, then 468 is most likely to succeed. The schemes 568 and 689 cover small regions and so are less likely to be needed, but are necessary to produce a complete cover. Note that there are two ways to reject a scheme when pursuing this strategy. First, it may not be possible to find a solution to the linear system (2.38)–(2.43) for a given stencil. Second, the solution may not satisfy the Gerschgorin stability condition. In any case, one of the above choices of stencils is guaranteed to lead to a solvable system whose solution does satisfy the Gerschgorin condition.

It remains to address the question of what to do if an interior irregular point occurs with 5 exterior points in its stencil. When this happens, the above discussion does not apply, and furthermore, we have found no schemes with three interior points that satisfy the Gerschgorin condition. The two canonical situations in which such an irregular point arises are illustrated in Figure 3.3. In part (a), the boundary curve has high local curvature and the mesh is not fine enough to resolve this curvature. A finer mesh should be used. In part (b), the curvature is small and yet there are 5 exterior points. One way to deal with this situation is to shift slightly the entire mesh relative to the domain of interest. Another possibility, discussed below in the context of solution of the heat equation, is to make use of both interior and exterior points

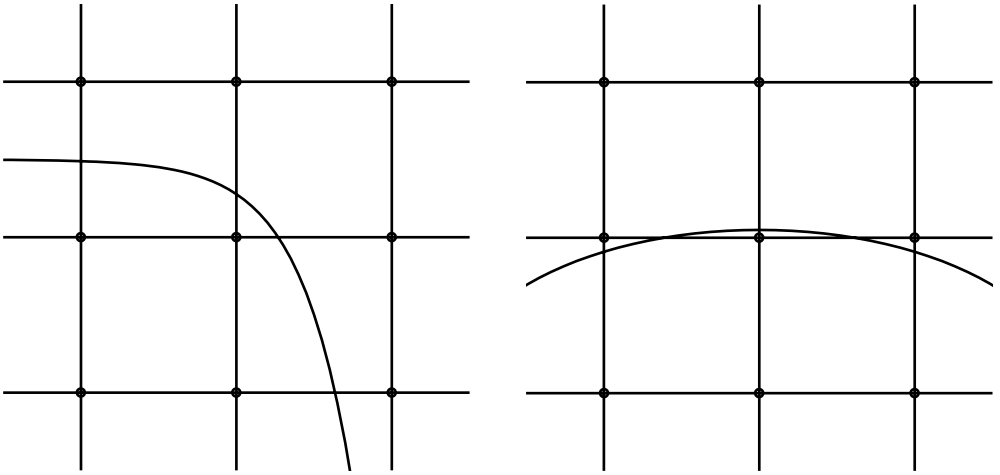


FIG. 3.3. *Situations in which there are five exterior points.*

for the stencil.

4. Three-dimensional problems. The extension of the above ideas to three-dimensional problems is straightforward, although there are a number of unresolved issues.

The derivation of the coefficient equations for the approximation to the Laplacian at irregular points follows the same arguments used in the two-dimensional case. We determine the point $\mathbf{x}^*$ on the boundary about which to make the local expansion. We introduce a new local orthogonal coordinate system (ξ, η_1, η_2), taking $(\xi, \eta_1, \eta_2) = (0, 0, 0)$ to correspond to the point $\mathbf{x}^*$. The surface is assumed to be described locally by the relationship $\xi = \chi(\eta_1, \eta_2)$. We denote $\frac{\partial \chi}{\partial \eta_j} = \chi_j$ and $\frac{\partial^2 \chi}{\partial \eta_j \partial \eta_k} = \chi_{jk}$ for $j = 1, 2, k = 1, 2$. We suppose that the coordinate system is chosen to be the principal coordinate system for the surface so that $\chi(0,0) = \chi_1(0,0) = \chi_2(0,0) = \chi_{12}(0,0) = 0$. The numbers $\chi_{11}(0,0)$ and $\chi_{22}(0,0)$ are the principal curvatures and the surface has the local representation $\xi = \frac{1}{2}\chi_{11}(0,0)\eta_1^2 + \frac{1}{2}\chi_{22}(0,0)\eta_2^2$.

The quantities that must be continuous across the boundary are v; $v_\xi \chi_j + v_{\eta_j}$ for $j = 1, 2$; $v_{\xi\xi}\chi_j\chi_k + v_{\xi\eta_k}\chi_j + v_\xi \chi_{jk} + v_{\xi\eta_j}\chi_k + v_{\eta_j\eta_k}$ for $(j,k) = (1,1), (1,2), (2,2)$; and $(v_{\xi\xi} + v_{\eta_1\eta_1} + v_{\eta_2\eta_2})$. The Neumann boundary condition (2.3) contributes the three equations

$$(v_\xi - v_{\eta_1}\chi_1 - v_{\eta_2}\chi_2)^+ = \frac{(1 + \chi_1^2 + \chi_2^2)^{1/2}\, g}{\beta}, \tag{4.1}$$

(4.2)

$$\begin{aligned}\big(v_{\xi\xi}\chi_j + v_{\xi\eta_j} - v_{\xi\eta_1}\chi_1\chi_j - v_{\eta_1\eta_2}\chi_1 - v_{\eta_1}\chi_{1j} - v_{\xi\eta_2}\chi_2\chi_j - v_{\eta_2\eta_j}\chi_2 - v_{\eta_2}\chi_{2j}\big)^+ \\ = \frac{\partial}{\partial \eta_j}\left((1 + \chi_1^2 + \chi_2^2)^{1/2}\, \frac{g}{\beta}\right)\end{aligned}$$

for $j = 1, 2$, which gives us a total of ten conditions relating limiting values at $\mathbf{x}^*$ from Ω^+ to those from Ω^-. We define the vector

$$V = (v, v_\xi, v_{\eta_1}, v_{\eta_2}, v_{\xi\xi}, v_{\xi\eta_1}, v_{\eta_1\eta_1}, v_{\xi\eta_2}, v_{\eta_2\eta_2}, v_{\eta_1\eta_2})^T$$

and write the resulting system of equations in matrix form as

$$A^+V^+ = A^-V^- + G, \tag{4.3}$$

where

$$A^+ = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \chi_{11} & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & \chi_{22} & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -\chi_{11} & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -\chi_{22} & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}. \tag{4.4}$$

The first seven rows of A^- are identical to those of A^+, and the last three rows of A^- are all zero. It is not difficult to determine that

$$(A^+)^{-1}A^- = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -\chi_{11}-\chi_{22} & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \chi_{11} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \chi_{11} & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \chi_{22} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \chi_{22} & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{4.5}$$

Finally, with

$$W = (0,0,0,0,1,0,1,0,1,0)^T, \qquad Z_k = (1,\xi,\eta_1,\eta_2,\frac{1}{2}\xi^2,\xi\eta_1,\frac{1}{2}\eta_1^2,\xi\eta_2,\frac{1}{2}\eta_2^2,\eta_1\eta_2)_k^T,$$

we require that

$$W = \sum_{k\in K^+} \gamma_k \left[(A^+)^{-1}A^-\right]^T Z_k + \sum_{k\in K^-} \gamma_k Z_k \tag{4.6}$$

as our system of equations for determining the coefficients γ_k. These equations simplify somewhat if we premultiply (4.6) with the matrix

$$T = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & \chi_{11}+\chi_{22} & 0 & -\chi_{11} & 0 & -\chi_{22} & 0 \end{bmatrix}, \tag{4.7}$$

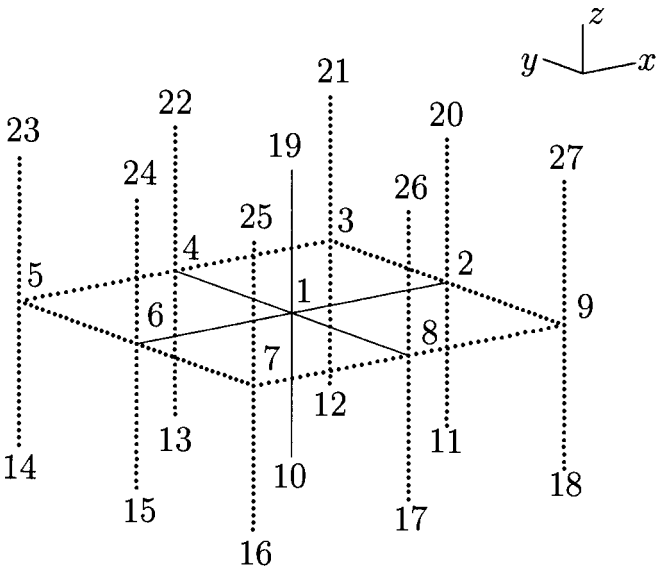


FIG. 4.1. *Labeling of potential stencil points for three-dimensional calculations.*

and then the equations take the form

$$TW = \sum_{k \in K^+} \gamma_k BZ_k \; + \; \sum_{k \in K^-} \gamma_k TZ_k, \tag{4.8}$$

where $BZ = (1, \eta_1 + \xi\eta_1\chi_{11}, \eta_2 + \xi\eta_2\chi_{22}, \frac{1}{2}\xi^2, \frac{1}{2}\eta_1^2, \frac{1}{2}\eta_2^2, \eta_1\eta_2, 0, 0, 0)^T$, $TZ = (1, \eta_1, \eta_2, \frac{1}{2}\eta_1^2, \frac{1}{2}\eta_2^2, \eta_1\eta_2, \xi\eta_1, \xi\eta_2, \xi + \frac{1}{2}\chi_{11}(\xi^2 - \eta_1^2) + \frac{1}{2}\chi_{22}(\xi^2 - \eta_2^2))^T$, and $TW = (0, 0, 0, 2, 2, 2, 0, 0, 0, 0)^T$.

There are now a number of simple consequences of this calculation, analogous to the two-dimensional case.

THEOREM 4.1. *With less that three exterior points, this system of equations is singular.*

With less than three exterior points, there is certain to be a "standard" seven-point scheme that can, and should, be used.

THEOREM 4.2. *Any scheme with exactly seven interior points has* $\gamma_k = 0$ *for the three exterior points.*

Now we face the difficult issue of choosing three-dimensional stencils that satisfy the Gerschgorin stability criterion. We assume that there are at least three exterior grid points and we seek seven-point stencils consisting entirely of interior points. We use a nomenclature similar to the two-dimensional nomenclature to identify the grid points with a number between 1 and 27. This grid nomenclature, depicted in Figure 4.1, assigns the numbers 1 through 9 to the points in the $z = 0$ plane exactly as in the two-dimensional case. In the plane $z = -h$, add 9 to the corresponding point at $z = 0$ and in the plane $z = h$ add 18 to the corresponding point at $z = 0$.

Next we simplify the problem slightly by assuming that the boundary surface is locally planar, having $\chi_{11} = \chi_{22} = 0$. Notice that in the limit of small grid size, the weights γ_j are independent of curvature, so there is no loss of accuracy in the limit $h \to 0$.

We assume that the boundary surface passes through the first octant (so that point 21 is certain to be an exterior point). Using a numerical search, we have found that all (admissible) points in the first octant can be covered by 45 seven-point stencils. These schemes are generated by the following 8 stencils:

1. 2 4 5 7 8 10 (6),
2. 6 8 9 10 11 13 (6),
3. 5 6 8 9 10 12 (3),
4. 5 7 9 10 11 26 (6),
5. 6 7 8 9 10 13 (6),
6. 2 6 7 8 10 13 (6),
7. 6 7 9 10 13 15 (6),
8. 6 7 8 9 13 18 (6).

Each of these stencils is used to generate additional stencils by reflections in the planes $x = y$, $x = z$, and $y = z$. The number in parentheses following the stencil indicates the total number of stencils generated by this group action. Seven of the stencils generate six separate stencils and one generates only three stencils because it is symmetric about the plane $x = y$.

With these stencils it is possible to find a scheme that satisfies the Gerschgorin stability criterion for any planar boundary surface that intersects the first octant. While this list of stencils covers the range of possibilities, it is not known if this is the minimal list.

5. Two-dimensional heat equation. We apply the formulas derived in section 2 to the two-dimensional heat equation in a variety of domains Ω^+ each of which is contained within the unit square Ω. For each application, the boundary $\Gamma = \partial\Omega^+$ consists of the union of one or more circles, and we assume that Neumann boundary conditions hold along each circle. These calculations were done before the analysis of sufficient conditions for obtaining Gerschgorin stability was complete, and so a somewhat different strategy for choosing the stencil was used. This strategy usually, but not always, led to our choosing a stencil which satisfies the Gerschgorin condition. The occasional failure to satisfy this condition seems to have had no detrimental effects. This suggests that the Gerschgorin condition is not necessary for stability, at least in this context.

We discretize time into steps of size k and place a uniform spatial grid with meshspace h over Ω. We denote by $u^n_{j_1,j_2}$ the numerical solution at grid point $\mathbf{x}_{j_1,j_2} = (j_1h, j_2h)$ at time $t_n = nk$, and we denote by $\mathbf{u}^n$ the vector of grid values of the numerical solution ordered in some specified way (typically row-by-row ordering is used). We use the Crank–Nicolson time-discretization, and so, in each timestep, we have a linear system of the form

$$\left(I - \frac{\beta k}{2}A\right)\mathbf{u}^{n+1} \;=\; \left(I + \frac{\beta k}{2}A\right)\mathbf{u}^n \;+\; \mathbf{C}, \tag{5.1}$$

where the coefficient matrix A is an approximation to the Laplacian determined by the approach of section 2. Thus, if $\mathbf{x}_{j_1,j_2}$ is a regular point, the corresponding row of A consists of the coefficients from the standard five-point approximation to the Laplacian, while if $\mathbf{x}_{j_1,j_2}$ is an irregular point, the corresponding row of A typically contains six nonzero coefficients determined by solving (2.38)–(2.43) and by defining a correction term C_{j_1,j_2} if necessary. In (5.1), the vector $\mathbf{C}$ contains these correction terms in the same order as the unknowns $u^n_{j_1,j_2}$ appear in $\mathbf{u}^n$. Recall that the correction term is nonzero only in the case of inhomogeneous Neumann conditions, and is easily calcu-

lated from (2.44) once the corresponding coefficients γ_k have been determined. Since the main issue is computing the γ_k, we assume for now that homogeneous Neumann conditions hold, and therefore we drop reference to $\mathbf{C}$.

The formulas derived in section 2 for determining the coefficients γ_k ensure that the local truncation error at each irregular point be $O(h)$. Because there are many possible choices of which six points of the nine-point stencil to use, there is still a lot of freedom left in the choice of γ_k. For these calculations we set satisfying the Gerschgorin stability condition as a goal in choosing the stencil points. Since we are using the Crank–Nicolson time-discretization, achieving this goal would ensure the stability of our calculations. In our attempt to satisfy the Gerschgorin condition (3.3), we adopted the following rule for choosing the γ_k: For the six stencil points, we use the five points of the standard stencil numbered as in Figure 2.1, and one of the corner points (circled in the figure) of the nine-point stencil. We consider each of the corner points in turn, rejecting a corner point if the corresponding system of six equations (2.38)–(2.43) for the γ_k is not solvable, or if the resulting γ_k fail to give a diagonally-dominant matrix A. We accept as the sixth point the first corner point for which the diagonal dominance condition does hold, and we use the corresponding γ_k in A.

In all of our numerical tests, we have not encountered a case where all four choices of corner points result in unsolvable systems, and we find only very rarely that all four sets of potential γ_k calculated from the corner points fail to satisfy the sufficient condition for stability. In these cases, use of one of these sets of γ_k does not cause any apparent instability.

To test the accuracy of our method, we construct a problem for which an exact analytic solution is available. Let $r_0 < 1/2 \le r_1$, and consider the radially symmetric heat equation

$$v_t(r,t) \;=\; \beta \frac{1}{r}(r v_r(r,t))_r \tag{5.2}$$

in the annulus $r_0 < r < r_1$ for $t > 0$, with $v(r_1,t) = 0$ and $v_r(r_0,t) = 0$ for $t > 0$, and $v(r,0) = b(r)$ for $r_0 < r < r_1$. This problem has the solution

$$v^{ex}(r,t) = \sum_{m=1}^{p} A_m \left\{ J_0(\lambda_m r) Y_0(\lambda_m r_1) - J_0(\lambda_m r_1) Y_0(\lambda_m r) \right\} e^{-{\lambda_m}^2 \beta t}, \tag{5.3}$$

provided that the initial function $b(r)$ is given by

$$b(r) = \sum_{m=1}^{p} A_m \left\{ J_0(\lambda_m r) Y_0(\lambda_m r_1) - J_0(\lambda_m r_1) Y_0(\lambda_m r) \right\}. \tag{5.4}$$

Here, J_0 and Y_0 are Bessel functions of order 0, and λ_m satisfies $J_1(\lambda r_0) Y_0(\lambda r_1) - J_0(\lambda r_1) Y_1(\lambda r_0) \;=\; 0$. We measure r from the center of the unit square $\Omega = [0,1] \times [0,1]$; we let $\Omega^+ = \Omega \cap \{r :\; r_0 < r\}$, $\Omega^- = \Omega \setminus \overline{\Omega^+}$, $\Gamma = \{r :\; r = r_0\}$; and we use our numerical method to solve the problem

$$v_t(\mathbf{x},t) \;=\; \beta \Delta v(\mathbf{x},t) \tag{5.5}$$

for $\mathbf{x} \in \Omega^+$ and $t > 0$, with boundary conditions $\frac{\partial v}{\partial n}(\mathbf{x},t) \;=\; 0$ for $\mathbf{x} \in \Gamma$ and $t > 0$, $v(\mathbf{x},t) \;=\; v^{ex}(\mathbf{x},t)$ for $\mathbf{x} \in \partial\Omega$ and $t > 0$, and initial condition $v(\mathbf{x},0) \;=\; b(r)$ for $\mathbf{x} \in \Omega^+$. To provide initial data for the numerical solution in all of Ω, we extend $b(r)$

TABLE 5.1
Errors for test problem 1.

h	t	$\| \ \|_\infty$	$\| \ \|_1$	$\| \ \|_2$
0.02	0.04	0.637E-04	0.246E-04	0.3298E-04
0.02	0.08	0.513E-04	0.204E-04	0.271E-04
0.02	0.12	0.367E-04	0.158E-04	0.199E-04
0.02	0.16	0.260E-04	0.129E-04	0.157E-04
0.02	0.20	0.204E-04	0.108E-04	0.133E-04
0.01	0.04	0.226E-04	0.654E-05	0.853E-05
0.01	0.08	0.131E-04	0.497E-05	0.650E-05
0.01	0.12	0.926E-05	0.317E-05	0.424E-05
0.01	0.16	0.624E-05	0.231E-05	0.294E-05
0.01	0.20	0.441E-05	0.188E-05	0.232E-05

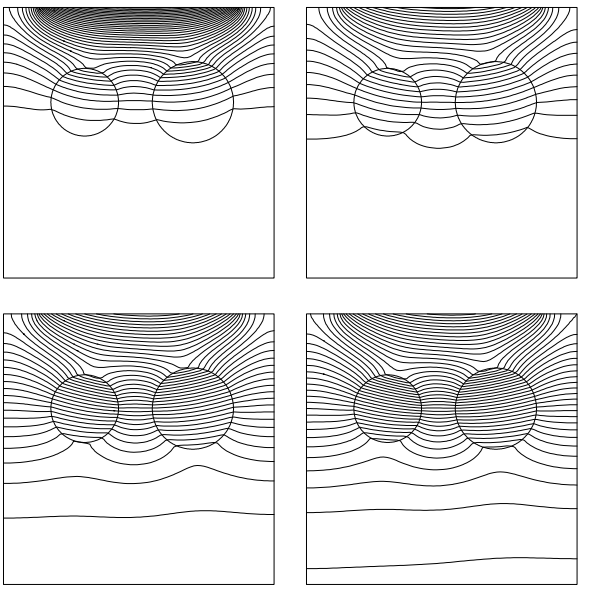


FIG. 5.1. *Contour plots of solution to heat equation at selected times. Homogeneous Neumann conditions along circles. Heat influx along top domain boundary.*

to the disc $\Omega^- = \{r < r_0\}$ as the constant value $b(r_0)$. We used $r_0 = 0.25$, $r_1 = 1.25$, and $p = 5$ in (5.3)–(5.4) and performed numerical experiments with stepsizes $h = 0.02$ or $h = 0.01$ and timesteps $k = 0.1h$. The results, given in Table 5.1, involve various error norms computed using grid points in Ω^+ and clearly show that the method is second order accurate for this problem.

The second problem we consider involves two disjoint circles contained within the unit square Ω (see Figure 5.1). The region of interest Ω^+ is the portion of the unit square outside of the two circles. The union of the interiors of the two circles is Ω^-, and Γ is the union of the circles. We are interested in solving the heat equation in Ω^+ with no-flux boundary conditions on both circles and along the left, right, and bottom boundaries of Ω and a prescribed nonzero flux along the top boundary of Ω. This flux is constant over much of the top boundary, and drops smoothly to zero as the top corners are approached. In Figure 5.1, we show contour plots at selected times in a computation performed on an 80×80 grid. Observe that the contour lines intersect

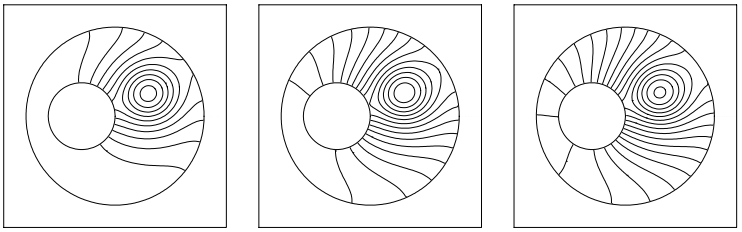


FIG. 5.2. *Contour plots of solution to heat equation at selected times. Homogeneous Neumann conditions on circles. Source between circles.*

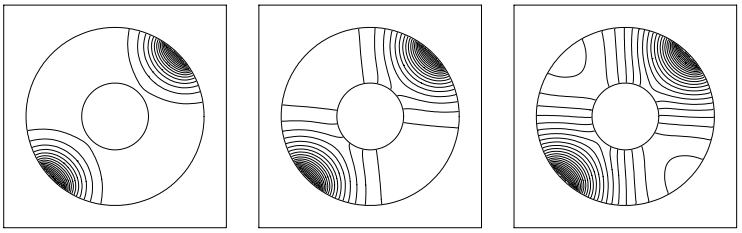


FIG. 5.3. *Contour plots of solution to heat equation at selected times. Inhomogeneous Neumann conditions on circles.*

the circles orthogonally. The contours inside the circle are physically meaningless; only those in Ω^+ are of interest. We compared the discrete integral of the solution over the region Ω^+ with the time integral of the flux and found excellent agreement.

The third problem involves heat flow in a region Ω^+ between two circles, so the two parts of Γ have curvatures of different sign. No-flux boundary conditions are imposed on both circles and there is a steady source term with circular support centered to the right and above the inner circle. Figure 5.2 shows contour plots from the solution of this problem using a 128×128 grid on the unit square. For clarity we show only contour lines within Ω^+. The boundary of the unit square is outside the region of interest and so the boundary conditions there have no physical meaning. For these calculations we applied homogeneous Dirichlet conditions along these boundaries. Again we see that the solution contours intersect the circles orthogonally.

In Figure 5.3, we show contour plots from a problem with inhomogeneous Neumann conditions on portions of Γ. Again the physical domain Ω^+ is the region between two circles. On the inner circle, homogeneous Neumann conditions are applied. On the outer circle, inhomogeneous Neumann conditions (2.3) hold with

$$\begin{aligned} g(s) \;=\; & 5 * \left\{ \tanh\left(100 r_2 \left(\Theta - \frac{3\pi}{16}\right)\right) - \tanh\left(100 r_2 \left(\Theta - \frac{5\pi}{16}\right)\right)\right\} \\ & + \; 5 * \left\{ \tanh\left(100 r_2 \left(\Theta - \frac{19\pi}{16}\right)\right) - \tanh\left(100 r_2 \left(\Theta - \frac{21\pi}{16}\right)\right)\right\}, \end{aligned}$$

where $r_2 = 0.4$ is the radius of the outer circle and $s = r_2\Theta$ is arclength. So the flux is approximately 5 over the angular intervals $(\frac{3\pi}{16}, \frac{5\pi}{16})$ and $(\frac{19\pi}{16}, \frac{21\pi}{16})$ and 0 elsewhere. The correction term C_{j_1,j_2} given by (2.44) is nonzero only for irregular points adjacent to these portions of Γ. Convergence studies for the problems in Figures 5.2 and 5.3 show second order convergence.

The last example in this section involves solution of the heat equation in the region shown in Figure 5.4. Again Ω is the unit square and this time periodic boundary

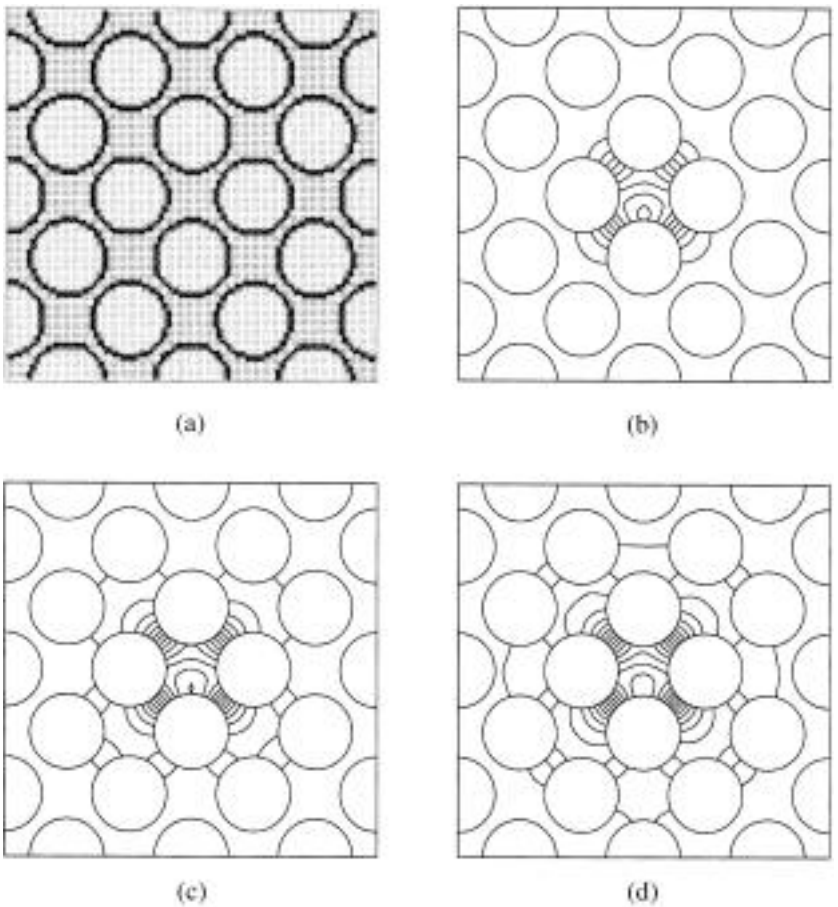


FIG. 5.4. (a) *Locations of irregular points.* (b)–(d) *Contour plots of solution to heat equation at selected times. Homogeneous Neumann conditions on circles. Source near domain center.*

conditions are imposed along its edges. The physical domain is that part of Ω exterior to all of the circles shown. Homogeneous Neumann conditions are imposed on each of the circles, and there is a source term located just below the center of the domain. Figure 5.4(a) shows the distribution of regular and irregular points for a 100×100 grid. In Figure 5.4(b)–(d), we show contours of the solution as it evolves and heat diffuses from the source through the gaps between the circular obstacles. In Figure 5.5, we show a perspective plot of the solution at a particular time. We have postprocessed the solution by setting its value to zero at points outside the physical domain.

To carry out a convergence study of our method applied to this problem, we performed calculations on grids with $h = 0.02$, 0.01, 0.005, and 0.0025, and with timestep $k = 0.1h$. In Table 5.2, we show the relative differences between each of the 0.02, 0.01, and 0.005 computations and the 0.0025 computation. There are several aspects of these results that should be noted. There is only a factor of 2 improvement between the $h = 0.02$ and $h = 0.01$ results, but an almost order of magnitude improvement between the $h = 0.01$ and $h = 0.005$ results. The first probably reflects the fact that with $h = 0.02$ and even $h = 0.01$, a relatively large part of the grid consists of irregular points. In fact, in the $h = 0.02$ version of Figure

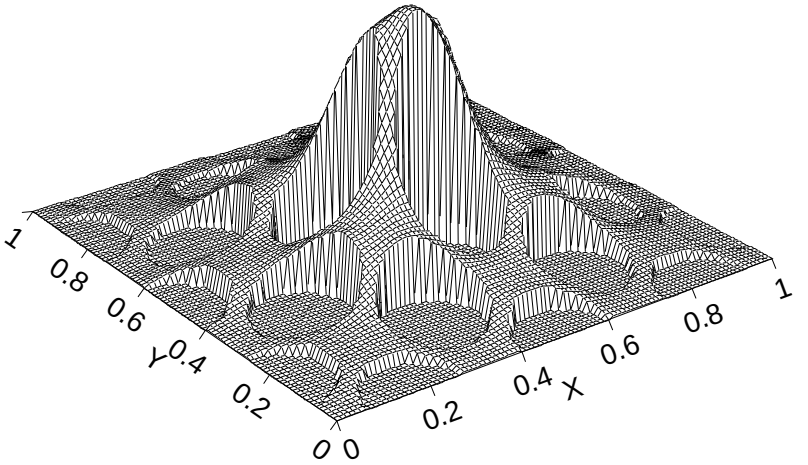


FIG. 5.5. *Perspective plot of solution to heat equation at same time as in* (d) *in preceding figure.*

TABLE 5.2
Errors for problem with many circles.

h	$\|\ \|_\infty$	$\|\ \|_1$	$\|\ \|_2$
0.02	0.238E-01	0.173E-01	0.166E-01
0.01	0.993E-02	0.803E-02	0.787E-02
0.005	0.130E-02	0.822E-03	0.820E-03

5.4(a) (not shown), there are no regular points in the narrow spaces between adjacent circles. Yet with a further reduction of h to 0.005, the results seem to "catch-up," so there is about a factor of 16 reduction in error from $h = 0.02$ to $h = 0.005$. We also think that it is noteworthy that even with $h = 0.02$, the errors were only 1–2%. We think these results suggest that this method may be quite useful for studying diffusion in porous media.

6. Three-dimensional Poisson equation. We apply the formulas described in section 4 to solve the Poisson equation in the region shown in Figure 6.1(a). For this problem the region Ω^+ is bounded on the outside by a sphere and on the inside by an ellipsoid. Homogeneous Neumann conditions hold on both of these surfaces. A source is distributed over a region of Ω^+ above the ellipsoid, and a sink of equal magnitude is distributed over the symmetric region below the ellipsoid. In Figure 6.1(b) are shown iso-surfaces of the solution obtained on a 64^3 grid with equal spacing in all coordinate directions. While the problem set-up is symmetric about an axis through the ellipsoid and sphere centers, the numerical method was not modified in any way to exploit this symmetry.

7. Conclusions. We have shown that Cartesian-grid finite-difference methods can be used very effectively to solve elliptic and parabolic equations with Neumann boundary conditions on general boundary curves or surfaces. The methods we discuss involve modifications to the finite-difference stencil at points near the boundary, and we described the results of a careful exploration of how these modifications should be made in order to achieve second order accuracy and stability. While the exploration itself was tedious, the results of it are easy to use in that we have specified exactly which choices of stencil work in each of a number of circumstances. This can be

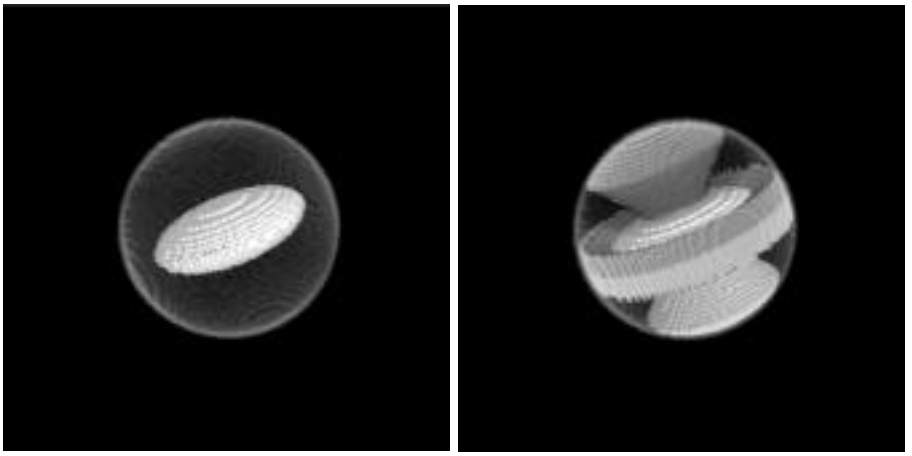


FIG. 6.1. (a) *Domain for three-dimensional calculation is bounded by sphere and ellipsoid.* (b) *Level surfaces of solution of three-dimensional Poisson equation.*

implemented simply in terms of look-up tables. Our approach works for both two-dimensional and three-dimensional problems as we illustrated through a variety of numerical examples.

One issue that we are still working on concerns iterative solution of the discrete system. For the two-dimensional heat-equation applications, line SOR or a very simple multigrid method was used. Even though the restriction and prolongation steps of this multigrid method did not take the boundary into account, the multigrid method was several times faster than line SOR. Perhaps incorporating information about the boundary into these steps, as in [1], will further improve the rate of convergence. Another issue that arises when a Galerkin formulation is used to define the coarse grid operators is whether it is important that the coarse grid operators inherit the Gerschgorin stability condition from the fine grid operator. We have preliminary evidence that by choosing appropriate operator-determined prolongation and restriction operators [2, 18] we can indeed preserve the Gerschgorin condition on coarse grids, and that this leads to a very effective multigrid method. When complete, this will be reported in a separate paper.

As mentioned in the introduction, Mayo and coworkers developed an alternative approach to solving certain elliptic problems in irregular regions. For the Laplace equation with Dirichlet boundary conditions their approach makes use of an integral representation of the solution in terms of an unknown double layer potential times a kernel derived from the fundamental solution to the Laplace equation. An integral equation on the domain boundary is solved to determine the double layer potential. As in our method, the irregular domain is embedded in a larger rectangular domain and a Cartesian grid is placed over the rectangle. The integral representation is used to define boundary values on the edges of the rectangle, to define a nonsmooth extension of the solution to the entire rectangle, and to determine the jumps in the extended function and its derivatives across the boundary of the irregular domain. A standard five-point formula for the discrete Laplacian is used at all points of the grid. For "regular" points, the truncation error made in setting the discrete Laplacian to 0 is order h^2. For "irregular" points, Taylor series manipulations combined with jump

conditions expressed in terms of the double layer potential and its derivatives are used to derive a correction term m such that the standard discrete Laplacian at the irregular point equals the true Laplacian there plus the correction term plus additional terms of order h or higher. Since the true solution is harmonic, the truncation error made in setting the discrete Laplacian equal to the correction term is equal to these additional terms. A standard fast Poisson solver for rectangular domains is then used to solve the algebraic system consisting of the equations obtained by setting the discrete Laplacian equal to 0 at regular points and to the correction term m at irregular points. For Poisson's equation, a particular solution to the differential equation is first determined without regard to boundary conditions and then the above procedure is used to solve Laplace's equation with modified boundary conditions. The preliminary step of finding the particular solution involves another discrete Poisson solve on the rectangle. In [12, 13, 14, 15], solution of the Dirichlet problem is discussed in detail and it is said that similar techniques can be used for the Neumann problem but, in fact, the details of this are not spelled out.

There are some similarities and many differences between the approach of Mayo et al. and that described in our paper. Both use Taylor series expansions and jump conditions to modify the standard discrete Laplacian at irregular points. Mayo's approach makes use of potential theory and the solution of an integral equation to derive the jump conditions; our's uses purely local information obtained from the differential equation and local continuity and boundary conditions. Mayo's approach has the advantage of using the standard discrete Laplacian for which (i) stability is well understood and (ii) there are fast solvers available. Our's modifies the difference stencil and so requires a new analysis of stability (as above) and rules out the use of conventional fast solvers. The latter would be a major disadvantage if not for the possibility of developing appropriate multigrid methods for these systems. As indicated earlier, this development is under way and preliminary results are very encouraging. The approach of Mayo et al. makes essential use of an accurate solution to an integral equation. Obtaining this solution, whether by a Nystrom method with trapezoidal rule quadrature as in [12, 13] or with a fast multipole method as in [15], is the dominant computational cost of the method and requires implementation of a lot of machinery not required in our approach. More importantly though, the approach of Mayo et al. is limited to problems with a known fundamental solution and fast solver. It is therefore not generalizable to variable coefficient or anisotropic diffusion problems which are important, for example, in exploring the mechanical or electrical-conduction properties of biological tissues. This generalization should be relatively straightforward with our method.

In this paper, we have limited discussion of our method to Poisson's equation and the heat equation. There are also extensions of the method that will make it useful on a wider class of problems. The method has been combined with advections algorithms [6] to produce methods for solving advection-diffusion equations in regions of complex shape. Preliminary results with the combined algorithm show second order convergence. This advection-diffusion code is being applied to study continuum models of platelet aggregation. Other applications include propagation of action potentials in a realistic, bidomain model of the intact heart, for which it is necessary to solve the Poisson equation in an irregular domain [5]. Other methodological extensions which are being explored are multigrid solvers and local mesh refinement.

Acknowledgments. The authors are very grateful to Randy LeVeque and Zhilin Li for many helpful discussions during the early stages of this work and to David Eyre

for helping produce the three-dimensional figures.

REFERENCES

[1] L. M. Adams, *A multigrid algorithm for immersed interface problems*, in Proceedings of the 1996 Copper Mountain Multigrid Conference, Copper Mountain, CO, University of Colorado, Boulder, CO, 1996.

[2] J. E. Dendy, Jr., *Black box multigrid*, J. Comput. Phys., 48 (1982), pp. 366–386.

[3] G. E. Forsythe and W. R. Wasow, *Finite-Difference Methods for Partial Differential Equations*, John Wiley and Sons, New York, 1960.

[4] C. Johnson, *Numerical Solution of Partial Differential Equations by the Finite Element Method*, Cambridge University Press, New York, 1987.

[5] J. Keener and K. Bogar, *The numerical solution of the bidomain equation for cardiac tissue*, Chaos, 8 (1998), pp. 234–241.

[6] R. J. LeVeque, *High-resolution conservative algorithms for advection in incompressible flow*, SIAM J. Numer. Anal., 33 (1996), pp. 627–665.

[7] R. J. LeVeque and Z. L. Li, *The immersed interface method for elliptic equations with discontinuous coefficients and singular sources*, SIAM J. Numer. Anal., 31 (1994), pp. 1019–1044.

[8] R. J. LeVeque and Z. Li, *Immersed interface methods for Stokes flow with elastic boundaries or surface tension*, SIAM J. Sci. Comput., 18 (1997), pp. 709–735.

[9] Z. Li, *A note on immersed interface methods for three dimensional elliptic equations*, Comput. Math. Appl., 31 (1996), pp. 9–17.

[10] Z. Li, *Immersed interface method for moving interface problems*, Numer. Algorithms, 14 (1997), pp. 269–293.

[11] Z. Li, *A fast iterative algorithm for elliptic interface problems*, SIAM J. Numer. Anal., 35 (1998), pp. 230–254.

[12] A. Mayo, *The fast solution of Poisson's and the biharmonic equations on irregular regions*, SIAM J. Numer. Anal., 21 (1984), pp. 285–299.

[13] A. Mayo, *Fast high order accurate solution of Laplace's equation on irregular regions*, SIAM J. Sci. Statist. Comput., 6 (1985), pp. 144–157.

[14] A. Mayo, *The rapid evaluation of volume integrals of potential theory on general regions*, J. Comput. Phys., 100 (1992), pp. 236–245.

[15] A. McKenney, L. Greengard, and A. Mayo, *A fast Poisson solver for complex geometries*, J. Comput. Phys., 118 (1995), pp. 348–355.

[16] K. W. Morton and D. Mayers, *Numerical Solution of Partial Differential Equations: An Introduction*, Cambridge University Press, New York, 1994.

[17] J. F. Thompson, Z. Warsi, and C. W. Mastin, *Numerical Grid Generation: Foundations and Applications*, North-Holland, New York, Amsterdam, 1985.

[18] P. M. D. Zeeuw, *Matrix-dependent prolongations and restrictions in a blackbox multigrid solver*, J. Comput. Appl. Math., 33 (1990), pp. 1–27.

SIAM J. SCI. COMPUT.
Vol. 22, No. 5, pp. 1655–1674
© 2000 Society for Industrial and Applied Mathematics

BLENDING FINITE-DIFFERENCE AND VORTEX METHODS FOR INCOMPRESSIBLE FLOW COMPUTATIONS*

M. L. OULD-SALIHI†, G.-H. COTTET†, AND M. EL HAMRAOUI†

Abstract. We describe and illustrate numerical procedures that combine grid and particle solvers for the solution of the incompressible Navier–Stokes equations. These procedures include vortex in cell (VIC) and domain decomposition schemes. Numerical comparisons with pure finite-difference methods demonstrate the effectiveness of these techniques for various flow geometries, bounded or unbounded.

Key words. vortex methods, finite-difference methods, Navier–Stokes equations, domain decomposition methods

AMS subject classifications. 65M06, 65M55, 76M06

PII. S1064827599350769

1. Introduction.

The idea of coupling finite-difference and particle methods is rather common in the field of Navier–Stokes and kinetic equations. The motivation for using particles is to reduce the dimensionality of the phase space and to deal in an elegant and robust way with the transport terms in the equations, while grid solvers can rely on fast Fourier transform (FFT) to evaluate the fields at a minimal numerical cost.

In plasma simulations, grid techniques are indeed the usual way to compute the fields that move the particles. For incompressible flow calculations, however, the common wisdom is that grid solvers not only go against the Lagrangian features of vortex methods, but they also cannot retain the subgrid information that particles carry with them.

The goal of this paper is to show that, when used with the appropriate tools, particle-grid methods combine accuracy, speed, and robustness and provide an efficient alternative to other high-order methods for a number of high Reynolds flows. This point will be made by direct comparisons with centered finite-difference, both from the accuracy and CPU time points of view. As a matter of fact, we believe that direct comparisons of vortex methods with other types of methods, although not very common in the literature, should be done in a systematic way in order to demonstrate their usefulness in computational fluid dynamics (CFD).

We will be concerned with two types of particle-grid techniques. In the first type, the particles and the grid are used in the same computational domain to deal with different aspects of the flow equations. Typically, particles solve the transport equations and the grid can be used to compute the velocity. The diffusion and the vorticity boundary conditions can be solved by either method. These methods are in the spirit of the vortex in cell (VIC) method pioneered by Christiansen [3]. They retain the key feature of particle methods; namely, the advection part of the equation is dealt with in a Lagrangian fashion and reduces to a set of differential equations. As a result, the stability of the vorticity transport and stretching equations is not constrained by classical CFL conditions. As for vortex-blob methods, the time step

*Received by the editors February 1, 1999; accepted for publication (in revised form) May 23, 2000; published electronically December 5, 2000.
http://www.siam.org/journals/sisc/22-5/35076.html
†LMC-IMAG, Université Joseph Fourier, B.P. 53, 38041 Grenoble Cédex 9, France (cottet@imag.fr).

is determined as a function of the maximum strain in the flow (typically a fraction of the inverse of the vorticity maximum). We will see that they can take advantage of high-order interpolation formulas for optimal particle-grid transfers.

In the second kind of method, the computational domain is split into several subdomains where different solvers are used. The coupling between the methods then has to be done within the framework of domain decomposition techniques and is most easily designed with overlapping subdomains. These methods have been proposed in [4]. The developments we propose here take advantage of high-order regridding of the particles in the overlapping zone. The gain that can be expected from this type of technique in the simulation of complex flows around obstacles is to combine the flexibility of grid and particle methods, respectively, near the solid boundaries and in the wake.

The outline of the paper is as follows. In section 2 we recall the principles of particle-grid assignment and interpolation, which are essential in all particle-grid techniques. Section 3 is devoted to VIC methods. We define these methods for 2D and 3D, ideal as well as viscous, flows and illustrate them on several classic 2D and 3D test cases with strong vortex–wall interactions. In section 4 we focus on particle-grid domain decomposition. We recall the key features of these methods and show that the principles of the method can be indifferently applied to the case when the finite-difference solver is used on a velocity-pressure or velocity–vorticity formulation of the Navier–Stokes equations. The method is then illustrated in the case of a flow over a backward-facing step and the wake of a cylinder. Finally, in section 5 we draw some conclusions on the suitability of particle-grid techniques for the simulation of complex 3D flows.

2. Particle-grid operators. All methods combining grids and particles rely heavily on the ability one has to transfer quantities such as vorticity and velocity between these two types of discretization. These transfers are most simply done by interpolation. Assume that we are given a distribution of particles located at x_p carrying circulations Γ_p, and thus a vorticity field is given by $\omega(x) = \sum_p \Gamma_p \delta(x - x_p)$. If ϕ_i is a basis of functions associated with the grid, a natural formula to assign vorticity values to the grid is

$$\omega_i = \frac{1}{V_i} \sum_p \Gamma_p \phi_i(x_p), \tag{1}$$

where V_i is the volume of the grid cell centered at x_i. Conversely, grid quantities, like velocities, are interpolated at particle locations with the formula

$$u_p = \sum_i u_i \phi_i(x_p). \tag{2}$$

In practice the grid often is deduced from a uniform Cartesian grid through a mapping given by explicit formulas (this is true, in particular, when one wishes to use finite-difference-type grid solvers, which is the case we have in mind). By using the same mapping formulas for the particles, one is led back to the case of a translation invariant grid, with grid size ε, and $V_i = \varepsilon^d$ (d is the dimension). The basis (ϕ_i) then reduces to a single kernel ϕ by the formula $\phi_i(x) = \phi(\frac{x - x_i}{\varepsilon})$, where ε is the grid size. Moreover, ϕ can be chosen as a tensor product of 1D kernels, and from now on we will focus on 1D kernels.

A class of kernels is given by successive convolutions of the top-hat filter χ with support in $[-1/2, 1/2]$. This gives kernels of increasing smoothness, which spread

the circulation of one particle onto an increasing number of points. To evaluate the accuracy of a given kernel it is customary to consider its conservation properties. Conservation of the total circulation, as measured by the particles and the grid, reads

$$\sum_i \omega_i V_i = \sum_p \Gamma_p.$$

It is clearly requires that the kernel ϕ satisfies

$$\sum_i \phi\left(\frac{x - x_i}{\varepsilon}\right) \equiv 1. \tag{3}$$

Conservation of linear and angular impulse, respectively, then requires that

$$\sum_i x_i \phi\left(\frac{x - x_i}{\varepsilon}\right) \equiv x, \tag{4}$$

$$\sum_i x_i^2 \phi\left(\frac{x - x_i}{\varepsilon}\right) \equiv x^2. \tag{5}$$

It is readily seen that these properties imply that the interpolation formulas will be of orders one, two, and three, respectively. Fourier analysis [23] gives simple criteria to determine the accuracy of a given kernel. Let g be the Fourier transform of ϕ:

$$g(k) = \int_{-\infty}^{+\infty} \phi(x) \varepsilon^{-ikx} dx.$$

One can show that ϕ yields an interpolation formula of order m (that is, it conserves the moments up to $m - 1$) if the following two conditions hold simultaneously:

$$k \to g(k) - 1 \text{ has a zero of order } m \text{ at } k = 0, \tag{6}$$

$$k \to g(k) \text{ has zeros of order } m \text{ at all } k = 2\pi n \qquad (n \neq 0). \tag{7}$$

Since the Fourier transform of χ is given by

$$g(k) = \frac{\sin(\pi k)}{\pi k},$$

this criterion clearly shows that χ is of order 1, while $\chi \star \chi$ and all the following kernels are second order.

In practice, the requirement one must have on the kernel accuracy very much depends on what is to be done on the grid. In a VIC calculation, where vorticity values on the grid are used only to compute the velocity, one can tolerate a certain level of dissipation in the assignment, and second-order formulas can give sufficient accuracy. However, if vorticity on the grid is directly used in the vorticity equation to solve for the diffusion or in a vorticity boundary condition, for instance, the numerical dissipation induced by these formulas can affect the overall performance of the method. In this case it is advisable to use higher-order formulas. All our calculations use the

following third-order kernel first proposed by Monaghan [20] in the context of smooth particle hydrodynamics (SPH) calculations:

$$\phi(x) = \begin{cases} 0 & \text{if } |x| > 2, \\ \frac{1}{2}(2-|x|)^2(1-|x|) & \text{if } 1 \le |x| \le 2, \\ 1 - \frac{5x^2}{2} + \frac{3|x|^3}{2} & \text{if } |x| \le 1. \end{cases} \tag{8}$$

The theoretical order of accuracy of a given kernel is actually obtained only insofar as the particles are able to represent a smooth function. Thus, besides the order of the interpolation kernel, another important factor in the accuracy of particle-grid schemes is the regularity of the particle distribution. Vortex-grid and vortex-blob methods are no different from this point of view. In the high Reynolds flows that we examine, the particles are likely to undergo severe distortions. Some cures can be proposed to maintain a minimal accuracy in the particle-grid transfers. One is to implement the formula (1) with a volume value deduced from the particles through the formula

$$V_i = \sum_p v_p \phi_i(x_p), \tag{9}$$

where v_p are the volumes of the particles. This formula has the effect of compensating for a local lack or excess of particles, because a constant vorticity field grid assignment with the formula (9) recovers the exact value, no matter how distorted the particles are. One may also process the weights of the particles in order to recover the correct vorticity values on the particles, along the same lines as in the iterative method proposed in the context of vortex-blob methods by Beale [1]. However, it is possible to maintain regularity in the particle distribution at almost no cost and with no discernible numerical dissipation (see [9]) by frequently regridding the particles. This has the effect of maintaining a good accuracy for the interpolation formulas. In all the vortex calculations shown in this paper, the same kernel (8) will be used for both the particle-grid transfers and for regridding the particles on regular locations.

3. VIC methods. Let us first recall the velocity–vorticity formulation of the incompressible Navier–Stokes equations

$$\frac{\partial \omega}{\partial t} + (u \cdot \nabla)\omega - (\omega \cdot \nabla)u - \sigma \Delta \omega = 0, \tag{10}$$

$$u = \nabla \times \psi, \tag{11}$$

$$-\Delta \psi = \omega. \tag{12}$$

This system has to be supplemented with initial and boundary conditions. Here we are interested in methods where grid and particle solvers coexist in the same computational domain. One important motivation for using Navier–Stokes vortex solvers lies in their robust and accurate treatment of the advection terms in (10): The vortex solution to (10) consists of moving vortices with their local velocities and updating their circulations to account for stretching and diffusion, and no explicit computation of the convection term $(u \cdot \nabla)\omega$ is required. On the other hand, grid solves can take advantage of FFT and fast Poisson solvers to compute velocity fields in (11)–(12).

In the early 1970s, grid Poisson solvers were actually the only way to do vortex simulations with more than a thousand particles. The advent of fast summation techniques has allowed completely grid-free simulations using up to a million particles, but in regular bounded geometries grid-based velocity solvers still offer a significant speed-up, unless particles occupy a very small part of the domain.

A typical VIC algorithm in two dimensions consists of the following sequence:

- Assign circulations to the grid with formula (1),
- solve (12) for the stream function on the grid with the desired boundary condition,
- compute grid velocities by finite differences,
- interpolate velocities on particles and move particles.

In this algorithm, most of the CPU time is devoted in the interpolation formulas involved in the particle-grid operators. A typical CPU time for the computation of the velocities on two million particles with the same number of grid points is a few seconds on a DEC workstation in three dimensions. To give an idea of the savings over a fast multipole algorithm in a vortex-blob method, from the figures given in [24] we can extrapolate that the same number of particles would require with these methods more than an hour of CPU time.

In a viscous code, it is also possible to solve for the diffusion on the grid. After the assignment step one computes $\Delta\omega$ on the grid by finite differences and then interpolates this quantity on the particles before updating their circulation. One must be aware that this scheme is not conservative. A conservative variant can be obtained with a finite-element formulation (see [6]). However, our simulations show that, when regridding, we maintain enough regularity in the particle distribution so the lack of conservation is not noticeable.

In the case where the grid is used for the diffusion, also it must handle a vorticity boundary condition to supplement (10) and translate the desired velocity boundary condition. One may either use an influence matrix technique to derive exact, at the grid level, vorticity boundary conditions, or rely on Taylor expansions to derive approximate boundary conditions, accurate to a given order. There has been in the last years extensive work in this direction. We refer in particular to [22, 11] and the references therein. We now illustrate these techniques on three flow examples: the 2D driven cavity, vortex dipoles, and vortex rings.

3.1. Driven cavity flow. In the VIC method implemented here, the diffusion is solved on the grid. It is compared with a finite-difference scheme using a centered second-order approximation of the advection and diffusion terms. When used together with fourth-order Runge–Kutta time stepping, this scheme does not suffer any cell Reynolds number limitations and is linearly stable under a convection CFL condition. We have implemented several other finite-difference schemes, in particular Arakawa and a fourth-order compact scheme [12]. We have observed that, when used with sufficient resolution, these schemes give very close results. When insufficiently resolved, all centered schemes ultimately lead to instabilities. The second-order scheme has been chosen in view of its simplicity and low CPU cost.

Concerning the vorticity boundary condition, we have implemented Thom's first-order formula, various second-order formulas, and Briley's fourth-order formulas. Here again we have observed that the difference between second- and fourth-order formulas was very marginal. (These formulas, however, clearly outperform Thom's formulas.) We have finally chosen the following second-order scheme: velocities at the interior grid points are computed from the stream function and at the boundary using the

TABLE 1
Run parameters and CPU times with VIC and finite-difference methods for various Reynolds numbers and resolutions.

Reynolds number	100	2000	10,000
M_{fd}	64	128	256
M_{vic}	64	128	256
δt_{fd}	0.01	0.008	0.004
δt_{vic}	0.01	0.02	0.04
CPU time for finite-difference scheme	3	24	225
CPU time for VIC scheme	5	16	32

velocity boundary conditions; then second-order one-sided finite differences are used to compute the vorticity at the boundary, which yields the Dirichlet boundary conditions used in the diffusion solver. The same vorticity boundary condition is used in the VIC method.

The finite-difference method used a fourth-order Runge–Kutta time stepping. This is actually necessary for the centered scheme to be stable [12] under a CFL condition. As for VIC method, it does not impose on the time step any stability constraint from the treatment of the convection term. In our simulations with the VIC method, we used second-order Runge–Kutta time stepping.

Due to the explicit treatment of the diffusion, for both methods the time step also must satisfy a diffusion CFL. In practice, for high Reynolds number flows, this last stability condition is far less drastic than the advective CFL condition. As a result, in most cases of interest the VIC code could be run with bigger time steps than the finite-difference code. Table 1 summarizes some typical run resolution and CPU times for a given simulation time on a Sun Sparc station. Due to the interpolation steps, each iteration of the VIC code requires slightly more computational time than the finite-difference code, but for Reynolds numbers beyond 10,000, the bigger time steps used in the VIC code lead to substantial CPU savings. Another nice feature of the VIC method is that it is insensitive to the lack of smoothness of the flow, whereas centered finite difference may become unstable when used without sufficient resolution if sharp vorticity gradients develop.

Figure 1 shows the vorticity contours obtained with the VIC and the finite-difference method for 256^2 and 512^2 resolutions.

The Reynolds number is 10,000, and the velocity profile at the top boundary has a parabolic profile. At the lower resolution, the two methods diverge from each other in the transient stage before reaching the same quasi-steady-state regime. As we refine the resolution, both methods converge toward the same solution. This is confirmed by looking at the vorticity profile on the first diagonal (Figure 2). The 256^2 resolution is enough in the VIC code to obtain converged results for a large time behavior. At this resolution it yields a sevenfold speed-up over the finite-difference method.

This CPU gain makes affordable large time simulations at higher Reynolds numbers. Figure 3 shows successive stages of the vorticity field in a driven cavity at a Reynolds number of 10^5, with 1024^2 grid points. (In this calculation, as in all our VIC simulations, we used the same mesh size for the grid and the particles.) The same resolution would have required a CPU time 17 times larger with the finite-difference method. (However, it is fair to mention that, due to the proliferation of small scales at this Reynolds number, it is virtually impossible to obtain converged

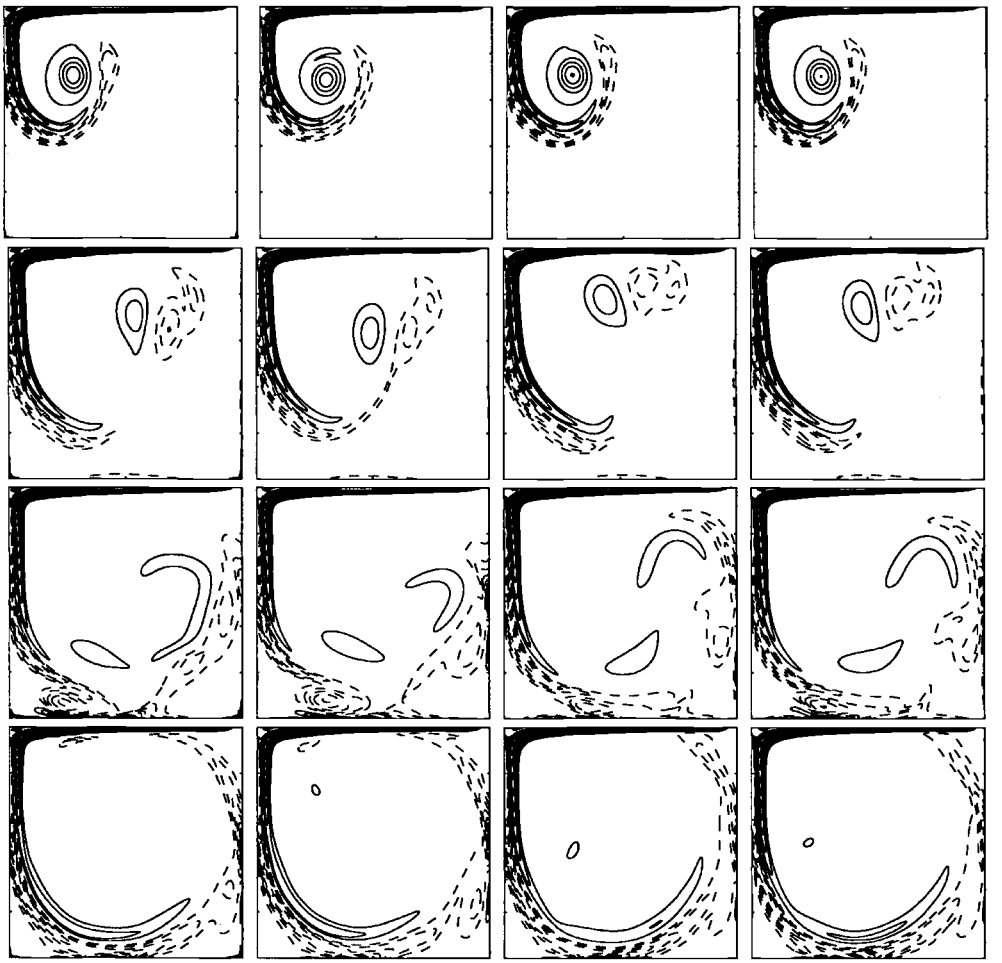


FIG. 1. *Sequence of vorticity profiles in a driven cavity at a Reynolds number of* 10^4. *From left to right: VIC method with* 256^2 *resolution, finite-difference method with* 256^2 *resolution, VIC method with* 512^2, *and finite-difference method with* 512^2. *From top to bottom: times* 10, 20, 30, 40.

results; these simulations only convey qualitative information, such as the number of eddies.) Demonstrations of this simulation can be downloaded from the World Wide Web site www-lmc.imag.fr/lmc-edp/Georges-Henri.Cottet/vortex.html.

3.2. Dipole-wall interaction. In the calculations we now show, the diffusion and vorticity boundary conditions are computed on the particles and the grid is used only to compute the velocity. The particle diffusion solver is a particle strength exchange (PSE) scheme [10]; that is, particles exchange vorticity among them through

$$\frac{d\omega_p}{dt} = \sigma\varepsilon^{-2}\sum_q v_q(\omega_q - \omega_p)\Lambda_\varepsilon(x_p - x_q).$$

In the above formula, v_q are the volumes of the particles (typically h^d when particles are initialized on a uniform mesh with mesh size h) and Λ is a kernel satisfying appropriate second-order moment conditions.

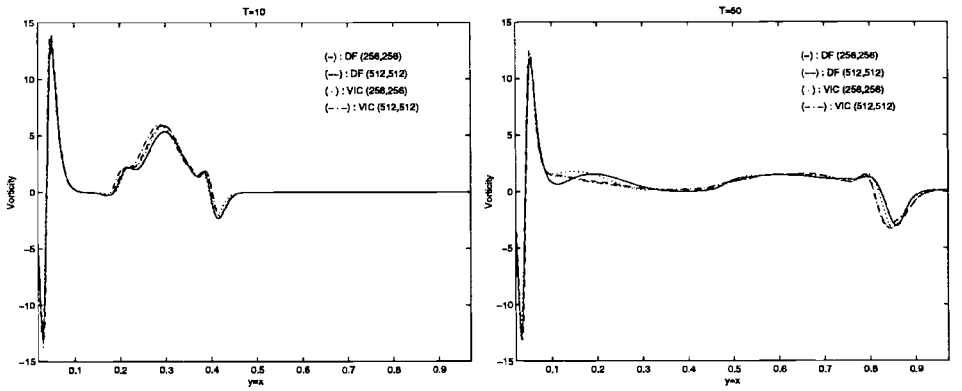


FIG. 2. *Vorticity profiles along the line* $x_1 = x_2$ *in a driven cavity at a Reynolds number of* 10^4.

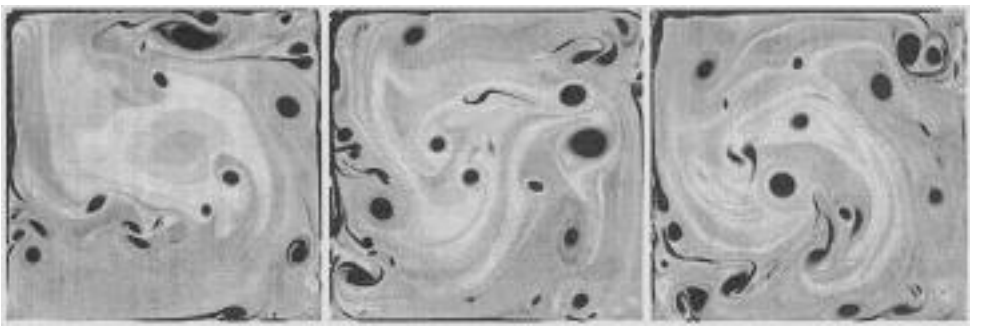


FIG. 3. *Successive stages of the evolution of the driven cavity at a Reynolds number of* 10^5. *VIC method using* 1024^2 *particles.*

This diffusion scheme is used in a viscous splitting algorithm with a Neumann vorticity boundary condition translating the no-slip condition at the wall. This algorithm is described in [5, 16]. It consists of first doing a time step of the PSE scheme, then evaluating the slip $u \cdot \tau$ at the wall. Finally, the diffusion step is repeated with a vorticity flux equal to $-u \cdot \tau / \delta t$. This last step is solved by means of integral equations.

This scheme has been applied successfully in various geometries [15, 17]. However, to our knowledge there has not been any direct comparison in terms of CPU time and accuracy of this method with grid-based schemes. On the basis of the results for the driven cavity, we have chosen to compare the VIC method with a second-order centered finite-difference scheme with fourth-order Runge–Kutta time stepping and the second-order vorticity boundary condition described above for the driven cavity problem.

Figure 4 shows a comparison of the results given by the VIC and finite-difference codes at a Reynolds number of 800. The initial condition is a Lamb dipole with circulation $\Gamma = 6.83$. The boundary conditions are periodic in the direction parallel to the wall. At the top boundary we assume no through flow and zero vorticity. The resolution is $\Delta x = 0.027$ for both simulations. Due to the advection CFL, the time step for the finite-difference scheme is five times smaller than for the VIC code. The results are in good agreement, although as time goes on the VIC solution seems to go slightly faster than the finite-difference solution.

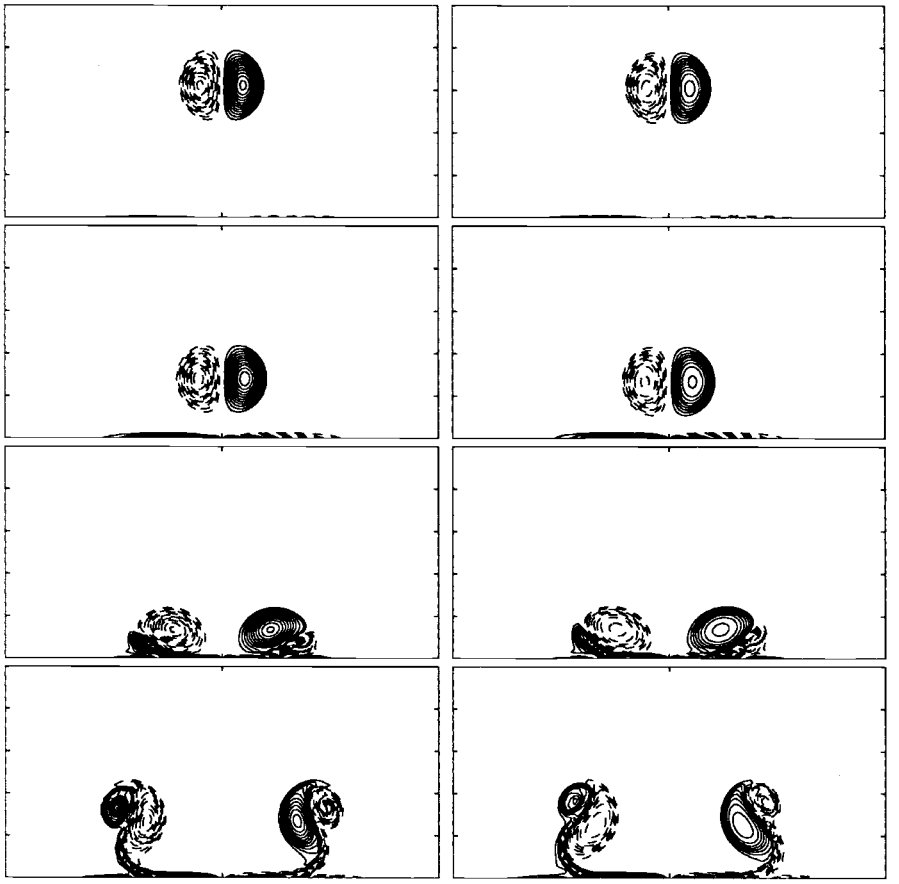


FIG. 4. *Contours of vorticity for a vortex dipole impinging a wall at times* $2, 4, 6, 8$. *Left column: finite-difference method,* $\Delta t = .02$*; right column: vortex method,* $\Delta t = 0.1$.

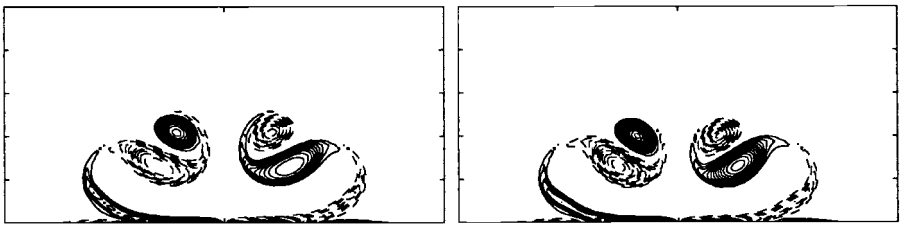


FIG. 5. *Contours of vorticity for a vortex dipole impinging a wall at time* 10. *Reynolds number* 800*, vortex method. Left picture:* $\Delta x = .027$*,* $\Delta t = .1$*; right picture:* $\Delta x = 0.0135$*,* $\Delta t = .05$.

Figure 5 is a refinement study at a later time for the VIC method (the resolution and time steps have been divided by a factor of 2), which shows that the VIC results in Figure 4 can be considered as nearly converged results.

Finally, Figure 6 is a VIC simulation at a Reynolds number of 1600. The resolution has been increased to $\Delta x = 0.0195$, but the time step has been kept equal to 0.1. For the same resolution, the finite-difference method would have required a time step 10 times smaller. Compared with the former case, the lower viscous dis-

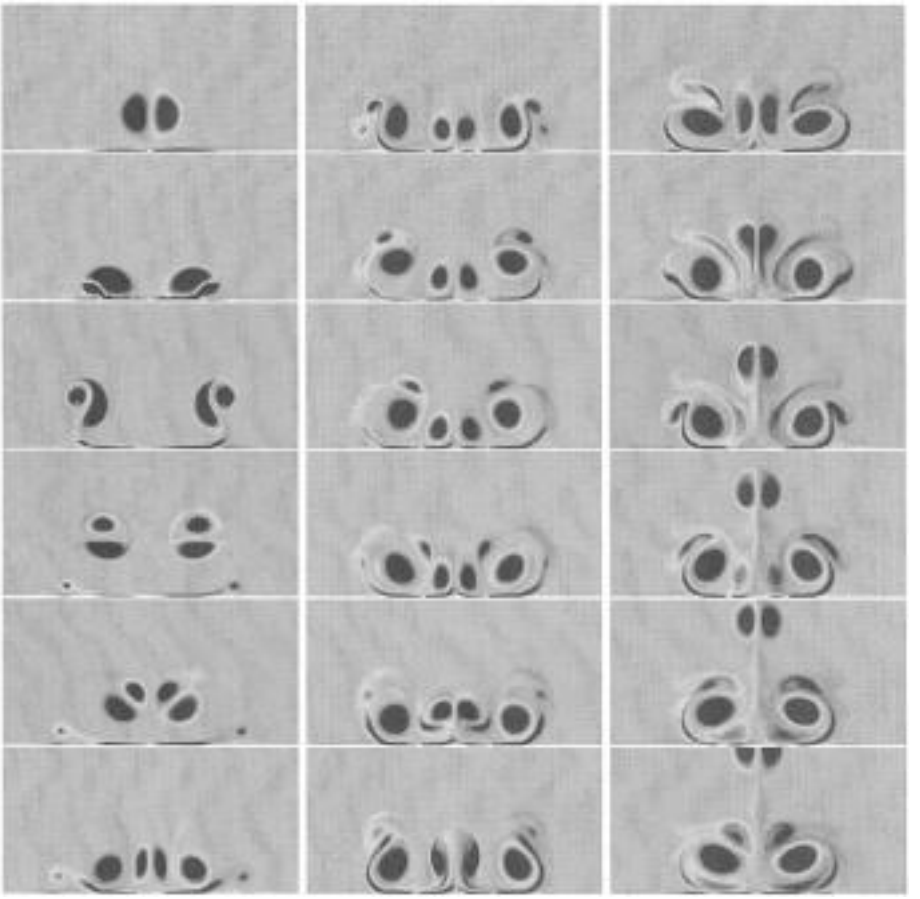


FIG. 6. *Vortex dipole impinging a wall by a VIC method with* $\Delta x = .0195$, $\Delta t = .1$. *Reynolds number of* 1600.

sipation yields larger circulations in the secondary vorticity produced at the wall. The resulting new dipole therefore gets enough circulation to eject itself from the wall.

These results are in good agreement with the results of Orlandi [21], however, with the same delay as noticed for the Reynolds 800 case.

It is interesting to note that, although first order in time, the vorticity flux boundary conditions give a satisfactory accuracy even when used with large time steps. By contrast, we have found that the vorticity boundary conditions given by Thom's formula, which is first-order accurate in space, failed to give acceptable results for this type of flow where it is crucial to capture accurately the sharp vorticity gradients produced at the boundary.

3.3. Ring-wall interactions. Here we consider the 3D counterpart of the dipole calculations just seen. Let us first describe how to extend the VIC schemes described in section 3.2 to three dimensions. First we must discretize the stretching term $(\omega \cdot \nabla)u$ appearing in the vorticity equation. One way is to obtain velocity derivatives by finite difference on the grid, then to interpolate these values at the particle locations. Particle vorticities are then updated, as in a grid-free vortex method,

by solving

$$\frac{d\omega_p}{dt} = [\nabla u(x_p)]\omega_p.$$

Another possibility, which we have elected in our calculations, is based on the conservative form $\mathrm{div}(\omega : u)$ of the stretching term. The idea is to multiply grid values of vorticity and velocity. The tensor $\omega_i u_j$ is then differentiated on the grid, and its divergence is finally interpolated on the particles to update their circulations. This last option has the advantage of being conservative at the grid level, even when the vorticity field is not exactly divergence-free. To avoid numerical dissipation in this procedure, we have used a fourth-order finite-difference formula for the divergence to compute the stretching term on the grid.

The second point that needs to be clarified is the extension to three dimensions of the vorticity flux boundary condition used in the 2D vortex schemes. One needs boundary conditions for each vorticity component. For no-slip velocity, the normal component of the vorticity must clearly vanish. For the components parallel to the wall, we adopt the same viscous-splitting point of view as in the 2D case and write for them a Neumann vorticity boundary condition to cancel the slip in the orthogonal direction of the wall. We refer to [6] for a more detailed account of this technique and a proof that it does not violate the divergence-free constraints for the vorticity.

As explained in [21], the features of the vorticity equation in three dimensions imply some important differences between the ring-wall and the dipole-wall interactions. These differences are noticeable particularly when the ring hits the wall with an angle. In this case the lower part of the ring is subject, as it approaches the wall, to intense stretching, which has the effect of pumping its circulation, which then feeds the upper part of the ring, causing it to rebound. We have chosen for these simulations the same parameter as in [25]: The angle is 38.5 degrees, the computational box is $[0,1]^2 \times [0,1/2]$ with periodic boundary conditions in the directions parallel to the wall $z = 0$, no slip at the wall, and no through flow and zero vorticity at the top boundary. The ring has radius $R = 1/8$ with a Gaussian core of radius equal to $0.4R$. At time zero it is located at a distance 0.25 of the bottom wall. The Reynolds number, based on the ring circulation, is 1400.

Particles were initialized and remeshed on a regular 128^3 grid. The same grid resolution was used for the computation of the velocity field. This resolution is roughly the same as in [25] (these authors used $129 \times 91 \times 97$ grid points, with grid refinement in the normal direction of about a factor of 2 near the wall). At the last stage of the computation considered here, there were about 900,000 particles (which means that the vorticity occupied roughly half of the computational domain).

Figure 7 shows the contours of the y-component of the vorticity in the symmetry (x, z) plane together with isosurfaces of the vorticity magnitude (for clarity, only half of the surfaces are shown) at times 24, 40, and 64. These results are in excellent agreement with the results of [25]. At this resolution the time steps used in the finite-difference and vortex methods are of the same order, and the goal of this example mostly was to demonstrate the ability of vortex methods to perform accurate direct numerical simulations of complex vortex-wall interactions. Quantitative comparisons with spectral methods for homogeneous turbulent flows and Crow instabilities are presented elsewhere [8].

4. Particle-grid domain decomposition. We now describe numerical techniques in which grid and particles are used in different subdomains of the computational domain. These techniques are rather natural: Vortex methods are attractive

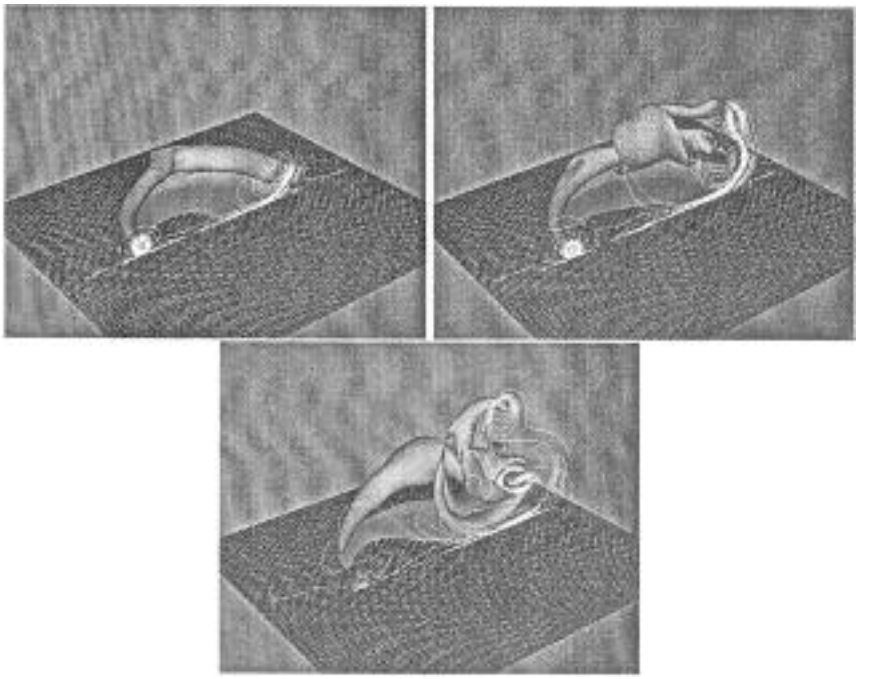


FIG. 7. *Vortex ring hitting a wall at an angle of* 38.5 *degrees. Contours of the vorticity component perpendicular to the symmetry plane, and isosurfaces of vorticity magnitude at times* 24, 40, *and* 64 *(from left to right, top to bottom).*

particularly in flow regions where the vorticity is confined to areas of small dimension and do not need to model far-field conditions, whereas grid-based methods offer more flexibility in dealing with boundary conditions at solid walls, in particular because using the velocity-pressure formulation is allowed.

4.1. Design. Particle-grid domains have been formalized for the first time in [4]. Note that at this time there was no clear understanding of how to accurately implement no-slip boundary conditions in vortex methods, which was a strong motivation for coupling vortex methods with grid techniques near obstacles. The method described in this reference is based on the velocity–vorticity formulation of the equations in the whole domain, but as we will see it extends readily to the case when a velocity-pressure formulation is chosen in the finite-difference subdomain.

As stressed in [4], for a clear-cut particle-grid domain decomposition algorithm, it is important to separate the kinematic and kinetic parts of the flow equations. They correspond to equations of a different nature and thus require different interface conditions.

Without loss of generality, we will assume no-slip boundary conditions and a domain decomposition as sketched in Figure 8. Ω_1 and Ω_2 are, respectively, the finite-difference and vortex domains. Important ingredients in the method will be the overlapping of Ω_1 and Ω_2 and the fact that there exists a domain Ω_2' including Ω_2 such that particles and grid coexist in $\Omega_1 \cap \Omega_2'$. The precise role played by Ω_2' and the geometrical constraints imposed on the width of the overlapping will appear in the discussion below.

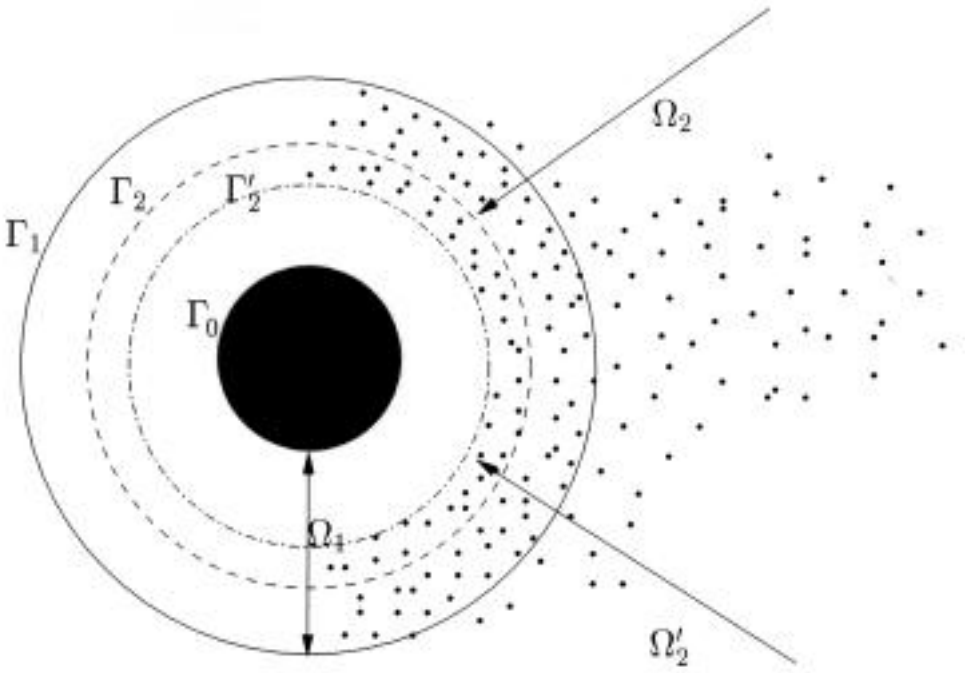


FIG. 8. *Particle-grid domain decomposition with overlapping.*

Let us mention that an alternative approach, without overlapping, has been proposed in [13]. This method, however, does not clearly indicate the interface conditions on which it relies and its consistency is not obvious.

Let us first consider the case when the velocity–vorticity formulation is used in both domains. At each time step—or substep in a Runge–Kutta scheme—one has to solve the elliptic equation $-\Delta\psi = \omega$ in each subdomain with the appropriate boundary conditions. While on Γ_0 the no-through-flow condition gives a homogeneous Dirichlet boundary condition, on Γ_1 and Γ_2 these boundary conditions are part of the calculations. A popular and efficient way to determine them is the Schwarz alternating method. In our case, it is slightly modified to take into account that different solvers are used in the two subdomains: a finite-difference solver and an integral one, given by the Poincaré identity, in the vortex zone. Note that, unlike for the bounded flows considered in the previous section, a VIC approach would not be appropriate in Ω_2, as it would not retain the ability of vortex methods to exactly satisfy the far-field condition. One iteration of the Schwarz algorithm proceeds as follows.

1. Solve $-\Delta\psi = \omega$ in Ω_1 by finite differences with $\psi = 0$ on Γ_0 and $\psi = a$ on Γ_1.
2. Compute $b = \psi$ and $c = \partial\psi/\partial n$ on source points on Γ_2.
3. Evaluate

$$\psi = \int_{\Omega_2} G(x-y)\omega(y)\,dy + \int_{\Gamma_2} G(x-y)c(y)\,dy + \int_{\Gamma_2} \frac{\partial G}{\partial \nu}(x-y)b(y)\,dy$$

on grid points of Γ_1, where ω is the vortex-blob solution in Ω_2 and G is the Green function $G(x) = \log|x|/2\pi$, to get new values of a on Γ_1.

Once the stream function, and thus the velocity, is known on both subdomains, we can describe how to update the vorticity. While the vortex method is always based on explicit time discretization of the equations, one may consider finite-difference solvers using either an explicit or implicit treatment of the diffusion. However, for the Reynolds numbers we are interested in, the time step is always limited by the advective CFL condition, and explicit diffusion solvers do not yield any additional stability constraints, so we will discard the option of implicit schemes.

To simplify the discussion we will consider the case of a forward Euler time discretization in both domains. To advance from time t_n to time t_{n+1}, explicit finite-difference solvers require only the knowledge at time t_n of the vorticity at the grid points on the boundary, assuming a second-order stencil, or a higher-order method with one-sided finite differences near the boundaries. We have already indicated in section 3 how to deal with the no-slip boundary on Γ_0. Since $\Gamma_1 \subset \Omega_2$, it is natural to obtain boundary values there by assigning vorticity values available at that time on particles of Ω_2, provided Ω_2 contains all the particles contributing to this assignment: If ε is the grid size and the interpolation kernel is given by (8), this means that $d(\Gamma_1, \Gamma_2) \geq 3\varepsilon$. Conversely, to update the circulations of the particles in Ω_2 one needs the circulations at time t_n of the particles in this domain surrounded by a layer of size at least $u_{\max}\Delta t + \rho$, where ρ is the width of the kernel used in the PSE scheme (of the order of the grid size ε). This dictates the geometrical constraints that have to be fulfilled by Ω_2 and Ω_2': One must have $d(\Gamma_2, \Gamma_2') \geq u_{\max}\Delta t + \rho$. In practice the various boundaries are separated by distances that are fractions of the body size, and these constraints are satisfied easily. To update the circulations of the particles in Ω_2, one finally interpolates grid values on the particles in $\Omega_2' - \Omega_2$ and includes these particles into the vortex scheme.

Note that the scheme just described is nothing but an extension of what would be done if a finite-difference scheme was used in both domains Ω_1 and Ω_2. In some sense the particles in $\Omega_2' - \Omega_2$ play the role of the outer grid points that would be needed in a finite-difference solver.

In summary, the algorithm proceeds as follows, assuming a forward Euler time stepping in both subdomains: Given a vorticity field ω^n known on the grid in Ω_1 and on particles in Ω_2,

- compute the stream function on Γ_1 and Γ_2 and its normal derivative on Γ_1, using the Schwarz alternating method;
- deduce velocity values on the grid and on particles in Ω_2;
- interpolate grid values of ω^n on particles in $\Omega_2' - \Omega_2$, and assign particle circulations to grid points on Γ_1;
- update vorticity in both domains to obtain ω^{n+1}.

In our calculations, this scheme is extended to a fourth-order Runge–Kutta time discretization in both domains.

The consistency of the interface conditions just described clearly relies on the ability of the particles in $\Omega_2' \cap \Omega_1$ to transfer vorticity between the two subdomains. This is achieved by periodically regridding particles in Ω_2' and by using high-order interpolation formulas like those described in section 2.

Let us mention that very similar ingredients can be used to design particle–particle domain decomposition methods. In this case, vortex methods are used in all subdomains, but they correspond to different mesh resolutions. We refer to [6, 7] for details.

We now turn to the case when a velocity-pressure formulation is used in the finite-difference domain. We have chosen in our simulations a second-order projection

technique that is a straight-forward extension, based on a second-order Runge–Kutta time advancing scheme, of Chorin's classical projection method [2]. For the sake of simplicity, as in the velocity–vorticity case, we will describe the domain decomposition method assuming a forward Euler discretization. In this case, starting from a divergence-free velocity u^n the projection method consists in computing

$$R^n = -(u^n \cdot \nabla)u^n + \sigma \Delta u^n \tag{13}$$

then

$$u^{n+1} = u^n + \Delta t(R^n + \nabla p^n). \tag{14}$$

The pressure p^n is adjusted to make u^{n+1} divergence-free and to satisfy the appropriate boundary condition for the normal component:

$$\Delta p^n = -\text{div } R^n \text{ in } \Omega_1, \tag{15}$$

$$\frac{\partial p^n}{\partial \nu} = -R^n \cdot \nu + \frac{u^{n+1} - u^n}{\Delta t} \cdot \nu \text{ on } \partial\Omega_1. \tag{16}$$

This system requires the knowledge of $u^{n+1} \cdot \nu$ on Γ_1. It may be obtained again by a Schwarz alternating method: Each Schwarz iteration consists of solving (15), (16), then evaluating the velocity on Γ_2 with (14), and finally using the Poincaré identity in Ω_2 to obtain new values for $u^{n+1} \cdot \nu$ on Γ_1.

One step of the complete algorithm, still in the case of a forward Euler scheme, proceeds as follows, starting from u^n in Ω_1 and ω^n in Ω_2:

- compute the particle velocities in Ω_2 with the Poincaré formula;
- differentiate u^n on the grid in Ω_1 and interpolate grid vorticity values on particles in $\Omega_2' - \Omega_2$;
- update particles in Ω_2 to get ω^{n+1};
- compute G^n, then $u^{n+1} \cdot \nu$ on Γ_1 through (14)–(16) and the Schwarz method, and finally u^{n+1}.

For a Runge–Kutta scheme, these steps have to be repeated at each substep.

Another type of particle-grid coupling method with overlapping has been proposed in [14]. The main difference with the method of [4] used here is that particle velocities are computed using a Biot–Savart law involving both particles and grid points. We believe that this approach is more expensive, since the Biot–Savart solver, even in the most efficient implementations of fast solvers, is significantly slower than a finite-difference one in simple geometries (in our calculations the number of grid points was always larger than the number of particles, even for a wake extending far downstream). Moreover, mixing grid points and particles in the same quadrature formula requires adjusting their volumes in the interface zone, which introduces some technical complications.

4.2. Numerical results. We have selected here a few results that illustrate the main features of the method. A more systematic study of particle-grid domain decomposition can be found in [18].

Our first example is to demonstrate that the interface conditions, although combining solvers of a different nature, maintain a smooth transfer of vorticity. Figure 9 shows the vorticity in a flow over a backward-facing step at a Reynolds number of 355. In the bottom picture, the finite-difference zone surrounds the top and bottom limits of the channel. The interface Γ_2 with particle is shown by a dotted line.

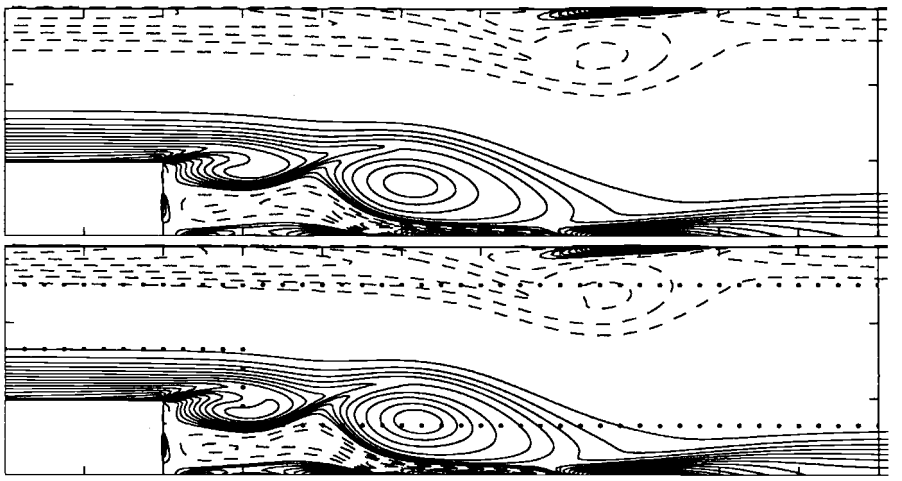


FIG. 9. *Vorticity contours for a flow over a backward-facing step at a Reynolds number of* 355. *Top picture: finite-difference calculation; bottom picture: particle-grid domain decomposition. The bullets materialize the interface.*

As a comparison, the top picture shows the results of a calculation using the finite-difference solver everywhere. The results are almost identical. Although the interface cuts through several recirculation zones with sharp vorticity gradients, no oscillations in the vorticity are discernible. Particles and grid have the same mesh size equal to 3% of the height of the step. This results in about 30,000 discretization points.

We now focus on flows past a cylinder. This is a good prototype of an unbounded flow producing complex dynamics near the boundary and it is well documented. In all our calculations the cylinder had a unit radius, and the velocity at infinity was equal to 1. The finite-difference domain extended to $r = 2$. The finite-difference method was a second-order centered scheme written in polar coordinates. The vortex and finite-difference schemes used fourth-order Runge–Kutta time stepping. In all cases the grid resolution in the finite-difference and vortex domains were identical.

We have first compared the particle-grid domain decomposition with a finite-difference method using an artificial boundary condition at the outer boundary. Both methods are based on a velocity–vorticity formulation of the flow equations. Figure 10 shows the vorticity on the cylinder at time 5, and the velocity profile on the symmetry axis behind the cylinder, for a Reynolds number of 550. The finite-difference domain extended to $r = 7$. One can observe that the vorticity values are almost identical, while the velocity profiles for late times differ in the wake due to the artificial boundary condition used in the finite-difference solver.

Figure 11 is concerned with a Reynolds number of 3000. It shows the vorticity contours obtained by the particle-grid domain decomposition method at times 3.2 and 8.4. Figure 12 shows the drag history. These results are in excellent agreement with the results of Koumoutsakos and Leonard [15]. The finite-difference method used 120×620 grid points on a polar grid extending to two times the radius of a cylinder. The number of particles at time 8 was about 36,000. At that time, the vorticity support extends to about $r = 7.1$, which means that a pure finite-difference method with the same resolution would have needed more than 720 grid points in the radial direction, and thus about four times more computational elements than the domain decomposition method. This example, although in a case where the vorticity is fairly

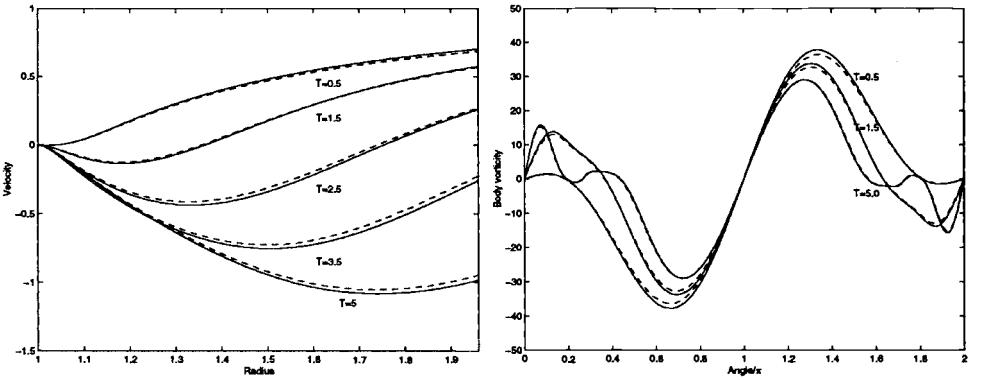


FIG. 10. *Flow past a cylinder at Reynolds* 550. *Vorticity values on the cylinder (right picture) and velocity in the back of the cylinder (left picture). Solid line: particle-grid domain decomposition; dashed line: finite-difference method.*

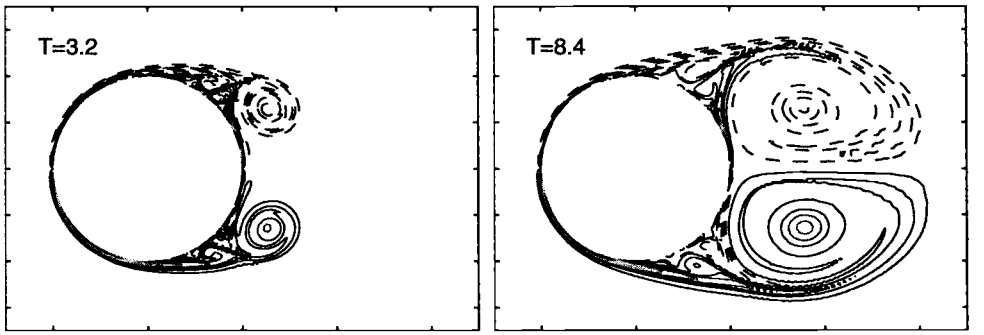


FIG. 11. *Flow past a cylinder at Reynolds number* 3000 *by a particle-grid domain decomposition. Vorticity contours.*

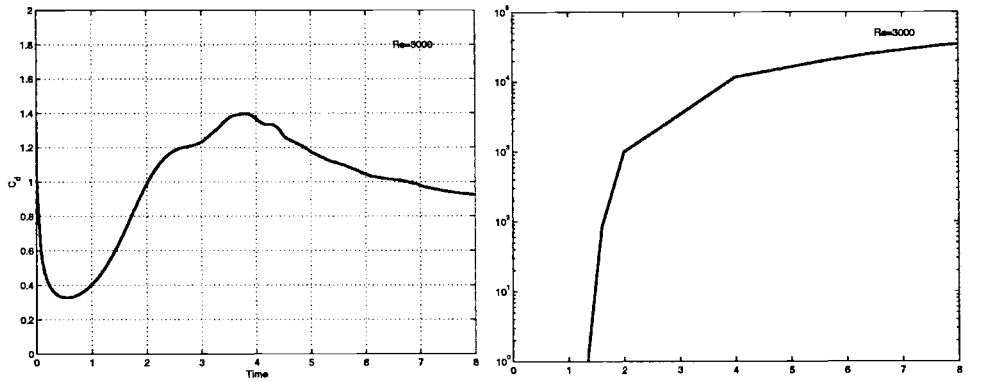


FIG. 12. *Flow past a cylinder at Reynolds number* 3000 *by a particle-grid domain decomposition. Drag history and number of particles.*

confined, illustrates the potential savings that the particle-grid method offers.

Our last examples are concerned with a particle-grid domain decomposition using a velocity-pressure formulation in the finite-difference domain. Figure 13 is a compar-

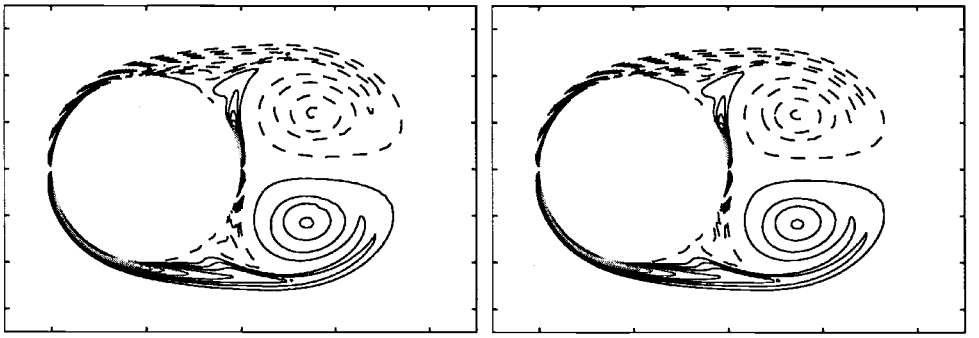


FIG. 13. *Vorticity contours for flow past a cylinder at Reynolds number* 1000. *Particle-grid domain decomposition based on a velocity–pressure formulation (left picture) and a velocity–vorticity formulation (right picture) in the finite-difference at time* $T = 6$.

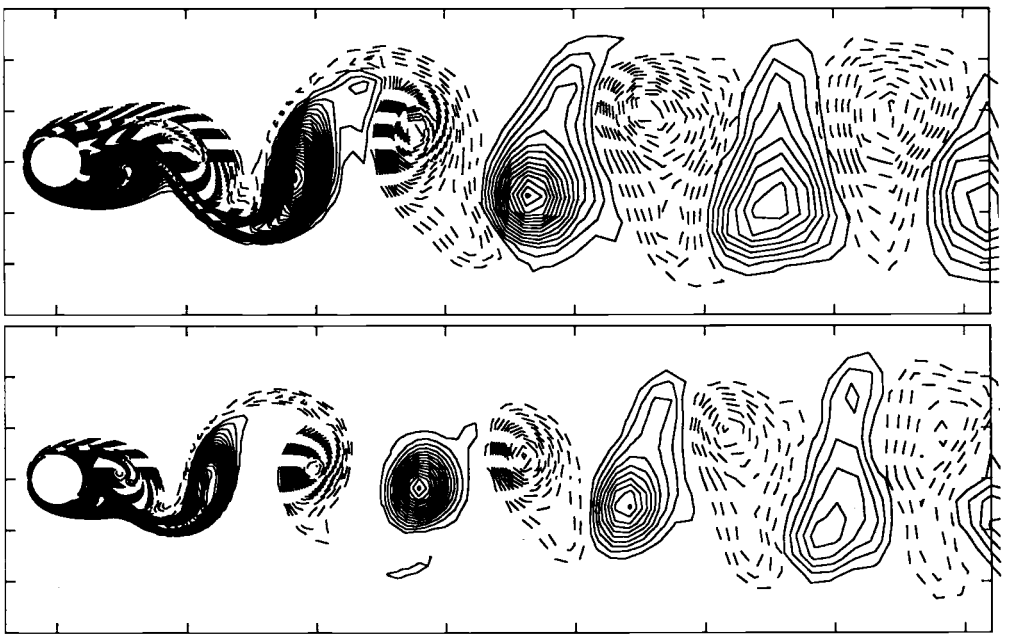


FIG. 14. *Karman streets behind a cylinder at Reynolds number of* 150 *(top picture) and* 300 *(bottom picture).*

ison of this technique with the domain-decomposition method based on a velocity–vorticity formulation in both domains at a Reynolds number of 1000. The vorticity contours are almost identical, which shows that combining different solvers based on different formulations of the equations does not introduce new difficulties, provided the interface conditions are written in a consistent way.

Finally, we now consider the case of a developed wake behind a cylinder. This is clearly the case where the vortex method is able to yield the most savings, as it uses computational elements only in the support of the vorticity and does not require any far-field artificial boundary condition. The finite-difference domain extended in this case to a six times the radius of a cylinder and the vortex domain started at three times

the radius of a cylinder. We have used a grid resolution with exponential stretching in the radial direction in both domains. The finite-difference domain extended to six times the radius of a cylinder and the vortex domain started at three times the radius of a cylinder. The vortex method used variable-size blobs with an exponential grid stretching in the radial direction (see [6, 7]). Figure 14 shows the Karman street obtained for Reynolds numbers of 150 and 300 at some late times. In the latter case the finite-difference grid used in the domain was 17,000 points, and the number of particles at the end of the simulation was slightly less than 12,000. The Strouhal numbers measured from these computations were, respectively, 0.183 and 0.208, in excellent agreement with the experimental values found by Williamson [26]. Note that, due to the loose resolution away from the cylinder, the far wake is obviously not accurately captured beyond about 10 diameters. However, this does not affect the computations of the forces on the body. In this case the particle-grid domain decomposition can be considered as a way to provide a good artificial boundary condition for the finite-difference solver for a limited cost.

5. Conclusion and future plans. We have presented a class of methods that combine grid-based and vortex schemes. High-order interpolation formulas are the key ingredients to transfer information between the two solvers at low cost and with minimal numerical dissipation.

VIC methods, where the grid essentially is used to compute the velocity fields and to take advantage of FFT-based fast Poisson solvers, provide robust and accurate solvers for flows in simple geometries. These features make them an attractive alternative to high-order grid-based methods, whenever it is crucial to compute complex dynamics on large time scales without numerical dissipation.

In two dimensions, stratified geophysical flows are good prototypes of such flows that could benefit from vortex methods. In three dimensions there are a number of flows in simple geometries that still lack accurate descriptions at high Reynolds numbers, in particular due to the time-step limitations that are in general implied by fine grids. Concerning 3D flows, work is underway to gain from classical homogeneous turbulence problems further insight into the effects at a subgrid level of the interpolations involved in grid-particle interpolations.

From another perspective, although vortex methods have proved that they are efficient in the simulation of complex vortex-wall interactions in two and three dimensions, particle-grid domain decomposition methods may be viewed as a more flexible tool, in particular in three dimensions when the grid solver near the obstacle is based on a velocity-pressure formulation. On the other hand, for 3D flows, it seems that pure finite-difference solvers still suffer from a dilemma between accuracy and stability. The numerical dissipation produced by upwind high-order schemes makes them inappropriate for large eddy simulations. Centered schemes do not have this difficulty, but they are very sensitive to grid regularity and to outflow boundary conditions [19]. Numerical schemes using vortex methods in the wake may be able to improve these aspects of current finite-difference solvers.

REFERENCES

[1] J. T. Beale, *On the accuracy of vortex methods at large time*, in Computational Fluid Dynamics and Reacting Gas Flows, B. Engquist, M. Luskin, and A. Majda, eds., Springer-Verlag, New York, 1988, pp. 19–32.

[2] A. J. Chorin, *Numerical solution of the Navier-Stokes equations*, Math. Comp., 22 (1968), pp. 745–762.

[3] J. P. Christiansen, *Vortex methods for flow simulation*, J. Comput. Phys., 13 (1973), p. 363–379.
[4] G.-H. Cottet, *Particle-grid domain decomposition methods for the Navier-Stokes equations in exterior domains*, in Vortex Dynamics and Vortex Methods, Lectures in Appl. Math. 28, AMS, Providence, RI, 1991, pp. 100–118.
[5] G.-H. Cottet, *A vorticity creation algorithm for the Navier-Stokes equations in arbitrary domain*, in Navier-Stokes Equations and Related Nonlinear Problems, Plenum, New York, 1995, pp. 335–349.
[6] G.-H. Cottet and P. Koumoutsakos, *Vortex Methods*, Cambridge University Press, Cambridge, 2000.
[7] G.-H. Cottet, P. Koumoutsakos, and M. L. Ould-Salihi, *Vortex methods with spatially varying cores*, J. Comput. Phys., 7 (1999), pp. 164–185.
[8] G.-H. Cottet, B. Michaux, S. Ossia, and G. VanderLinden, *A comparison of spectral and vortex methods in three dimensional incompressible flows*, J. Comput. Phys., submitted.
[9] G.-H. Cottet, M.-L. Ould-Salihi, and M. El-Hamraoui, *Multi-purpose regridding in vortex methods*, Vortex Flows and Related Numerical Methods, ESAIM Proc., 7 (1999), pp. 94–103; also available online from http://www.emath.fr/proc/vol.7.
[10] P. Degond and S. Mas-Gallic, *The weighted particle method for convection-diffusion equations*, Math. Comp., 53 (1989), pp. 485–526.
[11] Weinan E and J.-G. Liu, *Vorticity boundary conditions and related issues for finite difference schemes*, J. Comput. Phys., 124 (1996), pp. 368–382.
[12] Weinan E and J.-G. Liu, *Essentially compact schemes for unsteady viscous incompressible flows*, J. Comput. Phys., 126 (1996), pp. 122–138.
[13] J. L. Guermond, S. Huberson, and W. S. Shen, *Simulation of* 2D *external viscous flows by means of a domain decomposition method*, J. Comput. Phys., 108 (1993), pp. 343–352.
[14] J. L. Guermond and H. Z. Lu, *A Domain Decomposition Method for Simulating Advection Dominated, External Viscous Flows*, preprint, 1997.
[15] P. Koumoutsakos and A. Leonard, *High resolution simulations of the flow around an impulsively started cylinder using vortex methods*, J. Fluid Mech., 296 (1995), pp. 1–38.
[16] P. Koumoutsakos, A. Leonard, and F. Pepin, *Viscous boundary conditions for vortex methods*, J. Comput. Phys., 113 (1994), p. 52–61.
[17] P. Koumoutsakos and D. Shiels, *Simulations of the viscous flow normal to an impulsively started and uniformly accelerated flat plate*, J. Fluid Mech., 328 (1996), p. 177–227.
[18] M.-L. Ould-Salihi, *Couplage des méthodes numériques en simulation directe d'écoulements incompressibles*, Thèse de doctorat, Université Joseph Fourier, Grenoble, 1998.
[19] R. Mittal, *Large-eddy simulation of flow past a circular cylinder*, in CTR Annual Research Briefs, 1995, pp. 107–116.
[20] J. J. Monaghan, *Extrapolating B-splines for interpolation*, J. Comput. Phys., 60 (1985), pp. 253–262.
[21] P. Orlandi, *Vortex dipole rebound from a wall*, Phys. Fluids A, 2 (1990), pp. 1429–1436.
[22] L. Quartapelle, *Vorticity conditioning in the computation of two-dimensional viscous flows*, J. Comput. Phys., 40 (1981), pp. 453–477.
[23] I. J. Schoenberg, *Cardinal Spline Interpolation*, SIAM, Philadelphia, 1973.
[24] J. H. Strickland, L. A. Gritzo, R. S. Baty, G. F. Homicz, and S. F. Burns, *Fast multipole solvers for three-dimensional radiation and fluid problems*, ESAIM Proc., 7 (1999), pp. 408–417; also available online from http://www.emath.fr/proc/Vol.7.
[25] R. Verzicco and P. Orlandi, *Normal and oblique collision of a vortex ring with a wall*, Meccanica, 29 (1994), pp. 383–391.
[26] C. H. K. Williamson, 2D *and* 3D *aspects of the wake of a cylinder and their relation with wake computation*, in Vortex Dynamics and Vortex Methods, Lectures in Appl. Math. 28, AMS, Providence, RI, 1991, pp. 719–751.

SIAM J. SCI. COMPUT.
Vol. 22, No. 5, pp. 1675–1696
© 2000 Society for Industrial and Applied Mathematics

NUMERICAL SOLUTION OF THE TIME-DOMAIN MAXWELL EQUATIONS USING HIGH-ACCURACY FINITE-DIFFERENCE METHODS*

H. M. JURGENS† AND D. W. ZINGG†

Abstract. High-accuracy finite-difference schemes are used to solve the two-dimensional time-domain Maxwell equations for electromagnetic wave propagation and scattering. The high-accuracy schemes consist of a seven-point spatial operator coupled with a six-stage Runge–Kutta time-marching method. Two methods are studied, one of which produces the maximum order of accuracy and one of which is optimized for propagation distances smaller than roughly 300 wavelengths. Boundary conditions are presented which preserve the accuracy of these schemes when modeling interfaces between different materials. Numerical experiments are performed which demonstrate the utility of the high-accuracy schemes in modeling waves incident on dielectric and perfect-conducting scatterers using Cartesian and curvilinear grids. The high-accuracy schemes are shown to be substantially more efficient, in both computing time and memory, than a second-order and a fourth-order method. The optimized scheme can lead to a reduction in error relative to the maximum-order scheme, with no additional expense, especially when the number of wavelengths of travel is large.

Key words. computational electromagnetics, finite-difference schemes, wave propagation, phase error, Maxwell's equations

AMS subject classifications. 78M20, 65M06

PII. S1064827598334666

1. Introduction.

Numerical simulation of the propagation and scattering of electromagnetic waves has a wide range of applications in science and engineering, including antennas, microwave circuits, high-speed digital interconnects, all-optical devices, and many more [23]. The appropriate numerical algorithm for such simulations is dependent on the nature of the system being modeled. For geometries of low electrical size, i.e., spanning at most a few wavelengths, the method of moments is an efficient technique. However, scaling arguments presented by Petropoulos [18] show that this approach quickly becomes impractical as the electrical size increases. Geometries of moderate electrical size can be handled effectively using several available numerical methods for solving the Maxwell equations in the time domain. For such simulations, the geometric flexibility of the finite-element method can compensate for its relative inefficiency for hyperbolic equations. More efficient methods, which are effective for propagation distances on the order of 10 or 20 wavelengths, include the finite-difference methods of Yee [28] and Shang [20] and the finite-volume method of Mohammadian, Shankar, and Hall [16]. However, for propagation distances greater than 20 wavelengths, these methods, which are second-order accurate, typically require excessive grid densities with correspondingly large computational requirements.

The limitations of second-order methods in simulating wave phenomena have led to the development and application of higher-order and optimized finite-difference methods in several fields, including acoustics [13, 24] and seismology [10], as well as electromagnetics [12, 21, 25, 27, 29, 31, 34]. Higher-order methods offer increased accuracy for a given node density at the expense of increased cost per node. Optimized

*Received by the editors February 27, 1998; accepted for publication (in revised form) June 14, 2000; published electronically December 20, 2000.
http://www.siam.org/journals/sisc/22-5/33466.html
†Institute for Aerospace Studies, University of Toronto, 4925 Dufferin Street, Downsview, ON, Canada M3H 5T6 (hjurgens@celestica.com, dwz@oddjob.utias.utoronto.ca).

schemes sacrifice the order of accuracy in return for low error over an increased range of wavenumbers. Fourier analysis provides a simple means of analyzing the phase and amplitude errors of finite-difference methods [14, 26, 31, 34]. Several high-order and optimized schemes were compared by Zingg [30] using this approach. Zingg showed that one of the optimized compact schemes developed by Haras and Ta'asan [6] is capable of accurate simulations involving 200 wavelengths of travel with less than 4 grid points per wavelength. This scheme requires the solution of a pentadiagonal system of equations in each coordinate direction. The optimized noncompact schemes of Lockard, Brentner, and Atkins [15] and Zingg, Lomax, and Jurgens [34, 35] require less than 10 grid points per wavelength for 200 wavelengths of travel with a much lower cost per node. Numerical experiments are required to determine the relative efficiencies of these methods in a practical context.

In order to avoid the use of a very small time step, the accuracy of the time-marching method should be comparable to that of the spatial operator. For simulating wave phenomena, Runge–Kutta and Adams–Bashforth methods are natural candidates, with the former generally preferred due to their low memory requirements. For linear ordinary differential equations with constant coefficients, Runge–Kutta methods of up to fourth order require only two memory locations per dependent variable [31]. Five- and six-stage methods with the same memory requirement have been proposed by Haras and Ta'asan [6] and Zingg, Lomax, and Jurgens [34, 35].

The need for stable and accurate numerical boundary schemes presents a major obstacle in the application of high-order finite-difference methods. Since long propagation distances are generally associated with multiple interactions with media interfaces, the numerical boundary schemes can have a significant impact on the overall accuracy of a simulation, and hence on the relative accuracy of various methods. Progress in the development of numerical boundary schemes was reported in [4, 17, 33]. Another important consideration in the application of high-order methods is the choice of gridding strategy. Uniform Cartesian grids were used, for example, by Taflove and co-workers [23], while body-fitted curvilinear grids were favored by Shankar, Mohammadian, and Hall [22] and others. The Cartesian approach produces two significant advantages in the interior of the domain. Virtually all numerical methods are most accurate on a uniform grid. In addition, the need to deal with the metrics of a curvilinear coordinate transformation leads to a substantial penalty in terms of both speed and memory. On the other hand, it is extremely difficult to develop stable and accurate boundary treatments for higher-order methods on Cartesian grids. Furthermore, some geometries, such as a curved surface with a thin coating, are not well suited to Cartesian grids. Clearly, the choice of a gridding strategy is problem dependent.

A further issue in the development of numerical methods for the time-domain Maxwell equations is the need to truncate the domain, which inevitably leads to spurious reflections. Boundary conditions based on locally one-dimensional characteristic splitting generally produce excessive reflection. Thus, development of nonreflecting boundary conditions for hyperbolic problems has been an active area of research for many years. Important contributions have been made by Engquist and Majda [5], Bayliss and Turkel [2], Berenger [3], and Petropoulos, Zhao, and Cangellaris [19].

This paper presents the implementation and validation of the high-order and optimized finite-difference methods presented in [34, 35] for the solution of the time-domain Maxwell equations. The objective is to demonstrate the efficiency of these methods on nontrivial sample problems. The methods combine a noncompact spatial

operator with a seven-point stencil and a low-storage six-stage time-marching method. Two methods are studied, one of which produces the maximum order of accuracy and one of which is optimized for propagation distances less than roughly 300 wavelengths. In the next section, the Maxwell equations are presented in generalized curvilinear coordinates. The two finite-difference methods, including both spatial and temporal operators, are then presented. This is followed by a description of the treatment of interfaces and boundaries, including dielectric interfaces, perfect conductors, and domain boundaries. The two methods are applied to a number of test problems involving the propagation and scattering of electromagnetic waves. In order to demonstrate the efficiency of the present methods, the computational requirements are compared with two other methods, one fourth order in space, the other second order.

2. Governing equations. In two dimensions, Maxwell's equations decouple into two sets, the transverse magnetic (TM) and the transverse electric (TE) equations. Without any loss of physics, we consider only the TM set. We restrict our attention to linear, isotropic, perfect dielectric materials with constant properties, no charge density, and no current sources. Under these conditions, the TM equations can be written in the following form:

$$\frac{\partial \mathbf{Q}}{\partial t} + \mathbf{A}\frac{\partial \mathbf{Q}}{\partial x} + \mathbf{B}\frac{\partial \mathbf{Q}}{\partial y} = \mathbf{0}. \tag{2.1}$$

The vector of unknowns is given by

$$\mathbf{Q} = \begin{bmatrix} D_z \\ B_x \\ B_y \end{bmatrix}, \tag{2.2}$$

where D_z is the z component of the electric flux density, B_x and B_y are the x and y components of the magnetic flux density, and the two matrices are

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & -\dfrac{1}{\mu} \\ 0 & 0 & 0 \\ -\dfrac{1}{\varepsilon} & 0 & 0 \end{bmatrix}, \qquad \mathbf{B} = \begin{bmatrix} 0 & \dfrac{1}{\mu} & 0 \\ \dfrac{1}{\varepsilon} & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}; \tag{2.3}$$

μ is the magnetic permeability and ε is the electric permittivity. The constitutive relations are $\mathbf{D} = \varepsilon\mathbf{E}$ and $\mathbf{B} = \mu\mathbf{H}$, where $\mathbf{E}$ is the electric field intensity and $\mathbf{H}$ is the magnetic field intensity. In our tests, we assume a nondimensional form of the equations where, in free space, $\varepsilon = \mu = 1$. Note that the divergence relations are not explicitly enforced. However, numerous tests with the numerical methods studied in this paper show that the divergence of the magnetic field remains of the order of the truncation error for all time.

When numerically modeling electromagnetic fields, it is often the case that the geometry of the problem lends itself to a more generalized coordinate system than Cartesian. A method that allows the independent variables of the partial differential equations to be transformed into a uniformly spaced computational domain was given in [1]. The two-dimensional TM form of Maxwell's curl equations can be expressed in a generalized coordinate system using the following transformation:

$$\xi = \xi(x, y), \qquad \eta = \eta(x, y). \tag{2.4}$$

This allows the Cartesian coordinates to be transformed into curvilinear coordinates in such a way that a curvilinear grid will map to a uniform and square computational space with $\Delta\xi = \Delta\eta = 1$. The mapping is usually defined by assigning integer curvilinear coordinate values to each node in the grid and calculating the metrics of the transformation numerically. The transformed equations can be written in the following form:

$$\frac{\partial \mathbf{Q}}{\partial t} + \hat{\mathbf{A}}\frac{\partial \mathbf{Q}}{\partial \xi} + \hat{\mathbf{B}}\frac{\partial \mathbf{Q}}{\partial \eta} = \mathbf{0}, \tag{2.5}$$

where the two matrices are

$$\hat{\mathbf{A}} = \begin{bmatrix} 0 & \frac{1}{\mu}\frac{\partial \xi}{\partial y} & -\frac{1}{\mu}\frac{\partial \xi}{\partial x} \\ \frac{1}{\varepsilon}\frac{\partial \xi}{\partial y} & 0 & 0 \\ -\frac{1}{\varepsilon}\frac{\partial \xi}{\partial x} & 0 & 0 \end{bmatrix}, \qquad \hat{\mathbf{B}} = \begin{bmatrix} 0 & \frac{1}{\mu}\frac{\partial \eta}{\partial y} & -\frac{1}{\mu}\frac{\partial \eta}{\partial x} \\ \frac{1}{\varepsilon}\frac{\partial \eta}{\partial y} & 0 & 0 \\ -\frac{1}{\varepsilon}\frac{\partial \eta}{\partial x} & 0 & 0 \end{bmatrix}. \tag{2.6}$$

Extension to spatially variable material properties requires the use of a strong-conservation-law form of the equations, as discussed in [11].

3. Numerical method. Our aim is to solve for the total electric and magnetic field components by approximating Maxwell's equations using finite-difference methods. The finite-difference schemes used consist of a seven-point spatial operator in conjunction with an explicit six-stage time-marching method. These high-accuracy methods were originally presented by Zingg, Lomax, and Jurgens [35], and an extensive description of the methods, including an error and stability analysis, can be found in [34].

The spatial operator is made up of two parts, an antisymmetric or central-difference operator,

$$(\delta_x^a u)_j = \frac{1}{\Delta x}[a_1(u_{j+1} - u_{j-1}) + a_2(u_{j+2} - u_{j-2}) + a_3(u_{j+3} - u_{j-3})], \tag{3.1}$$

and a symmetric operator,

$$(\delta_x^s u)_j = \frac{1}{\Delta x}[d_0 u_j + d_1(u_{j+1} + u_{j-1}) + d_2(u_{j+2} + u_{j-2}) + d_3(u_{j+3} + u_{j-3})]. \tag{3.2}$$

The operator described by (3.1) has a maximum formal order which varies as Δx^6 when $a_1 = \frac{3}{4}$, $a_2 = -\frac{3}{20}$, and $a_3 = \frac{1}{60}$. The operator described by (3.2) has a maximum formal order which varies as Δx^5 when $d_1 = -\frac{3}{4}d_0$, $d_2 = \frac{3}{10}d_0$, and $d_3 = -\frac{1}{20}d_0$. The symmetric operator is used to add a small amount of numerical dissipation to the scheme. We use a value of $d_0 = \frac{1}{10}$. A characteristic splitting is used when the operator is applied to a hyperbolic system of equations. For example, in order to approximate the term $\mathbf{A}\frac{\partial \mathbf{Q}}{\partial x}$, the scheme is applied as follows:

$$\mathbf{A}\left(\frac{\partial \mathbf{Q}}{\partial x}\right)_j = \mathbf{A}(\delta_x^a \mathbf{Q})_j + |\mathbf{A}|(\delta_x^s \mathbf{Q})_j \tag{3.3}$$

at any interior node j. Note that $|\mathbf{A}| = \mathbf{X}|\Lambda|\mathbf{X}^{-1}$, where $\mathbf{X}$ is the matrix of right eigenvectors of $\mathbf{A}$, and Λ is the matrix of eigenvalues of $\mathbf{A}$. The method is applied in a similar way in order to approximate the y-derivatives of the field values.

The time-marching method, when applied to the ordinary differential equation $\frac{du}{dt} = f(u,t)$, can be written as

$$
\begin{aligned}
u^{(1)}_{n+\alpha_1} &= u_n + h\alpha_1 f_n, \\
u^{(2)}_{n+\alpha_2} &= u_n + h\alpha_2 f^{(1)}_{n+\alpha_1}, \\
u^{(3)}_{n+\alpha_3} &= u_n + h\alpha_3 f^{(2)}_{n+\alpha_2}, \\
u^{(4)}_{n+\alpha_4} &= u_n + h\alpha_4 f^{(3)}_{n+\alpha_3}, \\
u^{(5)}_{n+\alpha_5} &= u_n + h\alpha_5 f^{(4)}_{n+\alpha_4}, \\
u_{n+1} &= u_n + h f^{(5)}_{n+\alpha_5},
\end{aligned}
\tag{3.4}
$$

where $h = \Delta t$ is the time step, $t_n = nh$, $u_n = u(t_n)$, and $f^{(k)}_{n+\alpha} = f(u^{(k)}_{n+\alpha}, t_n + \alpha h)$. Setting $\alpha_5 = \frac{1}{2}$ gives a method that is second-order accurate. Additionally setting $\alpha_4 = \frac{1}{3}$, $\alpha_3 = \frac{1}{4}$, $\alpha_2 = \frac{1}{5}$, and $\alpha_1 = \frac{1}{6}$ gives the method with the highest formal order for linear homogeneous ordinary differential equations, which is sixth-order accurate. For inhomogeneous and nonlinear ordinary differential equations, it is second-order accurate. Two memory locations are required per dependent variable. The stability region for this method was shown in [34].

The above operators contain coefficients that can be determined in a number of ways. We have already described the operators in which the coefficients are determined by maximizing the formal order (referred to as the maximum-order (MO) operator). Zingg, Lomax, and Jurgens [34, 35] presented an additional method, in which the coefficients are determined by specifying that the phase and amplitude errors of the operators are minimized for waves which are resolved by the computational grid with at least 10 points per wavelength (referred to as the optimized operator or O10). The coefficients for this optimized scheme can be found in the appendix.

These schemes have a number of useful properties. First, the amplitude error is less than the phase error for all wavenumbers [34]. Hence any wavenumber components that are excessively damped also have excessive phase error. Second, the time-marching method produces significantly smaller errors than the spatial discretization. Hence the phase and amplitude errors are virtually independent of the Courant number (for Courant numbers less than unity). This contrasts with some other schemes that are accurate only within a narrow range of Courant numbers. Similarly, the present schemes do not require that the direction of wave propagation be aligned with the grid. The errors are largest for propagation angles of zero and 90 degrees. Hence the one-dimensional analysis given in [34] gives an upper bound on the error.

4. Interface and boundary treatment. In order to simulate the scattering of electromagnetic waves off of materials which have differing dielectric properties, each region is modeled as a distinct numerical domain. These domains are then coupled using the appropriate physical boundary conditions. This procedure allows the grids for each region to be generated independently, which is useful since the resolution of the grid is dependent on the material properties of the domain. Three separate boundary types are considered, an interface between two dielectric regions, the boundary between a dielectric and a perfect conductor, and the far-field boundary which is located at the outer region of the domain. At the interface between two dielectric media and at the surface of a perfect conductor, a locally one-dimensional characteristic formulation of the governing equations is used [8]. Using this approach,

characteristic variables can be found that represent incoming and outgoing waves along lines normal to the interface. For example, if we consider a boundary that lies at the right side of the domain on a line of constant x, the incoming waves are associated with $\mathbf{A}^-$ and the outgoing waves with $\mathbf{A}^+$, where

$$\mathbf{A}^{\pm} = \frac{\mathbf{A} \pm |\mathbf{A}|}{2} \tag{4.1}$$

and $|\mathbf{A}|$ is defined after (3.3). Similar expressions can be found for boundaries that lie on lines of constant y and for boundaries in curvilinear coordinates. For the curvilinear case, the incoming and outgoing waves will be associated with $\hat{\mathbf{A}}^{\pm}$ and $\hat{\mathbf{B}}^{\pm}$. When calculating spatial derivatives near boundaries and interfaces, flux-vector splitting is used. Regular fifth-order one-sided and upwind-biased schemes are used for the outgoing waves, while the following numerical boundary scheme (NBS) is used for the incoming waves:

$$\begin{aligned}(\delta_x u)_1 &= \frac{1}{60\Delta x}[-3u_0 - 119u_1 + 255u_2 - 240u_3 + 155u_4 - 57u_5 + 9u_6],\\ (\delta_x u)_2 &= \frac{1}{60\Delta x}[9u_0 - 66u_1 + 70u_2 - 60u_3 + 75u_4 - 34u_5 + 6u_6].\end{aligned} \tag{4.2}$$

The above NBS is a modified version of the one presented by Zingg, Lomax, and Jurgens [34, 35], which was found to be unstable on some curvilinear grids.

4.1. Dielectric interfaces. On the interface between two dielectrics the following conditions must hold:

$$E_t^{(1)} = E_t^{(2)}, \tag{4.3}$$

$$H_t^{(1)} = H_t^{(2)}, \tag{4.4}$$

$$D_n^{(1)} = D_n^{(2)}, \tag{4.5}$$

$$B_n^{(1)} = B_n^{(2)}. \tag{4.6}$$

The superscripts refer to the field values in materials (1) and (2). Note that these relations assume that there is no surface current flowing along the interface and that there is no surface charge density present at the interface. In the TM case, the electric intensity and electric flux density vectors point out of the x-y plane, and thus (4.5) holds automatically and (4.3) simplifies to

$$E_z^{(1)} = E_z^{(2)}. \tag{4.7}$$

Consider a body-fitted grid which ensures that the interface between different materials will lie on a ξ = constant curve. The grid is constrained to be orthogonal near the body, thus η = constant curves are normal to the interface. In this example, we assume that the positive ξ direction points from region (1) to region (2). Figure 1 shows an example of such a body-fitted coordinate system. We can generalize this example to different cases in curvilinear coordinates and to Cartesian coordinates quite easily. There are six unknown values which need to be stored on the interface: $D_z^{(1)}$, $D_z^{(2)}$, $B_x^{(1)}$, $B_x^{(2)}$, $B_y^{(1)}$, and $B_y^{(2)}$. The fields are extrapolated to the interface using a sixth-order method (see the appendix). The characteristic variables are then calculated from

$$w_+^{(1)} = Z^{(1)} D_z^{(1)} + B_t^{(1)} \tag{4.8}$$

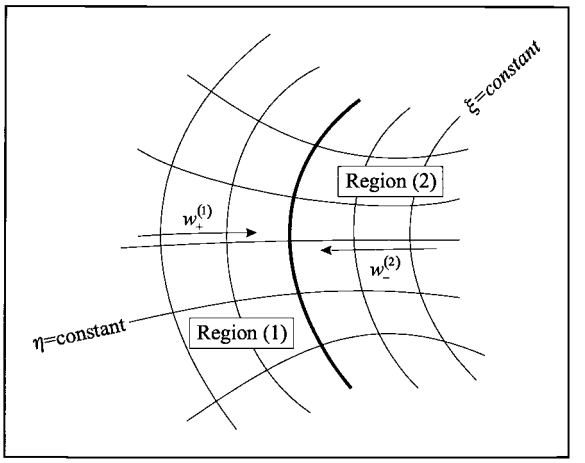


FIG. 1. *Example of a body-fitted grid showing the characteristic variables associated with the outgoing waves.*

and

$$w_-^{(2)} = Z^{(2)} D_z^{(2)} - B_t^{(2)}, \tag{4.9}$$

where $Z = (\mu/\varepsilon)^{1/2}$ is the intrinsic impedance of the medium. Note that

$$B_t = \frac{\dfrac{\partial \xi}{\partial y} B_x - \dfrac{\partial \xi}{\partial x} B_y}{\left[\left(\dfrac{\partial \xi}{\partial x} \right)^2 + \left(\dfrac{\partial \xi}{\partial y} \right)^2 \right]^{1/2}} \tag{4.10}$$

is the component of $\mathbf{B}$ tangent to a $\xi =$ constant curve, and

$$B_n = \frac{\dfrac{\partial \xi}{\partial x} B_x + \dfrac{\partial \xi}{\partial y} B_y}{\left[\left(\dfrac{\partial \xi}{\partial x} \right)^2 + \left(\dfrac{\partial \xi}{\partial y} \right)^2 \right]^{1/2}} \tag{4.11}$$

is the component of $\mathbf{B}$ normal to a $\xi =$ constant curve. This gives two equations for the unknown values $D_z^{(1)}$, $D_z^{(2)}$, $B_t^{(1)}$, and $B_t^{(2)}$. Note that the quantities extrapolated depend on the orientation of the local (ξ, η) coordinate system at the interface. Combining these with the jump conditions, we obtain

$$D_z^{(1)} = \frac{w_+^{(1)} + \dfrac{\mu^{(1)}}{\mu^{(2)}} w_-^{(2)}}{Z^{(1)} \left(1 + \dfrac{Z^{(1)}}{Z^{(2)}} \right)}, \tag{4.12}$$

$$D_z^{(2)} = \frac{\varepsilon^{(2)}}{\varepsilon^{(1)}} D_z^{(1)}, \tag{4.13}$$

$$B_t^{(1)} = \frac{Z^{(1)} w_+^{(1)} - \mu^{(1)} c^{(2)} w_-^{(2)}}{Z^{(1)} + Z^{(2)}}, \tag{4.14}$$

$$B_t^{(2)} = \frac{\mu^{(2)}}{\mu^{(1)}} B_t^{(1)}. \tag{4.15}$$

The variable $c = (\varepsilon\mu)^{-1/2}$ is the propagation speed. The normal component of the magnetic flux density is single valued on the interface, as shown by (4.6), and is determined by averaging the two values obtained by extrapolation. The values of B_x and B_y are determined from B_n and B_t using (4.10) and (4.11).

The above interface treatment was closely examined in [32]. The error introduced at interfaces is of the same order of magnitude as that introduced in the interior using the high-accuracy schemes. Both the interior discretization and the interface treatment, including the NBS (4.2), contribute to the overall error, with the relative contribution depending on the distance between interfaces.

4.2. Perfect conductors. At the interface between a dielectric material and a perfect conductor, the physical boundary conditions given in the above section must be modified. If, for example, region (2) is a perfect conductor—that is, having an infinite conductivity—then all the field values inside the region must be zero. Using this fact, and the four relations given in the above section, one obtains the following boundary conditions at the surface of a perfect conductor:

$$E_t = 0, \tag{4.16}$$

$$H_t = J_s, \tag{4.17}$$

$$D_n = 0, \tag{4.18}$$

$$B_n = 0. \tag{4.19}$$

In general, there will always be a finite current density, J_s, on the surface of a perfect conductor, but we have assumed a zero surface charge density. For the TM case, the third condition is automatically satisfied since the electric flux density points out of the plane, and the first condition sets its magnitude to zero, i.e., $D_z = 0$. The tangential component of the magnetic flux density, and thus the surface current, is found by extrapolating the fields to the surface and then calculating the outgoing characteristic variable. For example, if the surface of the perfect conductor lies along a grid line of constant ξ, with ξ increasing as the boundary is approached, then w_+ is determined from (4.8). Since $D_z = 0$, (4.8) gives

$$B_t = w_+, \tag{4.20}$$

where B_t is defined by (4.10). The Cartesian components of the magnetic flux density are determined using this value and the fact that $B_n = 0$.

4.3. Far-field boundaries. To model the problem of waves moving out into free space, the numerical domain must be artificially truncated. There is a considerable body of literature on the subject of far-field boundary conditions which tries to minimize the spurious reflection introduced when a disturbance propagates out of the numerical domain [2, 3, 5, 19, 23, 24]. There are many papers, such as [9], that discuss the merits of different types of far-field conditions.

We implement a modified version of the radiating boundary condition introduced by Bayliss and Turkel [2], which was found to be the best method of those tested by

Hixon, Shih, and Mankbadi [9]. This method truncates the domain by using a far-field radiation condition which only allows disturbances to propagate in the outward direction. The condition is applied by enforcing the following partial differential equation on the boundary region of the domain:

$$\left(\frac{1}{c}\frac{\partial}{\partial t}+\frac{\partial}{\partial r}+\frac{a}{2r}\right)\begin{bmatrix} D_z \\ B_x \\ B_y \end{bmatrix}=\mathbf{0}. \tag{4.21}$$

When $a = 1$ the operator given in [2] is recovered, which is $O(r^{-5/2})$. Using the operator with this value of a results in the numerical schemes becoming unstable for some curvilinear grids. We use a value of $a = 3$, which results in a stable and accurate method. The numerical domain is split into two regions, one of which contains the total electromagnetic field and one of which contains only the scattered field (see, for example, [23]). We store the scattered field on ghost-points which lie outside the domain, and the total field at all other nodes. To approximate the spatial derivatives in Maxwell's equations where the computational stencil will overlap the ghost-point region, the incident field is added to the scattered field at the necessary nodes in this region. When calculating the spatial derivatives for the radiation condition, the incident field is subtracted from the total field at the necessary nodes in the interior domain.

5. Results and discussion.

5.1. Dielectric square. We first simulate the scattering of a pulsed plane wave off of a dielectric square. Figure 2 shows an example grid as well as the geometry of the problem. The domain is square, with x and y ranging from 0 to 1, and the dielectric is located in the center of this region, at $0.4 \le x \le 0.6$, $0.4 \le y \le 0.6$. The permittivity of the dielectric is four times that of the free space region. The wavespeed in free space is normalized to one, resulting in a wavespeed of one-half in the dielectric. The domain is discretized using a Cartesian grid, in which $\Delta x = \Delta y$, and the grid density in the interior of the dielectric is twice that of free space. Results are obtained for a Gaussian pulse incident upon the dielectric square at an angle of 45°. The incident electric field is given by

$$E_z(x,y,t)=\exp\left[-\frac{1}{2\sigma^2}\left(x\cos\frac{\pi}{4}+y\sin\frac{\pi}{4}+\frac{1}{2}-t\right)^2\right] \tag{5.1}$$

with $\sigma = 0.03$. For all test cases, the time step is chosen in order to give a Courant number of unity in free space. This results in a value of Δt equal to the Δx value of the free space domain. As a result of both the wave speed and the grid spacing inside the dielectric being one half the values found outside the dielectric, the Courant number will also have a value of unity inside the dielectric. This ensures that the spatial resolution inside the dielectric is matched with that found in free space, since the width of the pulse will narrow as it slows down. Because of this, the errors will not be as localized to the interior of the dielectric as they were in the tests performed in section 5.1 of [11], where the grid resolution was kept constant over the entire domain. The better grid resolution inside the dielectric also allows for an accurate simulation of a narrower pulse. Note, however, that the accuracy of the high-accuracy finite-difference schemes is equally good at Courant numbers less than unity. Spurious reflections are not an issue for this case because the incident field is imposed along

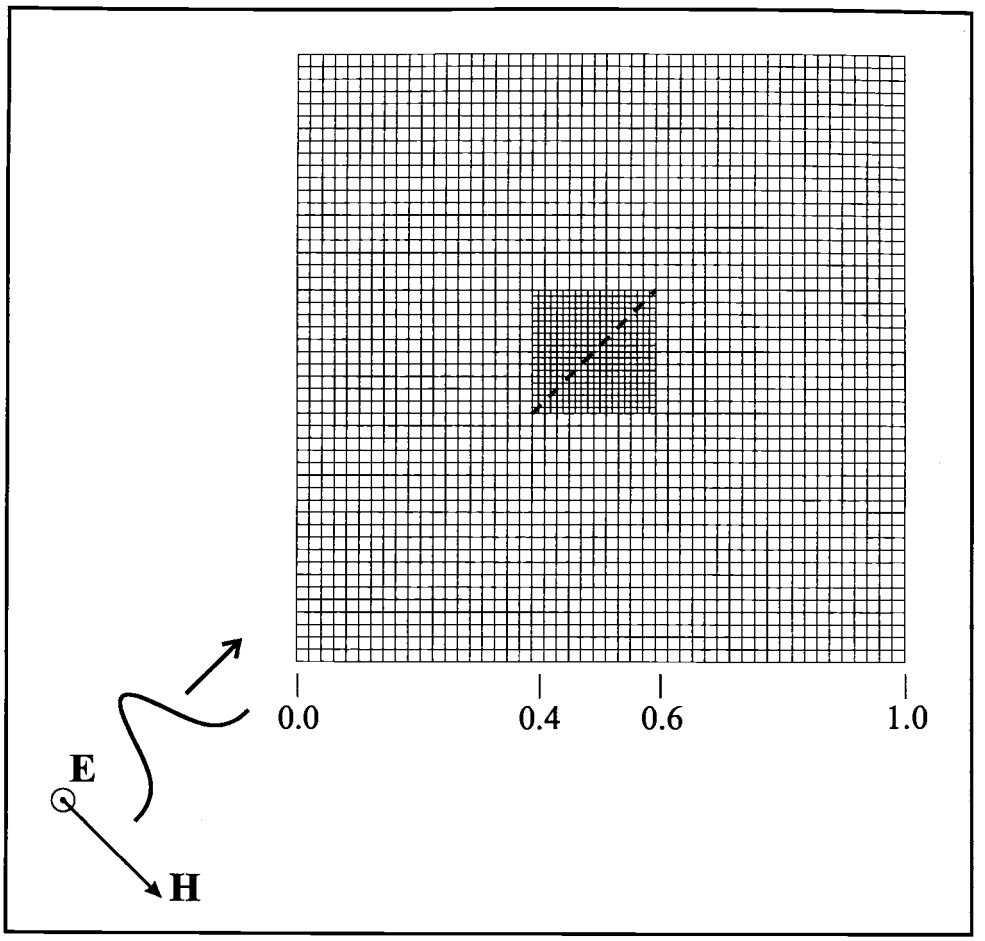


FIG. 2. *Example grid showing pulsed plane wave with Gaussian cross-section incident at* 45° *on a dielectric square.*

the entire outer boundary, and the simulation is not run long enough for the reflected pulse to reach the outer boundary.

For this simulation the interfaces between the different materials lie along lines in the Cartesian grid. The jump conditions discussed in section 4.1 can be used to obtain the field values along grid lines that cross the interfaces. In order to obtain the field unknowns along the interface at nodes inside the dielectric, which do not have a corresponding node outside the dielectric, interpolation and extrapolation of values from neighboring interface nodes are used. The interpolation and extrapolation schemes can be found in the appendix. Note that the extrapolation scheme and the noncentered interpolation schemes are used for the nodes near the corners of the dielectric. The corners must also be treated as special cases. It is only necessary to store the field values at the corner nodes for the free space part of the domain since the values in the interior of the dielectric are never used during the simulation. These values are obtained by taking the average value of the results of interpolating the field values along the x and y grid lines which intersect the corner node. Again, the relevant inerpolation schemes can be found in the appendix. Note that the presence of the

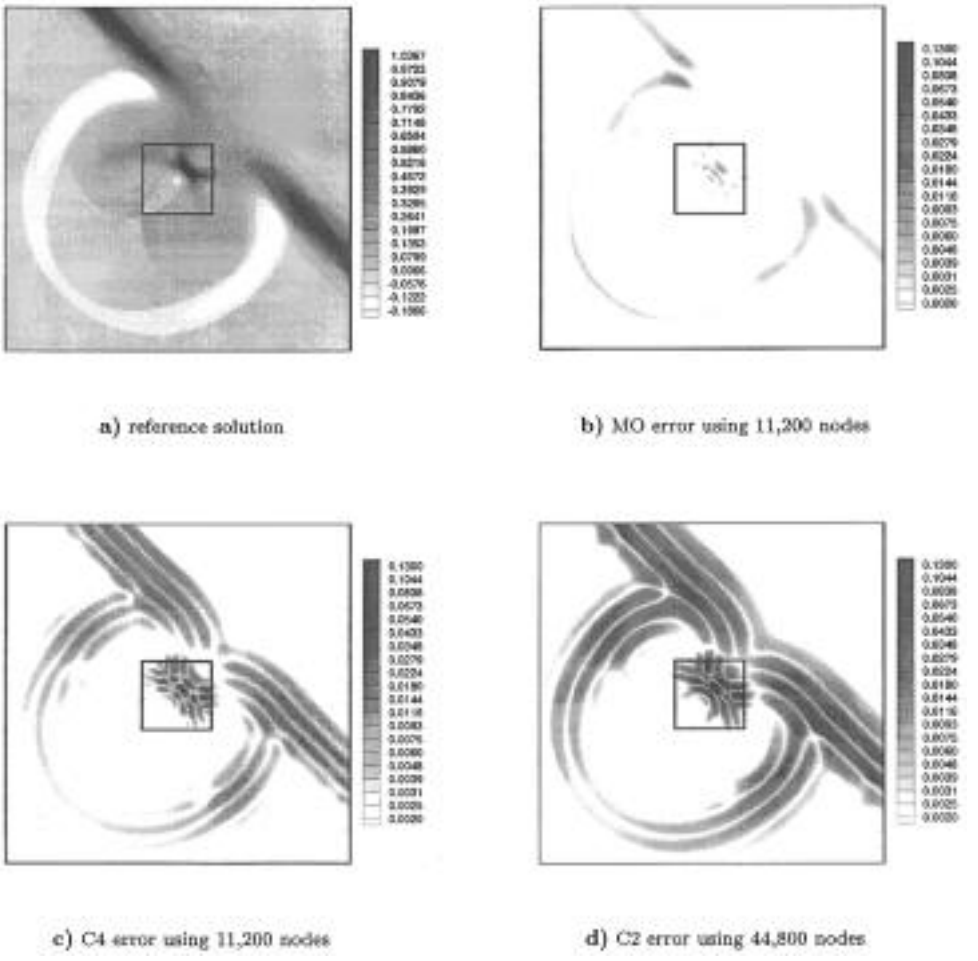


FIG. 3. *Contour plots of electric field intensity and errors in electric field intensity.* (a) *Reference solution,* (b) *MO error using* 11,200 *nodes,* (c) *C4 error using* 11,200 *nodes, and* (d) *C2 error using* 44,800 *nodes.*

singularity in the neighborhood of the corners introduces a low-order error locally. The finite-difference scheme can be modified locally to account for the singularity, but our results will show that the high-accuracy methods produce significant error reduction without implementing such a strategy.

In addition to the MO and O10 schemes described here, results are presented for second-order centered differences in space combined with fourth-order Runge–Kutta time marching (C2) and fourth-order centered differences in space combined with fourth-order Runge–Kutta time marching (C4).

Contour plots of the absolute value of the error in the electric field intensity at $t = 1.4$ are shown for three different cases in Figure 3. Figure 3(a) shows the electric field intensity for a reference solution calculated using the MO scheme on a grid with 179,200 nodes. The error obtained using the MO scheme on a grid with 11,200 nodes is shown in Figure 3(b). Note that the contour levels grow exponentially, and the error is extremely small. Plots of the errors obtained using method C4 on the grid with

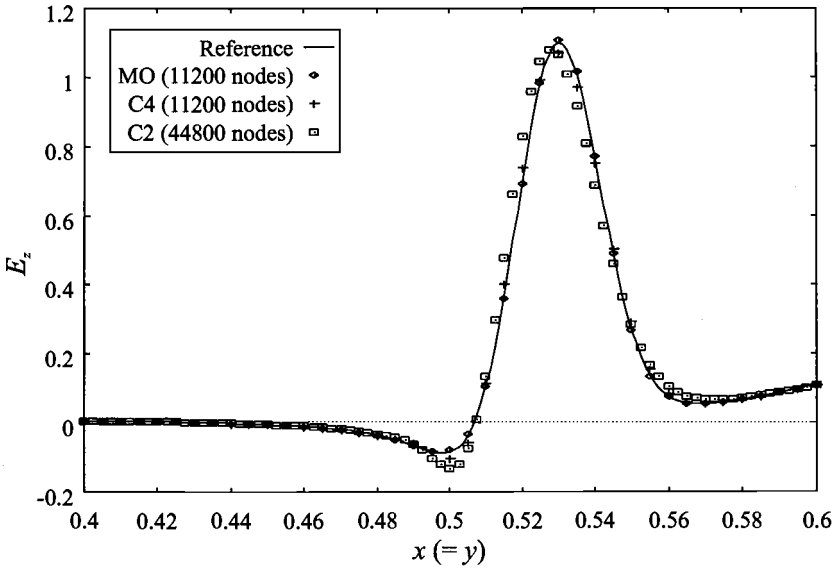


FIG. 4. *Electric field intensity across diagonal of dielectric square.*

11,200 nodes and method C2 on a grid with 44,800 nodes are shown in Figures 3(c) and (d), respectively. The errors obtained using method C4 are much larger than those obtained using the MO scheme, and even greater errors are evident in the C2 results, which are computed on a grid with four times as many nodes.

The electric field intensity along the diagonal of the dielectric square is shown in Figure 4 for the reference solution, the MO scheme on a grid with 11,200 nodes, the C4 scheme on a grid with 11,200 nodes, and the C2 scheme on a grid with 44,800 nodes. The results for the MO scheme lie very close to the reference solution. The C4 results are also fairly accurate, but there are some discernible deviations from the reference solution near the maximum and minimum field values. The solution obtained using C2 deviates considerably from the reference values.

Figure 5 compares the L_2 norm of the error in the electric field intensity, normalized by the number of nodes in the numerical domain, versus the number of nodes, for the C2, C4, MO, and O10 schemes. The errors are determined by considering the results obtained from a simulation on the highly resolved grid as an accurate reference solution. Since the number of nodes is inversely proportional to the mesh spacing squared, the C2 and C4 schemes display roughly second- and fourth-order convergence, respectively. This indicates that the effect of the corner singularity is not large. The MO and O10 schemes, which are formally second order as a result of the time-marching method, produce fourth-order accuracy over the range shown. However, our emphasis here is not on the asymptotic error behavior but on the grid density required to achieve specified error levels. The high-accuracy schemes show a significant advantage in this regard.

Figure 6 plots the error as a function of the computer time needed to obtain the solution. All schemes are implemented in a consistent manner. Hence computing time is an objective measure of cost, and similar relative performance is expected across computing platforms. It is clear that the high-accuracy schemes (MO and O10) are more efficient than the C4 and C2 schemes. To obtain the accuracy of the MO and

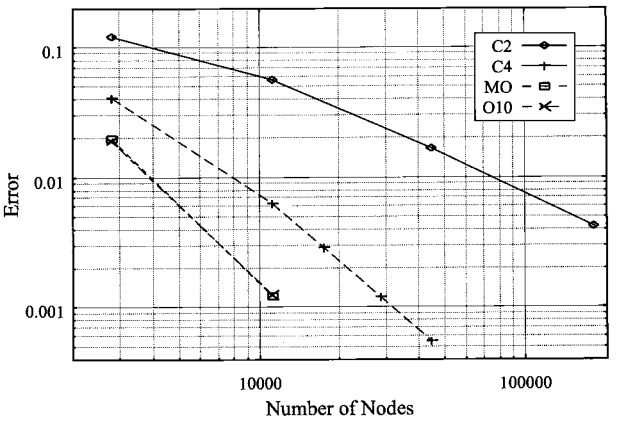


FIG. 5. L_2 *norm of the error in the electric field intensity for the dielectric square case, as a function of the number of nodes in the grid.*

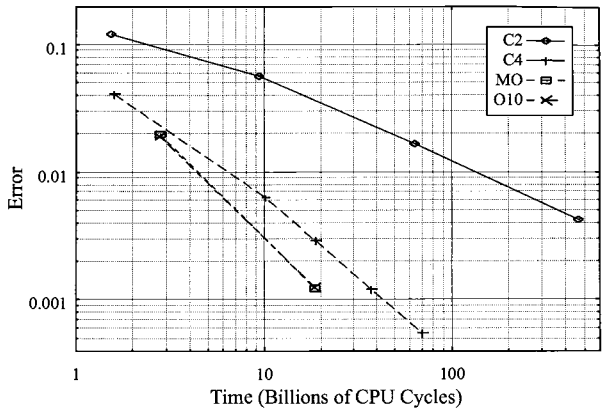


FIG. 6. L_2 *norm of the error in the electric field intensity for the dielectric square case, as a function of the CPU cycles necessary to complete simulation.*

O10 methods on a grid with 11,200 nodes, the C4 scheme needs to be run on a grid with 2.5 times as many grid nodes, requiring twice as much CPU time. The results obtained using C2 on a grid with 16 times as many nodes gives an error which is about 3.5 times as large as that obtained by the MO scheme, taking about 25 times longer to run. Even though the high-accuracy schemes take longer to run per grid node per time step, this is easily offset by the fact that smaller errors can be obtained by using a grid with far fewer nodes, and, as a result, a larger time step. The savings in memory is a direct result of using a grid with fewer nodes. The savings in both time and memory will be even more evident in three dimensions.

For the above cases the optimized method (O10) gives the same results as the MO scheme. This is primarily due to the nature of the Gaussian pulse, which has significant low wavenumber content. The majority of the Fourier components of the pulse have wavenumbers that give a value of $\kappa\Delta x < 0.35$ on the grid with 11,200 nodes. These are resolved more accurately by the MO scheme.

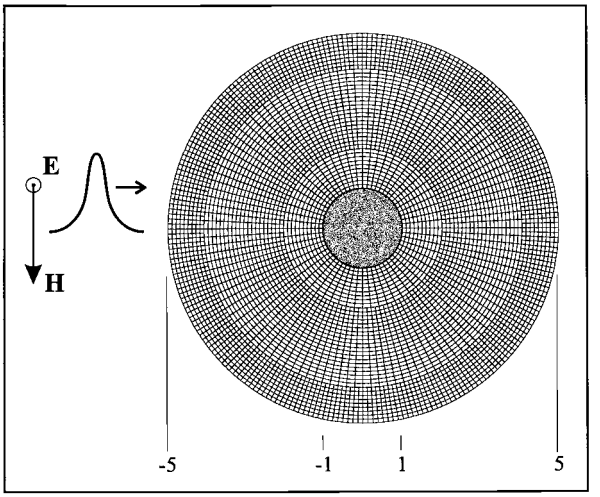


FIG. 7. *Example grid showing pulsed plane wave with Gaussian cross-section incident on a perfectly conducting cylinder.*

5.2. Perfectly conducting cylinder. We now consider an example using a curvilinear grid which consists of a waveform incident on a perfectly conducting cylinder. Figure 7 shows the geometry of the problem. The cylinder has a radius of unity, and the outer limit of the numerical domain is at $r = 5$. The speed at which a wave travels in free space is again normalized to unity. The waveform enters the domain from the left. The simulations described in section 5.2 of [12] are improved by modifying the grids so they are made up of three distinct zones. These grids, an example of which is shown in Figure 7, limit the widening of individual cells as one moves away from the scatterer, which allows the modeling of narrower pulses. This is a result of doubling the resolution of the grid, in the θ direction, whenever the value of r is doubled. This ensures that the waves are evenly resolved over the entire computational domain. The zones in the domain are allowed to overlap slightly in order to avoid the use of boundary operators where they interface. Interpolation must be used in order to obtain the field values at nodes that do not have corresponding nodes in the zone being overlapped. The interpolation schemes are described in the appendix. The grids are generated in order to obtain cells that have an aspect ratio close to 1 near the perfect conductor. When the aspect ratio deviates too much from this value, it is difficult to find boundary schemes that are stable and accurate for the high-accuracy methods. For all the tests, the time step is chosen so that the Courant number falls in the range 0.5 to 1.0. The grid metrics are calculated numerically using the same antisymmetric operator used in the calculation of the spatial derivatives of the electromagnetic fields. It should be emphasized that such a grid, i.e., one that produces a small range of Courant numbers, is not required in order to demonstrate the efficiency of the high-accuracy schemes, as shown in [12]. We have generated the grid in this manner simply to show that it can easily be done and to avoid a simulation in which the numerical errors are introduced primarily in one region of the grid.

5.2.1. Gaussian incident pulse. Consider a pulsed plane wave with a Gaussian cross-section incident on the cylinder. The incident electric field intensity is given by

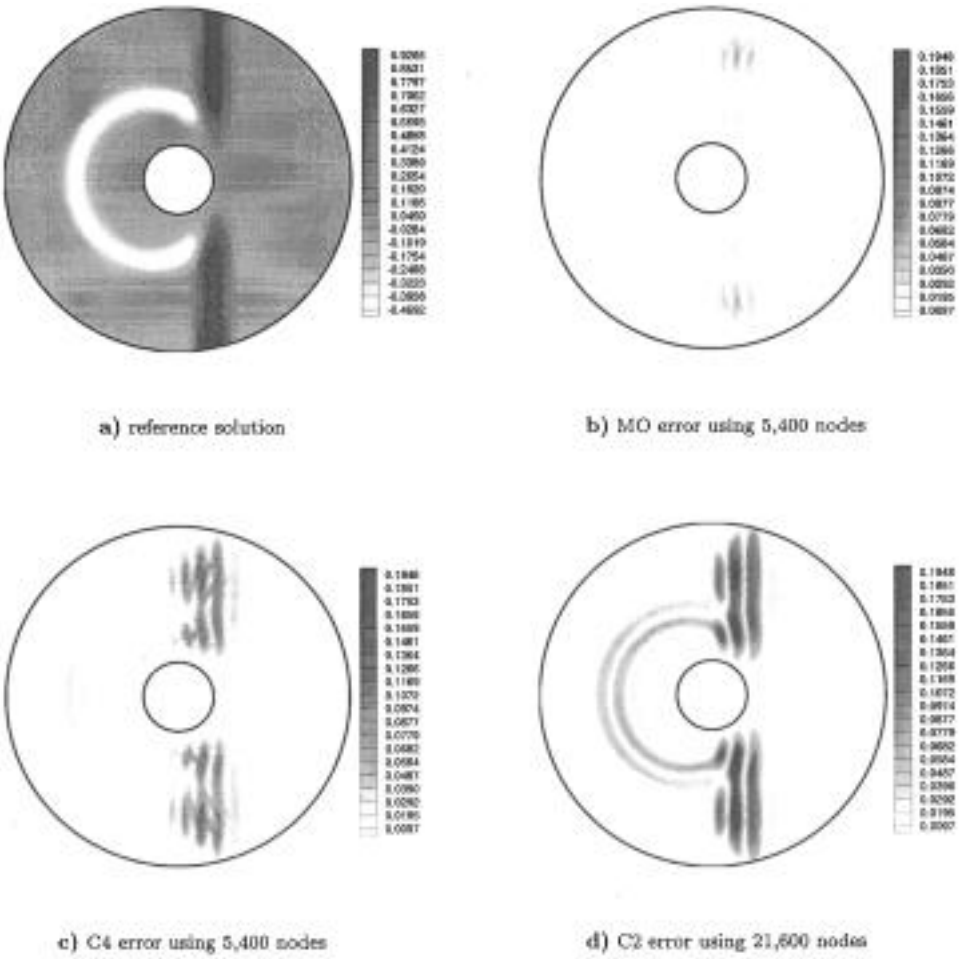


FIG. 8. *Contour plots of electric field intensity and errors in electric field intensity.* (a) *Reference solution,* (b) *MO error using* 5,400 *nodes,* (c) *C4 error using* 5,400 *nodes, and* (d) *C2 error using* 21,600 *nodes.*

$$E_z(x,y,t) = \exp\left[-\frac{1}{2\sigma^2}\left(x+\frac{15}{2}-t\right)^2\right] \tag{5.2}$$

with $\sigma = 0.3$, and the simulation is run until $t = 8.5$. Figure 8(a) shows a contour plot of the electric field intensity for a reference solution calculated using the MO scheme on a grid with 345,600 nodes. The absolute value of the error obtained using the MO scheme on a grid with 5,400 nodes is shown in Figure 8(b). Even though this grid is fairly coarse, the errors are seen to be very small. Plots of the errors obtained using method C4 on the grid with 5,400 nodes and method C2 on a grid with 21,600 nodes are shown in Figures 8(c) and (d), respectively. Both of these solutions show significant deviations from the reference solution.

The electric field intensity along the x-axis, behind the perfect conductor, is shown in Figure 9. In this region, the solution is not visible in Figure 8(a) and is very sensitive to numerical errors. The reference solution, the results from the MO

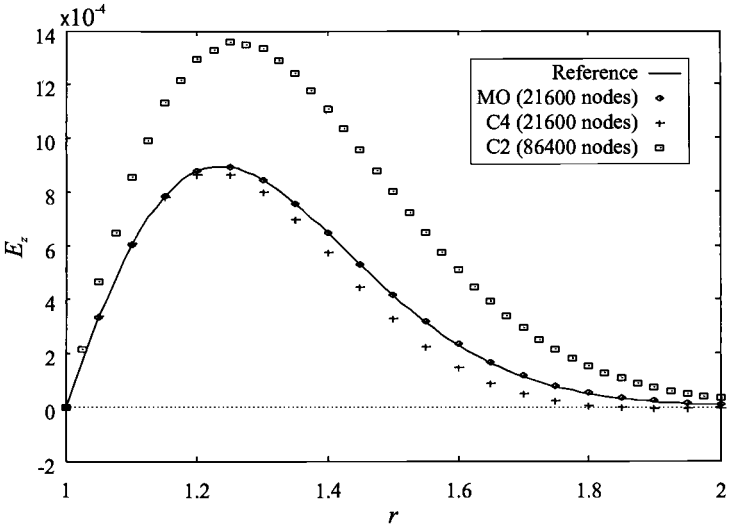


FIG. 9. *Electric field intensity along a radial line behind the perfectly conducting cylinder for an incident pulse with a Gaussian cross-section.*

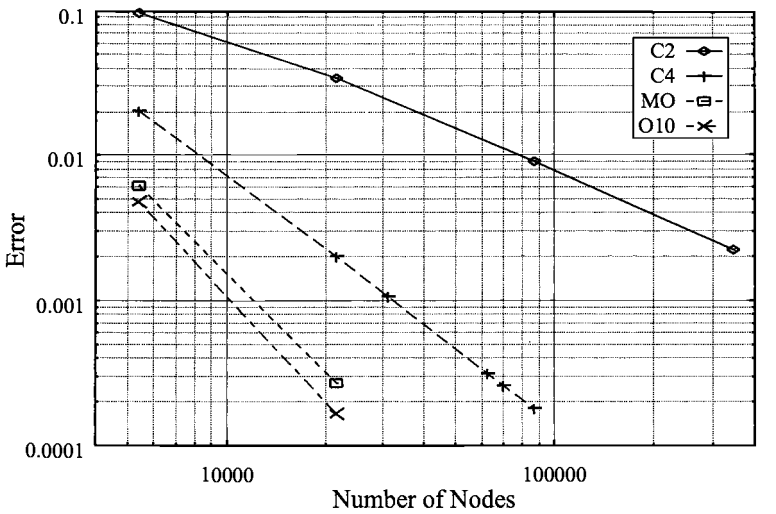


FIG. 10. L_2 *norm of the error in the electric field intensity as a function of the number of nodes in the grid for a pulsed plane wave incident on the perfectly conducting cylinder.*

scheme on a grid having 21,600 nodes, the results from the C4 scheme on a grid with 21,600 nodes, and the results from the C2 scheme on a grid with 86,400 nodes are all shown. The solution obtained using the MO scheme is very accurate. Using the same grid, the C4 method produces a solution that deviates significantly from the reference solution. The solution obtained by using C2 on a grid with four times as many nodes is not adequate for engineering purposes.

Figure 10 shows the normalized L_2 norm of the errors in the electric field intensity as a function of the number of nodes in the grid, and Figure 11 shows the error as a function of the CPU usage. The error value for each test is calculated using a reference

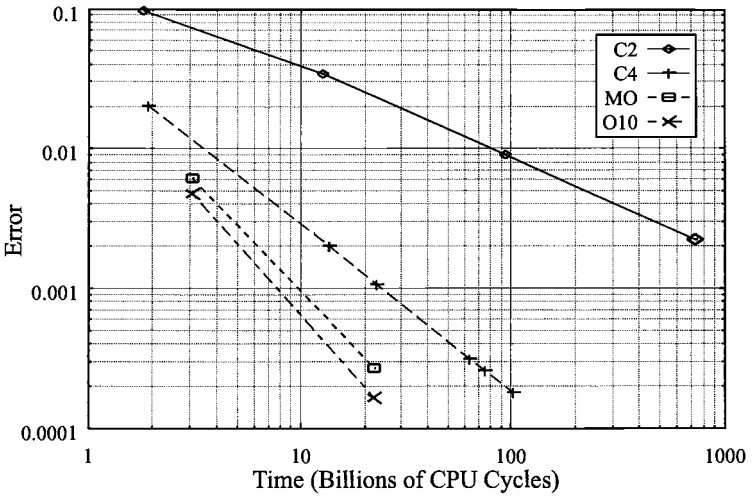


FIG. 11. L_2 *norm of the error in the electric field intensity as a function of the number of CPU cycles it takes to complete the simulation for a pulsed plane wave incident on the perfectly conducting cylinder.*

solution obtained using the MO scheme on a highly resolved grid. The trends shown in these figures are very similar to those obtained for the dielectric square case. To attain the same error level as the MO scheme used on a grid with 21,600 nodes, the C4 scheme must use about 3.25 times as many nodes and takes about 3.4 times as long to complete the simulation. Even when using a grid with 345,600 nodes, the C2 method does not produce a solution that can compare with those computed by the high-accuracy methods on a grid with 21,600 nodes. The O10 scheme is seen to produce slightly better results than the MO scheme when using the same grid. For example, on the grid with 21,600 nodes, the O10 scheme results in an error that is 61% of that obtained using the MO scheme, with no additional CPU expense.

5.2.2. Cosine incident wave. In this section, simulations of a cosine plane wave incident on the perfectly conducting cylinder are presented. The existence of an analytical solution for a plane harmonic electromagnetic wave incident upon a perfectly conducting circular cylinder is well known and can be found in, for example, [7]. The incident electric field intensity is given by

$$E_z(x, y, t) = \cos \kappa(x - t) \tag{5.3}$$

with $\kappa = 2\pi$. A shaded contour plot of the analytical solution for the total electric field intensity is shown in Figure 12.

Figure 13 compares the errors as a function of the number of nodes in the grids. Figure 14 plots these errors as a function of the CPU time needed to complete the simulations. These plots again show the superior efficiency of the high-accuracy methods. The second-order method produces significant errors for all three of the grids. The fourth-order method needs to use the most resolved grid to generate a solution with acceptable accuracy, which can be produced by the high-accuracy methods on a grid with four times fewer nodes. Figure 15 shows the electric field intensity along a radial line directly behind the scatterer for the analytical, fourth-order, and MO solutions. The numerical solutions were obtained on a grid with 21,600 nodes. The solution ob-

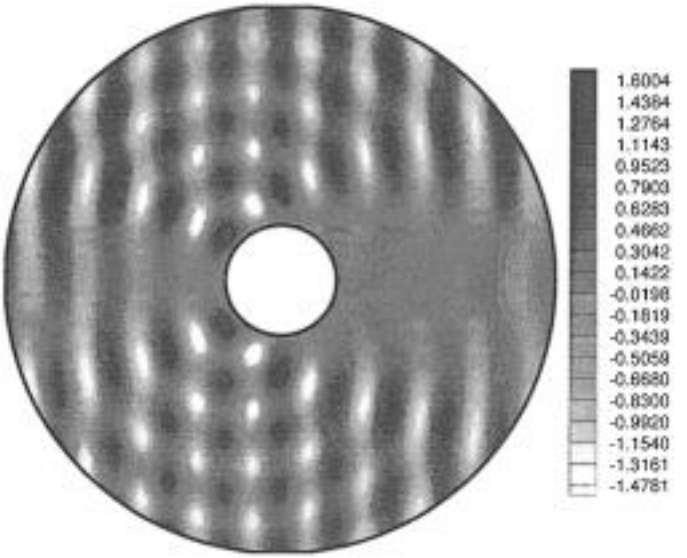


FIG. 12. *Electric field intensity for cosine wave, with* $\kappa = 2\pi$*, incident on a perfectly conducting cylinder.*

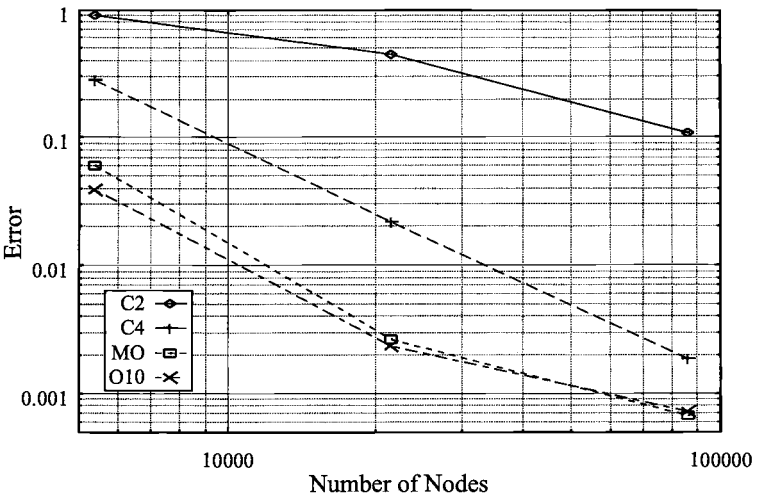


FIG. 13. L_2 *norm of the error in the electric field intensity as a function of the number of nodes in the grid for a cosine wave, with* $\kappa = 2\pi$*, incident on the perfectly conducting cylinder.*

tained using the MO method is considerably more accurate than that obtained using the fourth-order method.

The optimized scheme does not show significant improvement over the maximum-order scheme in the above tests because of the relatively short distance of travel for the waves being considered. To test longer distances of travel (in terms of wavelength) a highly resolved grid with 345,600 nodes is used, and incident waves with much higher wavenumbers are modeled. With $\kappa = 8\pi$, the wave is resolved with 10 to 20 PPW and travels 32 wavelengths during the simulation. For this case, the optimized method produces an error that is about 40% of that produced by the MO scheme. With

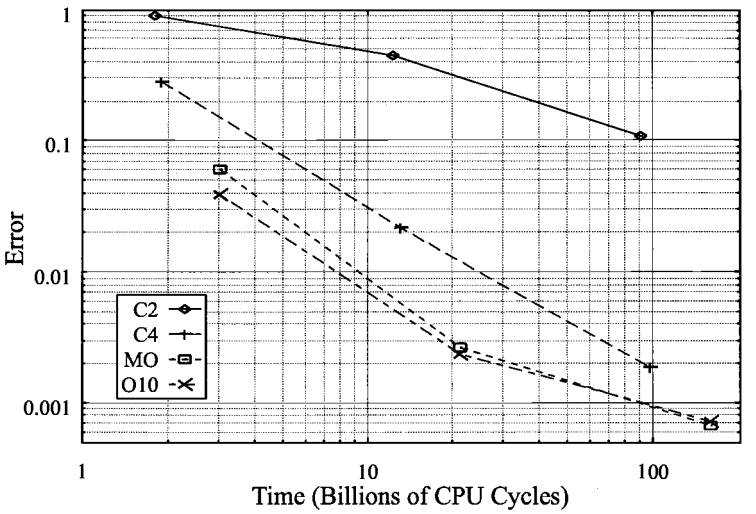


FIG. 14. L_2 *norm of the error in the electric field intensity as a function of the CPU time necessary to complete the simulation for a cosine wave, with* $\kappa = 2\pi$, *incident on the perfectly conducting cylinder.*

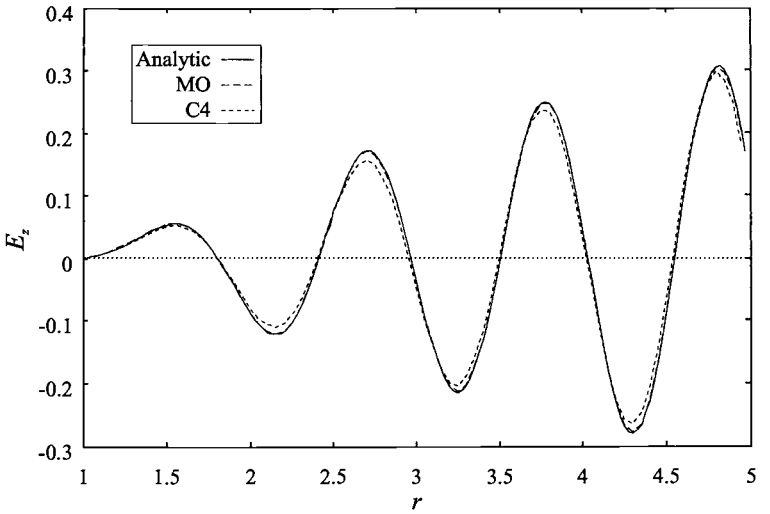


FIG. 15. *Electric field intensity along a radial line behind the perfectly conducting cylinder for an incident cosine wave with* $\kappa = 2\pi$.

$\kappa = 10\pi$, the resolution is 8 to 16 PPW, and the distance of travel is 40 wavelengths. For this wavenumber, the optimized method produces an error that is about 25% of that produced by the maximum-order scheme. This shows that the benefits of the optimized scheme increase as the number of wavelengths traveled between reflections increases.

6. Conclusions. Two high-accuracy finite-difference schemes have been used to solve the two-dimensional time-domain Maxwell equations to simulate electromagnetic wave propagation and scattering. The schemes have successfully modeled perfectly

conducting and dielectric scatterers on Cartesian and curvilinear grids. The high-accuracy methods have been shown to be more efficient than classical second-order and fourth-order methods in terms of computing time and memory. The optimized scheme can produce some error reduction relative to the maximum-order scheme with no additional expense, especially when the number of wavelengths of travel is large. Overall, the results demonstrate that these methods are an efficient option for simulating the propagation and scattering of high-frequency electromagnetic waves.

Appendix. The following are the coefficients of the optimized scheme:

$$a_1 = 0.759961, \quad a_2 = -0.158122, \quad a_3 = 0.0187609,$$
$$d_0 = 0.1, \quad d_1 = -0.0763846, \quad d_2 = 0.0322896, \quad d_3 = -0.005905,$$
$$\alpha_1 = 0.168850, \quad \alpha_2 = 0.197348, \quad \alpha_3 = 0.250038, \quad \alpha_4 = 0.333306, \quad \alpha_5 = 0.5.$$

The following are the extrapolation and interpolation schemes used along interfaces:

$$u_j = 6u_{j+1} - 15u_{j+2} + 20u_{j+3} - 15u_{j+4} + 6u_{j+5} - u_{j+6},$$
$$u_{j+1/2} = (3u_{j-2} - 25u_{j-1} + 150u_j + 150u_{j+1} - 25u_{j+2} + 3u_{j+3})/256,$$
$$u_{j+1/2} = (-7u_{j-1} + 105u_j + 210u_{j+1} - 70u_{j+2} + 21u_{j+3} - 3u_{j+4})/256,$$
$$u_{j+1/2} = (63u_j + 315u_{j+1} - 210u_{j+2} + 126u_{j+1} - 45u_{j+4} + 7u_{j+5})/256,$$
$$u_j = \frac{3}{4}(u_{j-1} + u_{j+1}) - \frac{3}{10}(u_{j-2} + u_{j+2}) + \frac{1}{20}(u_{j-3} + u_{j+3}).$$

The following can be used to calculate the absolute value of the flux Jacobians and the split flux Jacobians, for example, $|\hat{\mathbf{A}}| = |\hat{\mathbf{J}}(\xi)|$ and $\hat{\mathbf{B}}^{\pm} = \hat{\mathbf{J}}^{\pm}(\eta)$:

$$|\hat{\mathbf{J}}(k)| = \begin{bmatrix} c|\nabla k| & 0 & 0 \\ 0 & c\frac{1}{|\nabla k|}\left(\frac{\partial k}{\partial y}\right)^2 & c\frac{1}{|\nabla k|}\frac{\partial k}{\partial x}\frac{\partial k}{\partial y} \\ 0 & -c\frac{1}{|\nabla k|}\frac{\partial k}{\partial x}\frac{\partial k}{\partial y} & c\frac{1}{|\nabla k|}\left(\frac{\partial k}{\partial x}\right)^2 \end{bmatrix},$$

$$\hat{\mathbf{J}}^{\pm}(k) = \frac{1}{2}\begin{bmatrix} \pm c|\nabla k| & \frac{1}{\mu}\frac{\partial k}{\partial y} & -\frac{1}{\mu}\frac{\partial k}{\partial x} \\ \frac{1}{\varepsilon}\frac{\partial k}{\partial y} & \pm c\frac{1}{|\nabla k|}\left(\frac{\partial k}{\partial y}\right)^2 & \mp c\frac{1}{|\nabla k|}\frac{\partial k}{\partial x}\frac{\partial k}{\partial y} \\ -\frac{1}{\varepsilon}\frac{\partial k}{\partial x} & \mp c\frac{1}{|\nabla k|}\frac{\partial k}{\partial x}\frac{\partial k}{\partial y} & \pm c\frac{1}{|\nabla k|}\left(\frac{\partial k}{\partial x}\right)^2 \end{bmatrix},$$

where

$$|\nabla k| = \sqrt{\left(\frac{\partial k}{\partial x}\right)^2 + \left(\frac{\partial k}{\partial y}\right)^2}.$$

The following are the local characteristic variables in the ξ (use $k = \xi$) and η (use $k = \eta$) directions:

$$w_+ = ZD_z + \left(\frac{\partial k}{\partial y}B_x - \frac{\partial k}{\partial x}B_y\right) \Big/ |\nabla k|,$$

$$w_- = ZD_z - \left(\frac{\partial k}{\partial y}B_x - \frac{\partial k}{\partial x}B_y\right) \Big/ |\nabla k|.$$

REFERENCES

[1] D. A. Anderson, J. C. Tannehill, and R. H. Pletcher, *Computational Fluid Mechanics and Heat Transfer*, Hemisphere, New York, 1984.

[2] A. Bayliss and E. Turkel, *Radiation boundary conditions for wave-like equations*, Comm. Pure Appl. Math., 33 (1980), pp. 707–725.

[3] J.-P. Berenger, *A perfectly matched layer for the absorption of electromagnetics wave*, J. Comput. Phys., 114 (1994), pp. 185–200.

[4] M. H. Carpenter, D. Gottlieb, and S. Abarbanel, *Time-stable boundary conditions for finite-difference schemes solving hyperbolic systems: methodology and application to high-order compact schemes*, J. Comput. Phys., 111 (1994), pp. 220–236.

[5] B. Engquist and A. Majda, *Absorbing boundary conditions for the numerical simulation of waves*, Math. Comp., 31 (1977), pp. 629–651.

[6] Z. Haras and S. Ta'asan, *Finite difference schemes for long-time integration*, J. Comput. Phys., 114 (1994), pp. 265–279.

[7] R. F. Harrington, *Time-Harmonic Electromagnetic Fields*, McGraw-Hill, New York, 1961.

[8] C. Hirsch, *Numerical Computation of Internal and External Flows*, Vol. 2: *Computational Methods for Inviscid and Viscous Flows*, Wiley, London, 1990.

[9] R. Hixon, S.-H. Shih, and R. R. Mankbadi, *Evaluation of boundary conditions for computational aeroacoustics*, American Institute of Aeronautics and Astronautics J., 33 (1995), pp. 2006–2012.

[10] O. Holberg, *Computational aspects of the choice of operator and sampling interval for numerical differentiation in large-scale simulation of wave phenomena*, Geophysical Prospecting, 35 (1987), pp. 629–655.

[11] H. M. Jurgens, *High-Accuracy Finite-Difference Schemes for Linear Wave Propagation*, Ph.D. thesis, University of Toronto, ON, Canada, 1997.

[12] H. M. Jurgens and D. W. Zingg, *Implementation of a high-accuracy finite-difference scheme for linear wave phenomena*, in Proceedings of the Third International Conference on Spectral and High Order Methods, Houston J. Math., Houston, TX, 1995, pp. 363–372.

[13] S. K. Lele, *Compact finite difference schemes with spectral-like resolution*, J. Comput. Phys., 103 (1992), pp. 16–42.

[14] Y. Liu, *Fourier analysis of numerical algorithms for the Maxwell equations*, J. Comput. Phys., 124 (1996), pp. 396–416.

[15] D. P. Lockard, K. S. Brentner, and H. L. Atkins, *High-accuracy algorithms for computational aeroacoustics*, American Institute of Aeronautics and Astronautics J., 33 (1995), pp. 246–251.

[16] A. H. Mohammadian, V. Shankar, and W. F. Hall, *Computation of electromagnetic scattering and radiation using a time-domain finite-volume discretization procedure*, Comput. Phys. Comm., 68 (1991), pp. 262–281.

[17] P. Olsson, *Summation by parts, projections, and stability* I, Math. Comp., 64 (1995), pp. 1035–1065.

[18] P. G. Petropoulos, *Phase error control for FD-TD methods of second and fourth order accuracy*, IEEE Trans. Antennas and Propagation, 42 (1994), pp. 859–862.

[19] P. G. Petropoulos, L. Zhao, and A. C. Cangellaris, *A reflectionless sponge layer absorbing boundary condition for the solution of Maxwell's equations with high-order staggered finite difference schemes*, J. Comput. Phys., 139 (1998), pp. 184–208.

[20] J. S. Shang, *A fractional-step method for the time domain Maxwell equations*, J. Comput. Phys., 118 (1995), pp. 109–119.

[21] J. S. Shang and D. Gaitonde, *On High Resolution Schemes for Time-Dependent Maxwell Equations*, American Institute of Aeronautics and Astronautics paper 96-0832, 1996.

[22] V. Shankar, A. H. Mohammadian, and W. F. Hall, *A time-domain finite-volume treatment for the Maxwell equations*, Electromagnetics, 10 (1990), pp. 127–145.
[23] A. Taflove, *Computational Electrodynamics: The Finite-Difference Time-Domain Method*, Artech House, Norwood, MA, 1995.
[24] C. K. W. Tam and J. C. Webb, *Dispersion-relation-preserving finite difference schemes for computational acoustics*, J. Comput. Phys., 107 (1993), pp. 262–281.
[25] E. Turkel and A. Yefet, *Fourth order method for Maxwell equations on a staggered mesh*, IEEE Antennas and Propagation Society International Symposium 1997 Digest, 4 (1997), pp. 2156–2159.
[26] R. Vichnevetsky and J. B. Bowles, *Fourier Analysis of Numerical Approximations of Hyperbolic Equations*, SIAM, Philadelphia, 1982.
[27] B. Yang, D. Gottlieb, and J. S. Hesthaven, *Spectral simulations of electromagnetic wave scattering*, J. Comput. Phys., 134 (1997), pp. 216–230.
[28] K. S. Yee, *Numerical solution of initial boundary value problems involving Maxwell's equations in isotropic media*, IEEE Trans. Antennas and Propagation, 14 (1966), pp. 302–307.
[29] J. L. Young, D. Gaitonde, and J. S. Shang, *Toward the construction of a fourth-order difference scheme for transient EM wave simulation: Staggered grid approach*, IEEE Trans. Antennas and Propagation, 45 (1997), pp. 1573–1580.
[30] D. W. Zingg, *Comparison of high-accuracy finite-difference methods for linear wave propagation*, SIAM J. Sci. Comput., 22 (2000), pp. 476–502.
[31] D. W. Zingg and T. T. Chisholm, *Runge-Kutta methods for linear ordinary differential equations*, Appl. Numer. Math., 31 (1999), pp. 227–238.
[32] D. W. Zingg, P. D. Giansante, and H. M. Jurgens, *Experiments with High-Accuracy Finite-Difference Schemes for the Time-Domain Maxwell Equations*, American Institute of Aeronautics and Astronautics paper 94-0232, 1994.
[33] D. W. Zingg and H. Lomax, *On the eigensystems associated with numerical boundary schemes for hyperbolic equations*, in Numerical Methods for Fluid Dynamics, M. J. Baines and K. W. Morton, eds., Clarendon Press, Oxford, UK, 1993, pp. 473–481.
[34] D. W. Zingg, H. Lomax, and H. Jurgens, *High-accuracy finite-difference schemes for linear wave propagation*, SIAM J. Sci. Comput., 17 (1996), pp. 328–346.
[35] D. W. Zingg, H. Lomax, and H. Jurgens, *An Optimized Finite-Difference Scheme for Wave Propagation Problems*, American Institute of Aeronautics and Astronautics paper 93-0459, 1993.

SIAM J. SCI. COMPUT.
Vol. 22, No. 5, pp. 1697–1716
© 2000 Society for Industrial and Applied Mathematics

SUPERCONVERGENT DEFERRED CORRECTION METHODS FOR FIRST ORDER SYSTEMS OF NONLINEAR TWO-POINT BOUNDARY VALUE PROBLEMS*

MARNIX VAN DAELE† AND JEFF R. CASH‡

Abstract. Iterated deferred correction is a widely used approach to the numerical solution of first order systems of nonlinear two-point boundary value problems. Normally the orders of accuracy of the various methods used in a deferred correction scheme differ by 2, and, as a direct result, each time a deferred correction is applied the order of the overall scheme is increased by a maximum of 2. In this paper we consider the construction of mono-implicit Runge–Kutta (MIRK) methods where an increase of four orders of accuracy is obtained for each deferred correction. We develop a very powerful yet rather straightforward theory which allows us to identify the appropriate Runge–Kutta formulae for inclusion in such schemes. In particular, we will focus on the construction of pairs of MIRK formulae of order 4 and 8 which will allow this superconvergence to be realized. We will further show that it is possible to derive formulae of this type for which high order interpolants and accurate error estimates are readily available.

Key words. deferred correction, mono-implicit Runge–Kutta formula, superconvergence, stability, two-point boundary value problems

AMS subject classifications. 65L05, 65L06, 65L20

PII. S1064827599362004

1. Introduction.

In this paper we will be concerned with the numerical solution of the first order system of nonlinear two-point boundary value problems

$$\frac{dy}{dx} = f(x,y), \quad a \le x \le b, \quad g(y(a), y(b)) = 0. \tag{1.1}$$

A widely used technique for the solution of such problems is iterated deferred correction. A single step of a typical deferred correction scheme based on implicit Runge–Kutta formulae can be defined as follows.

Let ϕ_i, ϕ_j be two Runge–Kutta formulae of order i and j, respectively, where $i < j$. Consider the algorithm defined by

$$\phi_i(\eta) = 0, \tag{1.2}$$

$$\phi_i(\bar{\eta}) = -\phi_j(\eta). \tag{1.3}$$

Here η is the solution of the Runge–Kutta formula of order i while $\bar{\eta}$ denotes the (more accurate) solution of the deferred correction scheme (1.3). Then, providing that ϕ_i and ϕ_j have certain special properties, the deferred correction scheme defined by (1.2), (1.3) is of order j. There is a nice heuristic argument for this approach which goes as follows. Let η be the $O(h^i)$ approximation obtained from $\phi_i(\eta) = 0$, and let η^* be the $O(h^j)$ approximation corresponding to $\phi_j(\eta^*) = 0$. We could in principal compute $\bar{\eta}$ from $\phi_i(\bar{\eta}) = \phi_i(\eta^*)$. Of course we cannot do this in practice because the

*Received by the editors October 4, 1999; accepted for publication (in revised form) September 7, 2000; published electronically December 20, 2000.
http://www.siam.org/journals/sisc/22-5/36200.html

†Vakgroep Toegepaste Wiskunde en Informatica, Universiteit Gent, Krijgslaan 281 – S9, B9000 Gent, Belgium (Marnix.VanDaele@rug.ac.be).

‡Department of Mathematics, Imperial College, South Kensington, London SW7, England (j.cash@ic.ac.uk).

right-hand side of this expression is unknown. However, it can be estimated in the following way:

$$\begin{aligned}\phi_i(\eta^*) &= \phi_i(\eta^*) - \phi_j(\eta^*) \\ &\approx \phi_i(\eta) - \phi_j(\eta) \\ &= -\phi_j(\eta),\end{aligned}$$

which gives (1.3). The important question now is what are the conditions under which

$$\|\bar{\eta} - y\| = O(h^j), \tag{1.4}$$

where y is the true solution of (1.1).

Algorithms of this type based on mono-implicit Runge–Kutta (MIRK) methods have been derived in [7], [8], [9], while deferred correction schemes based on Lobatto formulae are developed in [1]. Two codes TWPBVP and ACDC which implement deferred correction schemes are available from NETLIB and from the web page of one of the authors.[1] The first of these codes is based on MIRK formulae, while the second uses Lobatto formulae with continuation.

A somewhat similar defect correction approach has been given by Schild [18]. His investigation started from the observation that Gaussian collocation is an excellent algorithm for the numerical solution of two-point boundary value problems but that the effort required to compute the collocation spline grows rapidly with order. Schild proposed to implement Gaussian collocation via defect correction. Whereas our approach uses only formulae of order 4 and 8, the Schild defect correction scheme uses Gaussian formulae of orders $2, 4, 6$, and 8. In this way Schild's defect correction algorithm typically reduces the error by a factor of $O(h^2)$ per iteration. The question of whether an order 2 starting formula would be more robust than an order 4 formula for difficult problems, where a good initial mesh and solution is hard to find, and whether the complete sequence of orders $2, 4, 6$, and 8 gives better a posteriori error estimates than just orders 4 and 8 is still an open question which is certainly worthy of further investigation.

In what follows we will consider the rather more general deferred correction scheme

$$\phi(\eta) = 0, \tag{1.5}$$

$$\phi(\bar{\eta}) = \psi(\eta)\,. \tag{1.6}$$

A general framework for proving accuracy results for deferred correction schemes of the form (1.5), (1.6) was given in an influential paper by Skeel [19]. In what follows we present his main theorem.

Consider the approximate numerical solution of (1.1) on a mesh

$$\pi : a = x_1 < x_2 < \cdots < x_{N+1} = b.$$

Denote by Δy the restriction of the continuous solution $y(x)$ to the finite grid π. Then we have the following theorem.

THEOREM 1.1. *Let ϕ be a stable numerical method, and assume that the following conditions hold for the deferred correction scheme* (1.5), (1.6):

(i) $\|\eta - \Delta y\| = O(h^p)$,

[1] http://www.ic.ac.uk/~jcash/BVP_software

(ii) $\|\psi(\Delta y) - \phi(\Delta y)\| = O(h^{r+p})$,

(iii) $\psi(\Delta w) = O(h^r)$

for arbitrary functions w having at least r continuous derivatives. Here $\|.\|$ is a suitable finite norm defined in [19], *and h is the maximum grid spacing. If $\phi(\bar{\eta}) = \psi(\eta)$, then*

$$\|\bar{\eta} - \Delta y\| = O(h^{r+p})\,.$$

The feature that is common to all of the deferred correction schemes that have been derived so far is that $r = 2$ and for these schemes the order of accuracy is increased by 2 for each application of the deferred correction. See also [16]. In [1] a sufficient condition to achieve this increase in accuracy was given and this was basically that the Runge–Kutta formulae ϕ_i and ϕ_j should be symmetric and that they should be written in a special way that is appropriate for boundary value problems. This condition is, of course, straightforward to satisfy. The main reason why it is hard to get more than two orders of accuracy improvement per iteration is the difficulty in satisfying condition (iii) for $r > 2$.

The purpose of the present paper is to address the question of whether it is possible to choose ϕ_i and ϕ_j so that we can achieve $r > 2$ while maintaining the efficiency of the deferred correction scheme. We will call deferred correction schemes with $r > 2$ superconvergent, and, in particular, we will investigate the minimum number of stages required to obtain superconvergent MIRK schemes. By focussing on pairs of MIRK formulae of order 4 and 8 we will show that it is possible to achieve this superconvergence and that the resulting deferred correction schemes have potential advantages over some other deferred correction schemes that have been derived.

2. MIRK methods. A special class of MIRK formulae for the numerical solution of stiff initial value problems was first proposed in [10]. These formulae were derived as reflections of standard explicit Runge–Kutta methods, and as a result their order and stability were easy to determine. A rather more general class of MIRK formulae was derived in [2], [11] for initial value problems and in [12] for boundary value problems. The analysis of these schemes was simplified considerably by the work of Enright and Muir [13], who introduced MIRK formulae as a particular class of what they called parameterized IRK methods. These formulae can be written in the general form

$$y_{n+1} = y_n + h \sum_{i=1}^{s} b_i\, f(x_n + c_i\, h,\, Y_i), \tag{2.1}$$

$$Y_i = (1 - v_i)\, y_n + v_i\, y_{n+1} + h \sum_{j=1}^{s} x_{ij}\, f(x_n + c_j\, h,\, Y_j), \qquad i = 1, \ldots, s\,. \tag{2.2}$$

Hence, an s-stage parameterized IRK method is completely determined by the tableau $(b,\, X,\, c,\, v)$, denoted as

$$\begin{array}{c|c|cccc} c_1 & v_1 & x_{11} & x_{12} & \ldots & x_{1s} \\ c_2 & v_2 & x_{21} & x_{22} & \ldots & x_{2s} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ c_s & v_s & x_{s1} & x_{s2} & \ldots & x_{ss} \\ \hline & & b_1 & b_2 & \ldots & b_s \end{array} \tag{2.3}$$

If we compare this representation with that of a general IRK method in terms of its Butcher tableau $(c,\, A,\, b)$ [4], it is easy to verify that the relationship $A = X + v.b^T$

holds. For all methods to be derived in this paper, we will assume that the row-sum condition holds, i.e., $A.e = X.e + v = c$, where e is the s-vector with unit entries. By imposing the restriction that X (or X after a rearrangement of its rows and columns) is a strictly lower triangular matrix, one obtains MIRK methods, and it is exactly this property of X which characterizes MIRK methods. This characterization of MIRK formulae in terms of the matrix X considerably helps in their derivation and shows clearly their special structure. We illustrate this by considering the well-known Clippinger–Dimsdale formula [17, p. 160].

$$\begin{array}{c|ccc} 0 & 0 & 0 & 0 \\ \frac{1}{2} & \frac{5}{24} & \frac{1}{3} & -\frac{1}{24} \\ 1 & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} \\ \hline & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} \end{array} \tag{2.4}$$

At first sight there is nothing remarkable about this formula. However, it is clear that it can be rewritten in the form

$$y_{n+1} - y_n = \frac{h}{6}(k_1 + 4k_2 + k_3), \tag{2.5}$$

where

$$k_1 = f(x_n, y_n), k_2 = f\left(x_{n+\frac{1}{2}}, \frac{y_n + y_{n+1}}{2} - \frac{h}{8}(k_3 - k_1)\right), k_3 = f(x_{n+1}, y_{n+1}). \tag{2.6}$$

Note that (2.5) and (2.6) are a representation of (2.4) in its "eliminated" form (i.e., where the internal stages Y_i ,$i = 1, 2, \ldots s$, have been eliminated). In the algorithmic realization of our deferred correction algorithms, it is the eliminated forms of the MIRK formulae that are used throughout so that the only unknowns in each mesh interval are y_n and y_{n+1}. If we now introduce the notation used in (2.3), we can rewrite (2.4) as

$$\begin{array}{c|cc|ccc} 0 & 0 & & 0 & 0 & 0 \\ 1 & 1 & & 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & & \frac{1}{8} & -\frac{1}{8} & 0 \\ \hline & & & \frac{1}{6} & \frac{1}{6} & \frac{2}{3} \end{array} \tag{2.7}$$

This can now be identified from (2.7) as a MIRK formula since the matrix X is strictly lower triangular. In [3] several results concerning the order and stability of MIRK methods have been established. Among the more important of these results are the following theorems.

THEOREM 2.1. *An s-stage pth order MIRK method satisfies $p \leq s + 1$.*

THEOREM 2.2. *The stage order of a MIRK method is 3 at most.*

THEOREM 2.3. *For the test equation $y' = \lambda y$, $y(x_0) = y_0$ one can write $y_n = R(\lambda h)^n y_0$, where $R(z) = P(e - v, z)/P(-v, z)$ with $P(w, z) = 1 + z\, b^T.(I - z\, X)^{-1}.w = 1 + \sum_{i=1}^{s} z^i\, b^T.X^{i-1}.w$.*

MIRK methods have not been widely used so far for the numerical solution of initial value problems. This is due primarily to the fact that if the nonlinear algebraic

equations arising from the application of a MIRK formula are solved using some form of Newton iteration, then the coefficient matrix of the Newton scheme is a polynomial in the Jacobian matrix. Furthermore, stability problems may also arise, as explained in [20]. However, the situation for boundary value problems is quite different. If the MIRK formulae for the solution of (1.1) are written in a deferred correction framework, then the deferred corrections $\psi(\eta)$ are explicit and thus cheap to compute. Furthermore, the linear algebra cost of computing $\bar{\eta}$ from (1.3) is relatively small, and the required stability properties of the underlying formulae are maintained. This in turn means that algorithms for local error estimation and mesh changing are relatively cheap, and this makes deferred correction methods based on MIRK formulae very competitive with other algorithms for the solution of (1.1).

3. Deferred correction with MIRK methods. As we have mentioned in the previous section, all deferred correction methods which have so far been used for the solution of two-point boundary value problems increase the order by 2 for each deferred correction. The reason why higher increases in the order of accuracy have not been obtained is because of the difficulty of satisfying condition (iii) of Theorem 1.1. In this section we examine the possibility of increasing the order of accuracy by 4 for each iteration. This problem has also been looked at in the context of initial value problems in [20], [21], and many of the ideas contained in these two papers will be important in what follows. Before we start our analysis we wish to remark that our derivation holds not only for MIRK methods but also for any paramaterized Runge–Kutta method.

Let ϕ be a MIRK method of order p, and define $\psi := \phi - \phi^*$, where ϕ^* corresponds to a MIRK method of order $p^* > p$. (We will systematically denote the quantities that relate to ϕ^* with a *-superscript : s^*, a^*_{ij}, b^*_i, c^*_i, ...) The formula $\phi(\Delta\eta)$ can then be written as

$$\phi(\eta)_{n+1} := \frac{1}{h}\,(\eta_{n+1} - \eta_n) - \sum_{i=1}^{s} b_i\, f(x_n + c_i\, h,\, Y_i)\,.$$

To analyze the conditions of Theorem 1.1, we write down the B-series expansions for ϕ and ϕ^*. We therefore rewrite $\phi(\eta)_{n+1}$ as

$$\begin{aligned} g_i &= (1 - v_i)\,\eta_n + v_i\,\eta_{n+1} + \sum_{j=1}^{s} x_{ij}\,k_j, \\ k_i &= h\,f(x_n + c_i\,h,\, g_i), \\ \phi(\eta)_{n+1} &= \frac{\eta_{n+1} - \eta_n - \sum_{i=1}^{s} b_i\,k_i}{h}. \end{aligned}$$

Now if $g_i = B(\boldsymbol{g_i},\, \eta_n)$, $k_i = B(\boldsymbol{k_i},\, \eta_n)$, $\eta_{n+1} = B(\boldsymbol{\eta_{n+1}}, \eta_n)$, and $h\,\phi(\eta)_{n+1} = B(\boldsymbol{\phi_{n+1}},\, \eta_n)$ are all B-series, then Corollary 12.7 in [14] allows us to transcribe these formulas for each labelled tree $t = [t_1 \cdots t_m]$ as

$$\begin{cases} \boldsymbol{g_i}(t) = v_i\,\boldsymbol{\eta_{n+1}}(t) + \displaystyle\sum_{j=1}^{s} x_{ij}\,\boldsymbol{k_j}(t), \\ \boldsymbol{k_i}(t) = \rho(t)\,\boldsymbol{g_i}(t_1)\,\boldsymbol{g_i}(t_2)\cdots\boldsymbol{g_i}(t_m), \\ \boldsymbol{\phi_{n+1}}(t) = \boldsymbol{\eta_{n+1}}(t) - \displaystyle\sum_{j=1}^{s} b_j\,\boldsymbol{k_j}(t) \end{cases}$$

with $\boldsymbol{g_i}(\emptyset) = 1$, $\boldsymbol{k_i}(\tau) = 1$, and $\boldsymbol{\phi_{n+1}}(\emptyset) = 1$.

For condition (i) of Theorem 1.1, it follows from $\phi(\eta)_{n+1} = 0$ that $\boldsymbol{\phi_{n+1}}(t) = 0$ for all t, such that

$$\begin{cases} \boldsymbol{g_i}(t) = \sum_{j=1}^{s} (v_i\, b_j + x_{ij})\, \boldsymbol{k_j}(t), \\ \boldsymbol{k_i}(t) = \rho(t)\, \boldsymbol{g_i}(t_1)\, \boldsymbol{g_i}(t_2) \cdots \boldsymbol{g_i}(t_m), \\ \boldsymbol{\eta_{n+1}}(t) = \sum_{j=1}^{s} b_j\, \boldsymbol{k_j}(t). \end{cases}$$

Let $y_n(h) = B(\boldsymbol{y_n}, \eta_n)$ be the true solution of $y' = f(x, y)$, $y(x_n) = \eta_n$; then $\boldsymbol{y_n}(t) = 1$ for all t, and the well-known order conditions up to order p are obtained by comparing the coefficients of the B-series $B(\boldsymbol{y_n}, \eta_n)$ and $B(\boldsymbol{\eta_{n+1}}, \eta_n)$.

Condition (ii) of Theorem 1.1 analogously describes that ϕ^* has order p^*.

For the third condition of Theorem 1.1, we replace η by an arbitrary function w and the corresponding B-series $B(\boldsymbol{\eta_{n+1}}, \eta_n)$ by $B(\boldsymbol{w_{n+1}}, \eta_n)$. Condition (iii) then leads to

$$\sum_{j=1}^{s} b_j\, \boldsymbol{k_j}(t) - \sum_{j=1}^{s^*} b_j^*\, \boldsymbol{k_j^*}(t) = 0 \tag{3.1}$$

for all t with $\rho(t) \le r$, whereby

$$\begin{cases} \boldsymbol{g_i}(t) = v_i\, \boldsymbol{w_{n+1}}(t) + \sum_{j=1}^{s} x_{ij}\, \boldsymbol{k_j}(t), \\ \boldsymbol{k_i}(t) = \rho(t)\, \boldsymbol{g_i}(t_1)\, \boldsymbol{g_i}(t_2) \cdots \boldsymbol{g_i}(t_m), \end{cases} \qquad i = 1, \ldots, s,$$

$$\begin{cases} \boldsymbol{g_i^*}(t) = v_i^*\, \boldsymbol{w_{n+1}}(t) + \sum_{j=1}^{s^*} x_{ij}^*\, \boldsymbol{k_j^*}(t), \\ \boldsymbol{k_i^*}(t) = \rho(t)\, \boldsymbol{g_i^*}(t_1)\, \boldsymbol{g_i^*}(t_2) \cdots \boldsymbol{g_i^*}(t_m), \end{cases} \qquad i = 1, \ldots, s^*.$$

We thus find for all trees t with $\rho(t) \le 4$

$$\sum_{j=1}^{s} b_j\, \boldsymbol{k_j}(\tau) = 1,$$

$$\sum_{j=1}^{s} b_j\, \boldsymbol{k_j}([\tau]) = 2\, b^T.v\, \boldsymbol{w_{n+1}}(\tau) + 2\, b^T.X.e,$$

$$\sum_{j=1}^{s} b_j\, \boldsymbol{k_j}([\tau^2]) = 3\, b^T.v^2\, \boldsymbol{w_{n+1}}(\tau)^2 + 6\, b^T.(v\, X.e)\, \boldsymbol{w_{n+1}}(\tau) + 3\, b^T.(X.e)^2,$$

$$\sum_{j=1}^{s} b_j\, \boldsymbol{k_j}([_2\tau]_2) = 3\, b^T.v\, \boldsymbol{w_{n+1}}([\tau]) + 6\, b^T.X.v\, \boldsymbol{w_{n+1}}(\tau) + 6\, b^T.X^2.e,$$

$$\sum_{j=1}^{s} b_j\, \boldsymbol{k_j}([\tau^3]) = 4\, b^T.v^3\, \boldsymbol{w_{n+1}}(\tau)^3 + 12\, b^T.(v^2\, X.e)\, \boldsymbol{w_{n+1}}(\tau)^2$$

$$
\begin{aligned}
&\qquad +12\,b^T.(v\,(X.e)^2)\,\boldsymbol{w_{n+1}}(\tau) + 4\,b^T.(X.e)^3,\\
\sum_{j=1}^{s} b_j\,\boldsymbol{k_j}([\tau[\tau]_2) &= 4\,b^T.v^2\,\boldsymbol{w_{n+1}}(\tau)\,\boldsymbol{w_{n+1}}([\tau]) + 4\,b^T.(v\,X.e)\,\boldsymbol{w_{n+1}}([\tau])\\
&\qquad +8\,[b^T.(X.e\,X.v) + b^T.(v\,X^2.e)]\,\boldsymbol{w_{n+1}}(\tau)\\
&\qquad +8\,b^T.(v\,X.v)\,\boldsymbol{w_{n+1}}(\tau)^2 + 8\,b^T.(X.e\,X^2.e),\\
\sum_{j=1}^{s} b_j\,\boldsymbol{k_j}([{}_2\tau^2]_2) &= 4\,b^T.v\,\boldsymbol{w_{n+1}}([\tau^2]) + 12\,b^T.X.v^2\,\boldsymbol{w_{n+1}}(\tau)^2\\
&\qquad +24\,b^T.X.(v\,X.e)\,\boldsymbol{w_{n+1}}(\tau) + 12\,b^T.X.(X.e)^2,\\
\sum_{j=1}^{s} b_j\,\boldsymbol{k_j}([{}_3\tau]_3) &= 4\,b^T.v\,\boldsymbol{w_{n+1}}([{}_2\tau]_2) + 12\,b^T.X.v\,\boldsymbol{w_{n+1}}([\tau]) + 24\,b^T.X^2.v\,\boldsymbol{w_{n+1}}(\tau)\\
&\qquad +4\,b^T.X^3.e.
\end{aligned}
$$

If condition (3.1) has to be fulfilled for any B-series $\boldsymbol{w_{n+1}}$ and for any method ϕ of order p and any method ϕ^* of order p^*, where $p^* \geq p \geq r$, we obtain an extra set of conditions. If we assume that both ϕ and ϕ^* have stage order at least 1 and we take into account that the order conditions up to order r are fulfilled, then we must have that $r \leq \max\{i|E_i = 0\}$, where

$$
\begin{aligned}
&E_1 := 0,\\
&E_2 := |b^T.v - b^{*T}.v^*|,\\
&E_3 := |b^T.v^2 - b^{*T}.v^{*2}| + |b^T.(c\,v) - b^{*T}.(c^*\,v^*)| + |b^T.X.v - b^{*T}.X^*.v^*|,\\
&E_4 := |b^T.v^3 - b^{*T}.v^{*3}| + |b^T.(c\,v^2) - b^{*T}.(c^*\,v^{*2})| + |b^T.(c^2\,v) - b^{*T}.(c^{*2}\,v^*)|\\
&\qquad +|b^T.X.v^2 - b^{*T}.X^*.v^{*2}| + |b^T.X^2.v - b^{*T}.X^{*2}.v^*|\\
&\qquad +|b^T.(v\,X.v) - b^{*T}.(v^*\,X^*.v^*)| + |b^T.(c\,X.v) - b^{*T}.(c^*\,X^*.v^*)|\\
&\qquad +|b^T.(X.c\,v) - b^{*T}.(X^*.c^*\,v^*)| + |b^T.X.(c\,v) - b^{*T}.X^*.(c^*\,v^*)|,\\
&E_5 := \cdots.
\end{aligned}
\tag{3.2}
$$

It is of interest to note that for a general MIRK method the stage order equations read

$$
kX.c^{k-1} = c^k - v, \qquad k = 1, 2, \ldots, q, \tag{3.3}
$$

whereas (3.2) involves $X.v$ rather than $X.c$. To really use the stage order conditions in this analysis we would need to choose $c = v$ and $c^* = v^*$. We will see later that this is indeed the choice we make.

The expressions E_i can be generalized: each E_i ($i \geq 2$) contains all strings of i elements of which b^T and v are the first and last ones. In between is a permutation with replacement of 3 in groups of $i - 2$ from the set $\{c, v, X\}$.

We thus find that, while the value of r in condition (iii) is 1 in general, it can be raised to 2 or even higher. In [5], [6], where symmetric methods are used, the value $r = 2$ is obtained since for all symmetric methods $b^T.v = 1/2$. Symmetry is, however, not necessary for $r = 2$. The above result also shows that convergence can be accelerated if some extra conditions are satisfied. We call such methods superconvergent.

Of course, the question of how to satisfy the conditions (3.2) still remains. For the case $q = 4$, for example, there are 13 extra conditions. It would seem that these

conditions will significantly increase the number of stages required by the method. Some of these conditions can be eliminated if one uses methods with high stage order, but most of them remain to be fulfilled. There is, however, one important case for which (3.2) imposes no extra conditions: indeed, if $c = v$ and $c^* = v^*$, then all expressions which appear in (3.2) are fixed numbers since they are order conditions (re-expressed in terms of X). If $c = v$ and $c^* = v^*$, the value $r = p$ is thus obtained automatically. In the following section, we will investigate the existence of deferred correction schemes based on MIRK methods for which $c = v$. We further restrict ourselves to symmetric methods, a natural choice in the context of BVPs.

The conditions of symmetry for a Runge–Kutta method are well known and can be expressed as follows: if x_n and x_{n+1} and y_n and y_{n+1} are swapped, and if h is replaced by $-h$, then the original method is obtained. The MIRK formula tableau is a particularly convenient representation for the investigation of symmetry. The condition for symmetry comes down to the requirement that there must exist a permutation σ of the stages such that for each stage Y_i there exists a stage $Y_{\sigma(i)}$ for which $c_{\sigma(i)} = 1 - c_i$, $v_{\sigma(i)} = 1 - v_i$, and $x_{\sigma(i)\sigma(j)} = -x_{ij}$.

4. Superconvergent deferred correction schemes based on symmetric MIRK methods with $c = v$. It was shown in the previous section that, in the quest for superconvergent deferred correction schemes, there are some important advantages to be gained by making the particular choice $v = c$. In what follows we will refer to formulae with this particular choice as MIRKA formulae. As was shown in the previous section, the superconvergence conditions for MIRKA formulae reduce to standard order conditions. The main result that we will give in this section is a lower bound on the number of stages of symmetric MIRKA methods of a given (even) order.

A MIRK formula is conveniently characterized by the properties of its X matrix. Hence in what follows we will develop a framework to derive high order MIRKA formulae in terms of the elements x_{ij}.

Stage order. If $c = v$, then stage order q is obtained if and only if

$$C(q) \ : X.c^{k-1} = \frac{1}{k}\,(c^k - c)\,, \qquad k = 1, \ldots, q\,.$$

In particular, the row-sum condition $A.e = c$ means $X.e = (0, 0, \ldots, 0)^T$.

We now investigate the relationship between the order p and the minimum number of stages s of a MIRKA formula. The situation for $s = 1$ is very straightforward.

The implicit midpoint rule is a symmetric MIRKA formula with only one stage and is of order 2. Its tableau is

(4.1)

$\frac{1}{2}$	$\frac{1}{2}$	0
		1

For higher values of p (p must be even because we are only considering symmetric formulae) we have the following theorem.

THEOREM 4.1. *A symmetric MIRKA formula of order $p \geq 4$ requires at least $s = 2\,p - 5$ stages.*

This theorem relies on the following lemmas, which are straightforward to prove.

LEMMA 4.2. *The stability function of a MIRKA formula is given by*

$$R(h) = \frac{1 + h - h\,S(h)}{1 - h\,S(h)} = 1 + \frac{h}{1 - h\,S(h)},$$

where $S(h) = b^T.(I - h\,X)^{-1}.c = \sum_{i=0}^{s}(b^T.X^i.c)\,h^i$,

Since $R(h)$ is an approximation to $\exp(h)$, it follows that $S(h) \approx \frac{1}{2} + \frac{1}{h} - \frac{1}{2}\coth\frac{h}{2}$.

LEMMA 4.3. *If* $S(h) = \frac{1}{2} + \frac{1}{h} - \frac{1}{2}\coth\frac{h}{2} + \mathcal{O}(h^{2\,l+1})$, *then* $R(h) = \exp(h) + \mathcal{O}(h^{2\,l+3})$.

LEMMA 4.4. *For a symmetric nonconfluent s-stage MIRK formula, one has* $X^{s'} = O$, *where* $s' = \lfloor\frac{s+1}{2}\rfloor$ *and where* O *is the matrix of appropriate size with zero entries.*

With the help of these lemmas, we can now prove Theorem 4.1.

Proof. Suppose we are looking for a symmetric formula of order $p \geq 4$. Then, according to Lemma 4.3, we must have

$$S(h) - \left(\frac{1}{2} + \frac{1}{h} - \frac{1}{2}\coth\frac{h}{2}\right) = \mathcal{O}(h^{p-1})\,,$$

which means that, since $\frac{1}{h} - \frac{1}{2}\coth\frac{h}{2}$ is odd, $S(h)$ must be a polynomial of degree $p-3$ at least and according to Lemma 4.2, X^{p-3} does not vanish. From Lemma 4.4 it follows that

$$p - 3 < s' = (s+1)/2.$$

This now gives the result that

$$\begin{cases} s \geq 2\,p - 4 & \text{if } s \text{ is even,} \\ s \geq 2\,p - 5 & \text{if } s \text{ is odd.} \end{cases} \qquad \square$$

For $p = 4$ it follows that $s = 3$ stages is sufficient, as is demonstrated by the Lobatto formula (2.7). In order to find higher order methods, it is standard practice to reduce the number of order equations by making various simplifying assumptions. One possibility is to choose the stage order to be as high as possible. This is the approach that we will adopt in this paper, and in what follows we will consider only MIRKA methods with stage-order 3. We now give a lemma which will help in the construction of symmetric MIRKA methods of this type.

LEMMA 4.5. (i) *If a symmetric MIRKA method is nonconfluent, then stage order* 3 *implies that* $c_3 = 1/2$ *(which means that s will be odd);* (ii) *the fourth and fifth stages can only have order of consistency at most* 3, *while the sixth and seventh stages have order of consistency (stage order) at most* 5, *and so on.*

For $p = 6$ and $p = 8$ the minimum number of stages is given by Theorem 4.1 as 7 and 11, respectively. In the next section we will investigate these two cases. First, we will show that $p = 8$ can only be achieved with at least 13 stages. We will construct such a family of methods with 13 stages, and we will show that it is possible (i) to add an 8th order interpolant and (ii) to add an embedded method of order 6 with a 6th order interpolant. Second, we will use some ideas from the case $p = 8$ to show that $p = 6$ can be obtained with seven stages.

5. The construction of symmetric MIRKA methods. In this section we consider the derivation of an order 8 method using the classical approach of Butcher.

5.1. Order 8 methods. It is well known that, if the stage order conditions are satisfied ad infinitum, the only order conditions that remain to be satisfied are the quadrature conditions. These take the form

$$b^T.c^k = \frac{1}{k+1}, \qquad k = 0,\,1,\,\ldots\,.$$

However, it is also well known that the stage order of MIRK methods is bounded by 3. In order to minimize the number of remaining order conditions, we will try to construct methods for which the stage order reaches the maximum value. If this is done, it follows that, apart from the quadrature conditions, there are 26 order conditions left. We will see later that the minimum number of stages for a symmetric MIRKA method of order 8 is 13, and this leads us to consider the scheme A given in the appendix.

We will choose $b_3 = b_4 = 0$ to reduce the effect of low stage order for stages 3, 4, and 5. It is clear that in this case x_{ij} $(1 \le j \le 3)$ can be expressed in terms of c_i and x_{ij} $(4 \le j < i)$.

We note that the matrix X contains 2 by 2 matrices of the form

$$\begin{pmatrix} x_{ij} & x_{i,j+1} \\ -x_{i,j+1} & -x_{ij} \end{pmatrix} = \frac{y_{ij}}{2} \begin{pmatrix} 1 & -1 \\ 1 & -1 \end{pmatrix} + \frac{z_{ij}}{2} \begin{pmatrix} 1 & 1 \\ -1 & -1 \end{pmatrix} = y_{ij}\, Y + z_{ij}\, Z,$$

where $y_{ij} = x_{ij} - x_{i,j+1}$ and $z_{ij} = x_{ij} + x_{i,j+1}$. The matrices Y and Z appearing on the right-hand side of this identity are very easy to work with when considering powers of X. This is first because they are nilpotent, and second, we have the identities $ZYZ = Z$ and $YZY = Y$. This motivates us to work with the y and z variables instead of the x variables. To complete this transformation we also need to define $z_{i3} = x_{i3}$.

We now constrain all stages to have order of consistency 5, except stages 3, 4, and 5, which have order of consistency 3. (This is the maximum that we can achieve for these stages).

These extra conditions imply that all coefficients in the upper seven stages are constants or are expressed in terms of c_4 and c_6. In the last six stages, some degrees of freedom are left and we let the undetermined variables be y_{86}, z_{86}, $y_{10,6}$, $y_{10,8}$, $z_{10,6}$, $z_{10,8}$, $y_{12,6}$, $y_{12,8}$, $y_{12,10}$, $z_{12,6}$, $z_{12,8}$, and $z_{12,10}$.

If the conditions that are to be satisfied by the different stages are written in terms of the y_{ij} and z_{ij} variables, it can be shown that for $i = 2, 3, \ldots, 6$,

$$y_{2i,j} = \mathcal{Y}^{[0]}_{2i,j} + \sum_{k=3}^{i-1} \mathcal{Y}^{[k]}_{j}\, y_{2i,2k}\,, \qquad j = 1,\, 4\,, \tag{5.1}$$

$$z_{2i,j} = \mathcal{Z}^{[0]}_{2i,j} + \sum_{k=3}^{i-1} \mathcal{Z}^{[k]}_{j}\, z_{2i,2k}, \qquad j = 1,\, 3,\, 4\,, \tag{5.2}$$

where the $\mathcal{Y}$ and $\mathcal{Z}$ coefficients depend only on the elements of the vector c.

Due to the above choices, many of the remaining conditions are automatically satisfied. Indeed, the choices made allow us to state that, for example, all order conditions associated with trees of the form $[\tau^i[\tau^3]_2$ and $[\tau^i[\tau^4]_2$ (i.e., in the expression for the order equation the matrix A appears only once) are automatically satisfied.

We now need to satisfy the 14 remaining conditions. We first re-express these conditions in terms of X instead of A since this will make the equations easier to work with. Thus, for example, we have the conditions

$$b^T.X.c^i = -\frac{i}{2\,(i+1)\,(i+2)}\,, \tag{5.3}$$

which still remain to be satisfied for $i = 5$ and $i = 6$. However, these conditions are not independent of each other. Defining

$$\begin{cases} Y_1 = b_6\, y_{61} + b_8\, y_{81} + b_{10}\, y_{10,1} + b_{12}\, y_{12,1}, \\ Y_4 = b_6\, y_{64} + b_8\, y_{84} + b_{10}\, y_{10,4} + b_{12}\, y_{12,4}, \\ Y_6 = b_8\, y_{86} + b_{10}\, y_{10,6} + b_{12}\, y_{12,6}, \\ Y_8 = b_{10}\, y_{10,8} + b_{12}\, y_{12,8}, \\ Y_{10} = b_{12}\, y_{12,10}, \end{cases}$$

the conditions (5.3) can be rewritten as $E_i = 0$, where

$$E_i := Y_1 + \sum_{j=2}^{5} (c_{2\,j}^i - (1 - c_{2\,j})^i)\, Y_{2\,j} + \frac{i}{2\,(i+1)\,(i+2)}\,.$$

It can then immediately be shown that

$$\sum_{i=0}^{n} (-1)^i \binom{n}{i} E_{2\,n-i} = 0\,.$$

Using the identities

$$(1-1)^n = \sum_{i=0}^{n} (-1)^i \binom{n}{i} = 0$$

and

$$x^n\,(x-1)^n = \sum_{i=0}^{n} (-1)^i \binom{n}{i} x^{2\,n-i} = \sum_{i=0}^{n} (-1)^i \binom{n}{i} (1-x)^{2\,n-i}\,,$$

it follows that

$$\sum_{i=0}^{n} (-1)^i \binom{n}{i} \left(x^{2\,n-i} - (1-x)^{2\,n-i}\right) = 0\,.$$

From this last term we obtain the expression

$$x^{n+2}\,(1-x)^n - x^n\,(1-x)^{n+2} = x^n\,(1-x)^n\,(2\,x-1) = \sum_{i=0}^{n} (-1)^i \binom{n}{i} (2\,x^{2\,n+1-i} - x^{2\,n-i}),$$

and after integration between 0 and 1 we have

$$\begin{aligned} 0 &= \sum_{i=0}^{n} (-1)^i \binom{n}{i} \left(\frac{2}{2\,n+2-i} - \frac{1}{2\,n+1-i} \right) \\ &= \sum_{i=0}^{n} (-1)^i \binom{n}{i} \frac{2\,n-i}{(2\,n+1-i)\,(2\,n+2-i)}\,. \end{aligned}$$

This shows that $E_6 = 0$ is automatically satisfied if $E_5 = 0$ is satisfied. We can thus impose the equivalent condition $E_5 - E_6 = 0$:

$$\sum_{i=3}^{5} Y_{2\,i}\, c_{2\,i}\, (1 - c_{2\,i})\, (c_{2\,i}^4 - (1 - c_{2\,i})^4) = -\frac{1}{168}. \tag{5.4}$$

To satisfy the remaining 13 conditions, we use the fact that

$$w_j = X\,c^j - \frac{1}{j+1}(c^{j+1} - c)\,, \qquad j = 3,\, 4, \tag{5.5}$$

has three nonzero components, and we replace every order condition of the form $b^T.M.X.c^j = \cdots$ (where M is any product of $C = \mathrm{diag}(c_1, \ldots, c_s)$ and X matrices) by $b^T.M.w_j = 0$.

This leads to the following five conditions:

$$0 = Y_4, \tag{5.6}$$

$$0 = \sum_{i=2}^{5} Y_{2\,i}\, z_{2\,i,3}, \tag{5.7}$$

$$0 = \sum_{i=3}^{5} Y_{2\,i}\, z_{2\,i,4}, \tag{5.8}$$

$$0 = \sum_{i=2}^{6} b_{2\,i}\ (2\,c_{2\,i} - 1)\ z_{2\,i,3}, \tag{5.9}$$

$$0 = \sum_{i=3}^{6} b_{2\,i}\ (2\,c_{2\,i} - 1)\ z_{2\,i,4}. \tag{5.10}$$

Due to the structure of (5.2), (5.9)–(5.10) can be rewritten as

$$\alpha(c_4,\, c_6)\, Z_6 + \alpha(c_4,\, c_8)\, Z_8 + \alpha(c_4,\, c_{10})\, Z_{10} = \frac{2 - 7\,c_4 + 7\,c_4^2}{2520}, \tag{5.11}$$

$$\beta(c_6)\, Z_6 + \beta(c_8)\, Z_8 + \beta(c_{10})\, Z_{10} = \frac{1}{2520} \tag{5.12}$$

in terms of the variables Z_6, Z_8, and Z_{10}, where

$$\begin{cases} Z_6 = b_8\,(1 - 2\,c_8)\, z_{86} + b_{10}\,(1 - 2\,c_{10})\, z_{10,6} + b_{12}\,(1 - 2\,c_{12})\, z_{12,6}, \\ Z_8 = b_{10}\ (1 - 2\,c_{10})\, z_{10,8} + b_{12}\,(1 - 2\,c_{12})\, z_{12,8}, \\ Z_{10} = b_{12}\,(1 - 2\,c_{12})\, z_{12,10}\,. \end{cases}$$

The equations (5.4) and (5.6) form a system of two equations in the three unknowns Y_6, Y_8, and Y_{10}. We can solve this system for Y_6 and Y_8 in terms of Y_{10} to give

$$Y_6 = -\frac{1}{\alpha(c_8,\, c_6)\,(1 - 2\,c_6)}\ \left[\frac{1 - 7\,c_8 + 7\,c_8^2}{840} + \alpha(c_8,\, c_{10})\,(1 - 2\,c_{10})\, Y_{10}\right], \tag{5.13}$$

$$Y_8 = -\frac{1}{\alpha(c_6,\, c_8)\,(1 - 2\,c_8)}\ \left[\frac{1 - 7\,c_6 + 7\,c_6^2}{840} + \alpha(c_6,\, c_{10})\,(1 - 2\,c_{10})\, Y_{10}\right], \tag{5.14}$$

where $\alpha(c,\, d) = d\,(1 - d)\,(c - d)\,(1 - c - d)$.

Substituting these results into (5.7) and (5.8), the following two relations are obtained:

$$\frac{7\,c_4^2 - 7\,c_4 + 2}{5040} = \alpha(c_4,\, c_6)\, Y_{10}\, z_{10,6} + \alpha(c_4,\, c_8)\, Y_{10}\, z_{10,8} - \frac{\alpha(c_4,\, c_6)}{\alpha(c_6,\, c_8)\,(1 - 2\,c_8)}\, Q\, z_{86}, \tag{5.15}$$

$$\frac{1}{5040} = \beta(c_6)\, Y_{10}\, z_{10,6} + \beta(c_8)\, Y_{10}\, z_{10,8} - \frac{\beta(c_6)}{\alpha(c_6,\, c_8)\,(1 - 2\,c_8)}\, Q\, z_{86}, \tag{5.16}$$

where $Q = \frac{1 - 7\,c_6 + 7\,c_6^2}{840} + \alpha(c_6,\, c_{10})\,(1 - 2\,c_{10})\, Y_{10}$ and $\beta(c) = c\,(1 - c)\,(1 - 2\,c)^2$.

From the above, it follows that solutions can exist only for $Y_{10} \neq 0$, i.e., $b_{12} \neq 0$. This implies that if stage order three is assumed, at least 13 stages are needed!

Although this result is somewhat disappointing, there are actual advantages in considering 13 stages. First, now that we know we cannot choose $b_{12} = 0$, we are able to choose $b_6 = 0$. It now follows that, for a method of order 8, the four remaining variables of the b-vector can be uniquely determined in terms of the c-values. Further, we can also raise the order of consistency of the lower six stages from 5 to 7. This in turn allows us to construct an eighth order interpolant $u_8(x)$. For reasons of symmetry, we could impose, for example, the following conditions on a polynomial of degree nine:

$$\begin{aligned}
&u_8(x_n) = y_n, \qquad u_8(x_n + h) = y_{n+1},\\
&u_8'(x_n) = k_1, \qquad u_8'(x_n + h) = k_2,\\
&u_8'(x_n + c_{2i}\,h) = f(x_n + c_{2i}\,h,\, u_8(x_n + c_{2i}\,h)) = k_{2i}, \qquad i = 4,\, 5,\, 6,\\
&u_8'(x_n + (1 - c_{2i})\,h) = f(x_n + (1 - c_{2i})\,h,\, u_8(x_n + (1 - c_{2i})\,h)) = k_{2i+1}\,, i = 4,\, 5,\, 6.
\end{aligned}$$

Raising the order of consistency of the last six stages from 5 to 6 means that (5.1) also holds for $j = 6$. (We are now left with $y_{10,8}$, $y_{12,8}$, and $y_{12,10} \neq 0$ as free y-coefficients.) It also follows that (5.4) is automatically satisfied. From (5.6) one obtains relation (5.14) between Y_8 and Y_{10}. If no further conditions are imposed, then we have the following procedure for obtaining an 8th order method.

- The system (5.15)–(5.16) can be solved for $Y_{10}\, z_{10,6}$ and $Y_{10}\, z_{10,8}$ in terms of Y_{10} and z_{86}.
- The system (5.11)–(5.12) can then be solved for Z_6 and Z_8 in terms of Z_{10}.

If we now analyze the construction process, we can conclude that we have constructed a family of methods which, apart from c_4, c_6, c_8, c_{10}, and c_{12}, still has four free parameters: $y_{12,10}$ ($\neq 0$), $z_{12,10}$, z_{86}, and either $y_{10,8}$ or $y_{12,8}$.

However, the interpolant requires that the order of consistency of the last six stages is raised from 6 to 7. This means that (5.2) also holds for $j = 6$ ($z_{10,8}$, $z_{12,8}$, and $z_{12,10}$ are the remaining free z-coefficients). Due to the structure of (5.2), the equations can be solved as follows.

- The system (5.15)–(5.16) is linear in Y_{10} and $Y_{10}\, z_{10,8}$. Solving this system, we can thus fix Y_{10} and $z_{10,8}$ in terms of the c-values.
- The system (5.11)–(5.12) can be solved for Z_8 and Z_{10} in terms of the c-values.
- Y_8 is determined by Y_{10} in (5.14). As a consequence, $y_{12,8}$ is a function of $y_{10,8}$ or vice versa.

In this way we have constructed a family of methods which, apart from c_4, c_6, c_8, c_{10}, and c_{12}, still has one free parameter: either $y_{10,8}$ or $y_{12,8}$. It follows from earlier

analysis that all of these methods will have the same stability function, which is

$$R_2(h) = \frac{N_2(h)}{D_2(h)} = \frac{1 + \frac{h}{2} + \frac{h^2}{12} - \frac{h^4}{720} + \frac{h^6}{30240}}{1 - \frac{h}{2} + \frac{h^2}{12} - \frac{h^4}{720} + \frac{h^6}{30240}}\,.$$

For IVPs the combination of (2.7), for which the stability function is given by

$$R_1(h) = \frac{N_1(h)}{D_1(h)} = \frac{1 + \frac{h}{2} + \frac{h^2}{12}}{1 - \frac{h}{2} + \frac{h^2}{12}},$$

and any of these 8th order methods leads to an unstable scheme since the stability function [20] is given by

$$\begin{aligned} R(h) &= \frac{[D_1(h) - D_2(h)]\, R_1(h) + N_2(h)}{D_1(h)} \\ &= \frac{1 - \frac{h^2}{12} + \frac{h^4}{144} + \frac{h^5}{720} - \frac{h^7}{30240}}{\left(1 - \frac{h}{2} + \frac{h^2}{12}\right)^2}. \end{aligned}$$

For BVPs, however, the stability function [6] is given by

$$R(h,\, n) = R_1(h)^n \,\left[R_1(h) + (n+1)\,(R_2(h) - R_1(h))\right],$$

where

$$R_2(h) - R_1(h) = \frac{\frac{h^5}{720} - \frac{h^7}{30240}}{\left(1 - \frac{h}{2} + \frac{h^2}{12}\right)\,\left(1 - \frac{h}{2} + \frac{h^2}{12} - \frac{h^4}{720} + \frac{h^6}{30240}\right)}.$$

Note now that

$$\lim_{h\to\infty} R(h,\, n) = 1\,,$$

i.e., the stability function is bounded for all values of h. We give the coefficients of our 8th order formula in section A of the appendix.

We now construct an embedded MIRK method $(\hat{b},\, X,\, c,\, c)$ of order 6. To do so, we note that apart from the quadrature conditions, three conditions remain to be satisfied: namely,

$$\hat{b}^T.X.X.c^3 = \frac{1}{240}, \qquad \hat{b}^T.(c\,X).c^3 = -\frac{1}{24}, \qquad \hat{b}^T.X.c^4 = -\frac{1}{15}\,.$$

It is clear that these conditions are satisfied if $\hat{b}_3 = 0 = \hat{b}_4$. In order to have a method of order 6, we need at least three nonzero $\hat{b}$-coefficients, and therefore it follows that there are 6th order embedded methods with nine stages. Due to the fact that all of the stages with nonzero $\hat{b}$-weight have order of consistency 5 at least, it follows that these embedded methods have an interpolant $u_6(x)$ of order 6. This can be realized by imposing the conditions

$$\begin{aligned} &u_6(x_n) = y_n, \qquad u_6(x_n + h) = y_{n+1}, \\ &u_6'(x_n) = k_1, \qquad u_6'(x_n + h) = k_2, \\ &u_6'(x_n + c_{2i}\,h) = f(x_n + c_{2i}\,h,\, u_6(x_n + c_{2i}\,h)) = k_{2\,i}, \qquad i = 3,\, 4, \\ &u_6'(x_n + (1 - c_{2i})\,h) = f(x_n + (1 - c_{2i})\,h,\, u_6(x_n + (1 - c_{2i})\,h)) = k_{2\,i+1}, \quad i = 3,\, 4. \end{aligned}$$

The $\hat{b}$ coefficients for the embedded formula are given in section B of the appendix.

5.2. Order 6 methods. We use the same approach as was followed in the previous section to construct methods of order 6 with seven stages. We propose a scheme of the form

$$\begin{array}{c|c|ccccccc}
0 & 0 & & & & & & & \\
1 & 1 & 0 & & & & & & \\
\frac{1}{2} & \frac{1}{2} & \frac{1}{8} & -\frac{1}{8} & & & & & \\
c_4 & c_4 & x_{41} & x_{42} & x_{43} & & & & \\
1-c_4 & 1-c_4 & -x_{42} & -x_{41} & -x_{43} & 0 & & & \\
c_6 & c_6 & x_{61} & x_{62} & x_{63} & x_{64} & x_{65} & & \\
1-c_6 & 1-c_6 & -x_{62} & -x_{61} & -x_{63} & -x_{65} & -x_{64} & 0 & \\
\hline
 & & b_1 & b_1 & b_3 & b_4 & b_4 & b_6 & b_6
\end{array}$$

which has stage order 3, and for which the last two stages have order of consistency 5. All x-variables can thus be expressed in terms of c_4 and c_6 only. We choose the same $c_i s$ as were used with the 8th order formula. In this case the x variables are exactly the same as those appearing in the 8th order formula because we impose the same stage order conditions on both formulae. If this is done, there are three nonquadrature conditions that remain to be satisfied. It turns out that all of these three conditions are satisfied if

$$b_3 = -32\, c_4^2\, (1-c_4)^2\, b_4\,. \tag{5.17}$$

The three quadrature conditions together with (5.17) gives a system which uniquely determines the four b-coefficients. In this way we have constructed a two-parameter family (c_4 and c_6) of symmetric mono-implicit methods of order 6 which have $c = v$ and have seven stages. The coefficients of this 6th order formula are given in section C in the appendix.

6. Computational aspects. In order to compare the computational effort required by the superconvergent deferred correction algorithm described in this paper with that required by a standard deferred correction method such as TWPBVP, we first recall exactly how the code TWPBVP is implemented. The classical deferred correction approach on which the code TWPBVP is based is as follows:

$$\begin{aligned}
&\phi_4(\eta) = 0,\\
&\phi_4(\bar{\eta}) = -\phi_6(\eta),\\
&\phi_4(\bar{\bar{\eta}}) = -\phi_6(\eta) - \phi_8(\bar{\eta}).
\end{aligned} \tag{6.1}$$

Here the vector solutions η, $\bar{\eta}$, and $\bar{\bar{\eta}}$ are of order 4, 6, and 8, respectively. We note, in particular, that since (6.1) is not superconvergent, it is not possible to compute the 8th order solution before the 6th order one is computed. Once $\bar{\eta}$ and $\bar{\bar{\eta}}$ have been computed, we can determine whether or not an accuracy requirement of the form

$$\|\bar{\eta} - \bar{\bar{\eta}}\| < Tol \tag{6.2}$$

is satisfied, where Tol is the requested tolerance. If (6.2) is satisfied, then the solution $\bar{\bar{\eta}}$ is accepted as the final solution, and if not, a remeshing is carried out. The particular remeshing algorithm used in TWPBVP is one which seeks to equidistribute the deferred correction term

$$\phi_8(\bar{\eta}). \tag{6.3}$$

In the case of superconvergent schemes, however, the situation is quite different. Now the deferred correction scheme can be regarded as having the form (6.1), where the third stage is

$$\phi_4(\bar{\bar{\eta}}) = -\bar{\phi}_8(\eta). \tag{6.4}$$

Note, in particular, that both the deferred correction terms in this modified superconvergent scheme depend only on the 4th order solution η. Furthermore, the only role of the $\bar{\eta}$ term is as an error estimate, and this particular aspect of the algorithm has some important consequences. The most important of these is the fact that $\bar{\eta}$ and $\bar{\bar{\eta}}$ can be computed independently of each other. Indeed, in a parallel environment these two terms can be computed at the same time. This means that in the linear case we can solve for the error estimate without knowing either $\bar{\eta}$ or $\bar{\bar{\eta}}$. This is done in the following way. From the last two equations of the deferred correction scheme we have

$$\phi_4(\bar{\bar{\eta}} - \bar{\eta}) = \phi_6(\eta) - \bar{\phi}_8(\eta). \tag{6.5}$$

Thus the revised algorithm for the linear case now is

$$\begin{aligned} \phi_4(\eta) &= 0, \\ \phi_4(\bar{\bar{\eta}} - \bar{\eta}) &= \phi_6(\eta) - \bar{\phi}_8(\eta). \end{aligned} \tag{6.6}$$

Having computed the error estimate $\bar{\bar{\eta}} - \bar{\eta}$ from (6.6), we now see if it satisfies the error criterion (6.2). If it does, then we accept the current grid as the final one and compute the solution $\bar{\bar{\eta}}$ on this grid using the first equation of (6.1) together with (6.4). If the error estimate (6.2) is not satisfied, then we remesh using an algorithm based on the equidistribution of $\phi_6(\eta) - \bar{\phi}_8(\eta)$. The saving in computational effort results from the fact that we do not need to compute $\bar{\bar{\eta}}$ until the error estimate is satisfied. Ideally we would like to be able to extend this algorithm to the nonlinear case, but we have been unable to do this.

The computational effort per iteration for the two schemes is rather similar. Both require the solution of the 4th order formula

$$\phi_4(\eta) = 0 \tag{6.7}$$

initially. The standard scheme (6.1) then requires 14 function evaluations to compute the deferred correction terms while the modified scheme requires 13. The relative performance of the two algorithms is, however, very problem dependent due to their complicated structure. In particular, it depends on the number of iterations required to solve $\phi_4(\eta) = 0$, the relative cost of the linear algebra compared with that of function evaluations, and the relative accuracies of the 6th and 8th order formulae in the deferred correction schemes. However, to give some idea of the relative performance we consider the following two problems.

PROBLEM 1 (see [15]). *The solution has a boundary layer of width $O(\sqrt{\epsilon})$ at $x = 0$.*

$$\epsilon y'' - y = 0, \quad y(0) = 1, \quad y(1) = 0.$$

The exact solution is $y(x) = (\exp(-x/\sqrt{\epsilon}) - \exp((x-2)/\sqrt{\epsilon}))/(1 - \exp(-2/\sqrt{\epsilon}))$.

PROBLEM 2 (see [15]). *The solution has a corner layer at $x = 0.745$.*

$$\epsilon y'' + (y')^2 = 1, \quad y(0) = 1 + \epsilon \ln\cosh(-0.745/\epsilon), \quad y(1) = 1 + \epsilon \ln\cosh(0.255/\epsilon).$$

The exact solution is $y(x) = 1 + \epsilon \ln \cosh(x - 0.745)/\epsilon)$.

In Tables 6.1 and 6.2 we give the results obtained for the numerical solution of these two problems using a tolerance of 10^{-8}. As we explained in the previous section, there are some arbitrary parameters to be chosen in our 8th order formula. In our code we chose $y_{12,8} = 0.1$, c_4=0.2, $c_6 = 0.25$, c_8=0.3, c_{10}=0.85, and $c_{12} = 0.6$, and the complete formula is given in the appendix.

TABLE 6.1
Results for Problem 1.

		TWPBVP			NEW	
ε	N	Time	Mesh err	N	Time	Mesh err
10^{-1}	7	.01	0.1d-8	28	.03	0.3d-13
10^{-2}	25	.03	0.3d-10	24	.02	0.2d-9
10^{-3}	35	.05	0.9d-9	35	.04	0.1d-9
10^{-4}	68	.06	0.7d-11	51	.05	0.2d-10
10^{-5}	52	.05	0.8d-9	45	.05	0.2d-8
10^{-6}	64	.10	0.5d-10	84	.07	0.3d-9
10^{-7}	90	.21	0.7d-11	101	.15	0.3d-9

The headings in the tables can be explained as follows. "N" indicates the number of mesh subintervals in the final mesh for each problem. "Time" denotes the run time in seconds. "Mesh err" is the maximum error that results from comparing the order 8 solution (computed at the mesh points) with the analytic solution. Under "NEW" we list the results obtained using the superconvergent scheme derived in section 5, while "TWPBVP" indicates the results obtained using the code TWPBVP.f obtained from NETLIB.

TABLE 6.2
Results for Problem 2.

		TWPBVP			NEW	
ε	N	Time	Mesh err	N	Time	Mesh err
10^{-1}	44	.13	0.42d-11	26	.09	0.4d-9
10^{-2}	51	.31	0.43d-10	63	.49	0.4d-8
10^{-3}	598	11.1	0.11d-10	94	9.4	0.4d-9

As can be seen from the two tables, the results obtained are quite comparable. Neither algorithm is consistently better than the other, and what these results indicate is that the superconvergent scheme is competitive with the widely used code TWPBVP.f. Also it is important to recall that the superconvergent scheme has associated with it an 8th order interpolant whereas the code TWPBVP.f has no interpolant. This makes it possible for the superconvergent scheme to provide a continuous solution, and this is an important advantage over previously derived MIRK schemes.

Finally, we wish to summarize the main points that have been achieved in this paper. The first important point that we have demonstrated is the possibility of obtaining superconvergence. In particular, we have derived an algebraic criterion for superconvergence and shown by some practical examples that this is actually achieved in practice. In addition, we have derived a framework which will allow us to derive high order MIRK methods. It is interesting to note that the original 8th order MIRK formula derived by Singhal took more than 40 pages of algebra to derive. The framework that we have derived makes the derivation of 8th order MIRK formulae very straightforward. We have not in any way optimized our 8th

order formula, and this is something worth doing in the future. In addition, we have derived a superconvergent MIRK formula with an 8th order interpolant. This has solved a very difficult outstanding problem since previous papers have derived only interpolants of order 4 and 6. Since our basic algorithm is of order 8, it is clear that it is an 8th order interpolant that is required. Finally, we remark that the theory of superconvergence developed in this paper is sufficiently general to be applicable to large classes of parameterized Runge–Kutta methods. We plan to investigate the extension of these ideas to other classes of Runge–Kutta methods in the future.

Appendix. In section A we give the MIRKA formula of order 8 in 13 stages. The $\hat{b}$ coefficients of the embedded method of order 6 are given in section B. In section C we present the MIRKA formula of order 6 in seven stages.

A. The 8th order MIRKA method. An 8th order formula in 13 stages is of the form

$$
\begin{array}{c|c|cccccccccccc}
0 & 0 & & & & & & & & & & & & \\
1 & 1 & 0 & & & & & & & & & & & \\
\frac{1}{2} & \frac{1}{2} & \frac{1}{8} & -\frac{1}{8} & & & & & & & & & & \\
c_4 & c_4 & x_{41} & x_{42} & x_{43} & & & & & & & & & \\
1-c_4 & 1-c_4 & -x_{42} & -x_{41} & -x_{43} & 0 & & & & & & & & \\
c_6 & c_6 & x_{61} & x_{62} & x_{63} & x_{64} & x_{65} & & & & & & & \\
1-c_6 & 1-c_6 & -x_{62} & -x_{61} & -x_{63} & -x_{65} & -x_{64} & 0 & & & & & & \\
c_8 & c_8 & x_{81} & x_{82} & x_{83} & x_{84} & x_{85} & x_{86} & x_{87} & & & & & \\
1-c_8 & 1-c_8 & -x_{82} & -x_{81} & -x_{83} & -x_{85} & -x_{84} & -x_{87} & -x_{86} & 0 & & & & \\
c_{10} & c_{10} & x_{10,1} & x_{10,2} & x_{10,3} & x_{10,4} & x_{10,5} & x_{10,6} & x_{10,7} & x_{10,8} & x_{10,9} & & & \\
1-c_{10} & 1-c_{10} & -x_{10,2} & -x_{10,1} & -x_{10,3} & -x_{10,5} & -x_{10,4} & -x_{10,7} & -x_{10,6} & -x_{10,9} & -x_{10,8} & 0 & & \\
c_{12} & c_{12} & x_{12,1} & x_{12,2} & x_{12,3} & x_{12,4} & x_{12,5} & x_{12,6} & x_{12,7} & x_{12,8} & x_{12,9} & x_{12,10} & x_{12,11} & \\
1-c_{12} & 1-c_{12} & -x_{12,2} & -x_{12,1} & -x_{12,3} & -x_{12,5} & -x_{12,4} & -x_{12,7} & -x_{12,6} & -x_{12,9} & -x_{12,8} & -x_{12,11} & -x_{12,10} & 0 \\
\hline
 & & b_1 & b_1 & b_3 & b_4 & b_4 & b_6 & b_6 & b_8 & b_8 & b_{10} & b_{10} & b_{12}\ b_{12}
\end{array}
$$

In what follows we give the coefficients of this formula.

$c_1 = 0, c_2 = 1.0, c_3 = 0.5, c_4 = 0.2, c_5 = 0.8, c_6 = 0.25, c_7 = 0.75,$

$c_8 = 0.3, c_9 = 0.7, c_{10} = 0.85, c_{11} = 0.15, c_{12} = 0.6, c_{13} = 0.4.$

$b_1 = \frac{2663}{59976}, b_3 = 0, b_4 = 0, b_6 = 0, b_8 = \frac{100}{14553},$

$b_{10} = \frac{24320}{106029}, b_{12} = \frac{995}{4536}.$

$x_{3,1} = \frac{1}{8}, x_{3,2} = -x_{3,1}, x_{4,1} = \frac{14}{125},$

$x_{4,2} = \frac{-6}{125}, x_{4,3} = \frac{-8}{125},$

$x_{6,1} = \frac{237}{4096}, x_{6,2} = \frac{-81}{4096},$

$x_{6,3} = \frac{-61}{576}, x_{6,4} = \frac{4625}{36864}, x_{6,5} = \frac{-2125}{36864},$

$x_{8,1} = \frac{155471}{4000000}, x_{8,2} = \frac{-74479}{4000000}, x_{8,3} = \frac{-7661}{140625}, x_{8,4} = \frac{40801}{126720},$

$x_{8,5} = \frac{-9473}{126720}, x_{8,6} = \frac{-33004}{171875}, x_{8,7} = \frac{-3604}{171875},$

$y_{12,8} = 0.1,$

$x_{10,1} = \frac{569379313995403}{56995228416000000} + \frac{111639.0}{15564800.0}\, y_{12,8},$

$x_{10,2} = \frac{-2759611659786347}{56995228416000000} - \frac{111639}{15564800}\, y_{12,8},$

$x_{10,3} = \frac{-198705503}{2439000000}, x_{10,4} = \frac{6805491736321}{85981373153280} - \frac{213129}{622592}\, y_{12,8},$

$x_{10,5} = \frac{-12000874456673}{85981373153280} + \frac{213129}{622592}\, y_{12,8}, x_{10,6} = \frac{-86344040636419}{524788654500000} + \frac{23681}{30400}\, y_{12,8},$

$x_{10,7} = \frac{-77917712095669}{524788654500000} - \frac{23681}{30400}\, y_{12,8}, x_{10,8} = \frac{1224392774341}{6839427409920} - \frac{37213}{77824}\, y_{12,8},$

$x_{10,9} = \frac{2149691448091}{6839427409920} + \frac{37213}{77824}\, y_{12,8},$

$x_{12,1} = \frac{480654349561207}{17343030480562500} - \frac{3}{400}\, y_{12,8},$

$x_{12,2} = \frac{-349408852772543}{17343030480562500} + \frac{3}{400}\, y_{12,8}, x_{12,3} = \frac{15647244464}{53086359375} x_{12,4} = \frac{-3470436703}{1539011780460} + \frac{63}{176}\, y_{12,8},$
$x_{12,5} = \frac{367646969159}{1539011780460} - \frac{63}{176}\, y_{12,8}, x_{12,6} = \frac{719445043186076}{1202352953484375} - \frac{224}{275}\, y_{12,8},$
$x_{12,7} = \frac{358555969186076}{1202352953484375} + \frac{224}{275}\, y_{12,8}, x_{12,8} = \frac{-702597274}{1308047895} + \frac{1}{2}\, y_{12,8}, x_{12,9} = \frac{-702597274}{1308047895.0} - \frac{1}{2}\, y_{12,8},$
$x_{12,10} = \frac{-168424068}{536350969}, x_{12,11} = \frac{-25336068}{536350969}.$

B. The 6th order embedded MIRKA method. The three nonzero $\hat{b}$ coefficients for the 6th order embedded formula are

$$\hat{b_1} = \frac{1431859}{17459442},$$
$$\hat{b_6} = \frac{10346336}{52378326},$$
$$\hat{b_8} = \frac{11547250}{52378326}.$$

C. The 6th order MIRKA method. The b coefficients for the 6th order formula appearing in section 5.2 are

$$b_1 = \frac{6623}{79920},$$
$$b_3 = \frac{18944}{239760},$$
$$b_4 = \frac{-23125}{239760},$$
$$b_6 = \frac{37888}{79920}.$$

The c_i and $x_{i,j}$ coefficients are the same as in section A.

Acknowledgments. The authors are grateful to two anonymous referees whose suggestions greatly improved the presentation of this paper.

REFERENCES

[1] Z. Bashir-Ali, J.R. Cash, and H.H.M. Silva, *Lobatto deferred correction for stiff two-point boundary value problems*, Comput. Math. Appl., 36 (1998), pp. 59–69.
[2] W.M.G. Van Bokhoven, *Implicit end-point quadrature formulae*, BIT, 20 (1980) pp. 87–99.
[3] K. Burrage, F.H. Chipman, and P.H. Muir, *Order results for mono-implicit Runge–Kutta methods*, SIAM J. Numer. Anal., 31 (1994), pp. 876–891.
[4] J.C. Butcher, *The Numerical Analysis of Ordinary Differential Equations: Runge–Kutta and General Linear Methods*, Wiley, Chichester, UK, 1987.
[5] J.R. Cash, *A variable order deferred correction algorithm for the numerical solution of nonlinear two point boundary value problems*, Comput. Math. Appl., 9 (1983), pp. 257–265.
[6] J.R. Cash and H.H.M. Silva, *Iterated deferred correction for linear two-point boundary value problems*, Comput. Appl. Math., 15 (1996), pp. 55–75.
[7] J.R. Cash, *On the numerical integration of nonlinear two-point boundary value problems using iterated deferred corrections. Part 2: The development and analysis of highly stable deferred correction formulae*, SIAM J. Numer. Anal., 25 (1988), pp. 862–882.
[8] J.R. Cash and M.H. Wright, *A deferred correction method for nonlinear two-point boundary value problems: Implementation and numerical evaluation*, SIAM J. Sci. Statist. Comput.,12 (1991), pp. 971–989.
[9] J.R. Cash, G. Moore, and R.W. Wright, *An automatic continuation strategy for the solution of singularly perturbed linear two point boundary value problems*, J. Comput. Phys., 122 (1995), pp. 266–279.
[10] J.R. Cash, *A class of implicit Runge-Kutta methods for the numerical integration of stiff ordinary differential equations*, J. ACM, 22 (1975), pp. 504–511.
[11] J.R. Cash and A. Singhal, *High order methods for the numerical solution of two-point boundary value problems*, BIT, 22 (1982), pp. 184–199.
[12] J.R. Cash and A. Singhal, *Mono-implicit Runge-Kutta formulae for the numerical integration of stiff differential equations*, IMA J. Numer. Anal., 2 (1982), pp. 221–227.
[13] W.H. Enright and P.H. Muir, *Efficient classes of Runge-Kutta methods for two-point boundary value problems*, Computing, 37 (1986), pp. 315–334.

[14] E. Hairer, S.P. Nørsett, and G. Wanner, *Solving Ordinary Differential Equations* I, *Non-stiff Problems*, 2nd ed., Springer-Verlag, Heidelberg, 1993.
[15] P.W. Hemker, *A Numerical Study of Stiff Two-Point Boundary Value Problems*, Mathematisch Centrum, Amsterdam, 1977.
[16] M. Lentini and V. Pereyra, *An adaptive finite difference solver for nonlinear two-point boundary problems with mild boundary layers*, SIAM J. Numer. Anal., 14 (1977), pp. 91–111.
[17] J.D. Lambert, *Computational Methods in Ordinary Differential Equations*, Wiley, London, 1973.
[18] K.H. Schild, *Gaussian collocation via defect correction*, Numer. Math., 58 (1990), pp. 369–386.
[19] R.D. Skeel, *A theoretical framework for proving accuracy results for deferred corrections*, SIAM J. Numer. Anal., 19 (1982), pp. 171–196.
[20] M. Van Daele, T. Van Hecke, G. Vanden Berghe, and H. De Meyer, *Deferred correction with mono-implicit Runge-Kutta methods for first order IVPs*, J. Comput. Appl. Math., 111 (1999), pp. 37–47.
[21] T. Van Hecke, *Mono-impliciete Runge-Kutta(-Nyström) methodes*, Ph.D. thesis, University of Ghent, Ghent, Belgium, 1998.

SIAM J. SCI. COMPUT.
Vol. 22, No. 5, pp. 1717–1740
© 2000 Society for Industrial and Applied Mathematics

FAST SOLUTION OF THE RADIAL BASIS FUNCTION INTERPOLATION EQUATIONS: DOMAIN DECOMPOSITION METHODS*

R. K. BEATSON†, W. A. LIGHT‡, AND S. BILLINGS§

Abstract. In this paper we consider domain decomposition methods for solving the radial basis function interpolation equations. There are three interwoven threads to the paper. The first thread provides good ways of setting up and solving small- to medium-sized radial basis function interpolation problems. These may occur as subproblems in a domain decomposition solution of a larger interpolation problem. The usual formulation of such a problem can suffer from an unfortunate scale dependence not intrinsic in the problem itself. This scale dependence occurs, for instance, when fitting polyharmonic splines in even dimensions. We present and analyze an alternative formulation, available for all strictly conditionally positive definite basic functions, which does not suffer from this drawback, at least for the very important example previously mentioned. This formulation changes the problem into one involving a strictly positive definite symmetric system, which can be easily and efficiently solved by Cholesky factorization. The second section considers a natural domain decomposition method for the interpolation equations and views it as an instance of von Neumann's alternating projection algorithm. Here the underlying Hilbert space is the reproducing kernel Hilbert space induced by the strictly conditionally positive definite basic function. We show that the domain decomposition method presented converges linearly under very weak nondegeneracy conditions on the possibly overlapping subdomains. The last section presents some algorithmic details and numerical results of a domain decomposition interpolatory code for polyharmonic splines in 2 and 3 dimensions. This code has solved problems with 5 million centers and can fit splines with 10,000 centers in approximately 7 seconds on very modest hardware.

Key words. radial basis functions, interpolation equations, fast solution method

AMS subject classifications. 65D05, 65D07, 65F10

PII. S1064827599361771

1. Introduction.

One of the most basic problems in approximation theory is to construct an approximation to data specified at m distinct points $x_1, \ldots, x_m$ in $\mathcal{R}^n$. A simple approach consists of choosing m functions and then looking for the unique combination of these functions which interpolates the data at the given points. For this process to be successful, the set of functions chosen must be linearly independent over the set of interpolation points $x_1, \ldots, x_m$. If the functions are $u_1, \ldots, u_m$, then this is equivalent to the matrix $\big(u_j(x_i)\big)$ being invertible. To implement this program, the user must make a choice of the functions $u_1, \ldots, u_m$. However, it is sometimes more natural to specify not the actual basis to be used, but rather the subspace given by $\text{span}\{u_1, \ldots, u_m\}$. This is the approach we intend to study. We consider interpolants of the form

$$s = p + \sum_{j=1}^{m} \lambda_j \Phi(\cdot, x_j). \tag{1.1}$$

*Received by the editors September 30, 1999; accepted for publication (in revised form) June 14, 2000; published electronically December 28, 2000.
http://www.siam.org/journals/sisc/22-5/36177.html

†Department of Mathematics and Statistics, University of Canterbury, Private Bag 4800, Christchurch, New Zealand (r.beatson@math.canterbury.ac.nz). The work of this author was partially supported by PGSF Subcontract DRF 601 and EPSRC Visiting Fellowship GR/M/52595.

‡Department of Mathematics and Computer Science, University of Leicester, Leicester, LE1 7RH, United Kingdom (pwl@mcs.le.ac.uk). The work of this author was supported by EPSRC grant GR/L/36222.

§Geophysical Technology Ltd., P.O. Box U9, The University of New England, Armidale, NSW 2351, Australia (stephenbillings@yahoo.com).

Here $X = \{x_1, \ldots, x_m\}$ is a given set of distinct centers, p is a polynomial of total degree at most $k-1$, and Φ is a mapping from $\mathcal{R}^n \times \mathcal{R}^n \to \mathcal{R}$. Because there are more parameters than data, we impose the further conditions that the coefficient vector λ satisfies

$$\sum_{j=1}^{m} \lambda_j q(x_j) = 0 \tag{1.2}$$

for all polynomials q of degree at most $k-1$. The related interpolation problem follows.

PROBLEM 1.1. *Given distinct nodes $X = \{x_1, \ldots, x_m\} \subset \mathcal{R}^n$ unisolvent with respect to the polynomials of degree at most $k-1$, and a vector $d \in \mathcal{R}^m$, find a function s of the form specified by* (1.1) *and* (1.2)*, satisfying the interpolation conditions*

$$s(x_i) = d_i, \qquad 1 \le i \le m.$$

Let π_{k-1} denote the subspace of $C(\mathcal{R}^n)$ consisting of all polynomials of total degree $k-1$, and let $\{p_1, p_2, \ldots, p_\ell\}$ be a basis for π_{k-1}. Then the corresponding system of equations is often written in matrix terms as

$$\begin{pmatrix} \mathsf{A} & \mathsf{P} \\ \mathsf{P}^T & \mathsf{O} \end{pmatrix} \begin{pmatrix} \lambda \\ c \end{pmatrix} = \begin{pmatrix} d \\ 0 \end{pmatrix}, \tag{1.3}$$

in which

$$\mathsf{A}_{ij} = \Phi(x_i, x_j), \qquad i, j = 1, \ldots, m, \tag{1.4}$$

and

$$\mathsf{P}_{ij} = p_j(x_i), \qquad i = 1, \ldots, m, \quad j = 1, \ldots, \ell. \tag{1.5}$$

Note that in this system, and throughout the paper, we use the symbol O to represent the zero matrix of appropriate size, while the symbol 0 is used for the zero vector/scalar.

The above setting covers natural spline interpolation in $\mathcal{R}$ and interpolation by radial basis functions in $\mathcal{R}^n$. In the latter case, the function Φ has the form $\Phi(x, y) = \phi(|x-y|)$ for $x, y \in \mathcal{R}^n$, where $\phi : \mathcal{R} \to \mathcal{R}$, and $|\cdot|$ is the Euclidean norm in $\mathcal{R}^n$. This case is of considerable practical importance. Problem 1.1 is known to have a unique solution for many choices of Φ. To discuss these results we will need the following definition.

DEFINITION 1. *A symmetric function $\Phi : \mathcal{R}^n \times \mathcal{R}^n \to \mathcal{R}$ is strictly conditionally positive definite of order k on $\mathcal{R}^n$ if for all sets $X = \{x_1, \ldots, x_m\} \subset \mathcal{R}^n$ of distinct points, and all vectors $\lambda \in \mathcal{R}^m$ satisfying the orthogonality condition* (1.2)*, the quadratic form*

$$\lambda^T \mathsf{A} \lambda = \sum_{i,j=1}^{m} \lambda_i \lambda_j \Phi(x_i, x_j)$$

is positive whenever $\lambda \neq 0$.

The paper of Micchelli [11] showed that the interpolation system (1.3) is invertible whenever the function $\Phi(x, y) = \phi(|x-y|)$ is strictly conditionally positive definite of

order k, and X is unisolvent for π_{k-1}.[1] Earlier work in a similar vein can be found in Schoenberg [21]. It is worth noting that if a function Φ is conditionally positive definite of order k, then it is also conditionally positive definite of order k' for any $k' \geq k$, and so there is some flexibility over the choice of the parameter k.

For radial basis function interpolation, the most popular choices of the function Φ are strictly conditionally positive definite, notably the multiquadrics and the polyharmonic splines. These interpolation methods are popular for a number of reasons. First, the interpolant depends only on the radial distance between the interpolation points. This feature makes radial basis function interpolation suitable for treating unstructured data—particular data which does not fall on, or close to, the points of some rectangular or regular triangular grid. Such data is often described as scattered. We have already indicated that in practical cases, the types of geometry for which the interpolant is not specified uniquely are extremely simple. For example, interpolation by thin-plate splines in $\mathcal{R}^2$ requires only that the nodes must not lie on a line. With a scattered data set, one can almost guarantee that this condition will be satisfied. This is in sharp contrast to the situation for polynomial interpolation, where the conditions on the data sets required in order that a unique interpolant exists get progressively more complex as the number of interpolation points increases. Furthermore, all the radial basis function interpolants in use have the feature that they minimize a seminorm. Sometimes (as is the case with polyharmonic spline interpolants—see below) the seminorm can be seen to force the interpolant to be the "smoothest" interpolant in some sense. This smoothness property can actually be seen in graphical representations of the interpolant and is akin to the smoothness experienced when interpolating by a low degree polynomial spline in $\mathcal{R}$.

Despite the many nice features of radial basis function interpolation, early workers in the field encountered significant computational difficulties. A small part of these difficulties was due to the computing equipment available at the time, but the major problem was that system (1.3) led to an very ill-conditioned system of equations with a full matrix. We refer the reader to [1] for quotes concerning these difficulties. Further discussion of the ill-conditioning can be found in [10]. In [12] lower bounds on the condition number were given, which showed the poor conditioning of the matrix of system (1.3) as a function of the minimum separation between the interpolation points. It is clear that direct solution methods are not feasible for large data sets. Therefore, a number of iterative methods have been proposed, of which [1] and [4] are recent examples.

This paper focuses on various aspects of the numerical solution of interpolation problems using interpolants of the type described in (1.1). Our prime objective is to present a method for large problems with many, many nodes. However, we will also need good methods for small subproblems that occur naturally in our solution strategy. Two questions arise immediately. First, is (1.1) the best way of couching the interpolation problem, or are there other bases for the finite-dimensional space

$$\mathcal{S} = \left\{ p + \sum_{j=1}^{m} \lambda_j \Phi(\cdot, x_j) : \textstyle\sum_{j=1}^{m} \lambda_j q(x_j) = 0 \text{ for all } q \in \pi_{k-1} \right\}$$

which are numerically superior? Second, once we have decided on a suitable choice of basis, what solver do we use for the linear system generated by this basis?

[1] A suitably modified statement holds when X is not unisolvent for π_{k-1}.

In section 2 we investigate the choice of a basis. We have a number of objectives in this section. First, we do not want to work with a spanning set for $\mathcal{S}$ which is "overspecified" in the sense that it consists of more functions than the dimension of $\mathcal{S}$. Second, we would like the basis to have some natural link to the problem. Third, many of the problems which come within our remit have a certain natural feature—that of scale independence. We want a basis for $\mathcal{S}$, and certainly a method of solution which possesses this same scale independence. Let us be a bit more precise.

DEFINITION 2. *Consider an interpolation method $L_{X,d}$ defined for some suitable class of finite subsets of points $X \subset \mathcal{R}^n$ and data values d. The interpolation method $L_{X,d}$ will be called scale independent if for all suitable sets of interpolation points X and data values d, the interpolant s fitted at the points X to data values d, and the interpolant s^h fitted at the scaled points hX to the unscaled data values d, are related by $s^h(h\,\cdot) = s$.*

As an example, consider the polyharmonic spline interpolants, corresponding to the strictly conditionally positive definite function of order $k+1$,

$$\Phi(x,y) = (-1)^{k+1}|x-y|^{2k}\ln|x-y| \qquad \text{for } x, y \in \mathcal{R}^n.$$

If n is even, then the interpolant corresponding to strict conditional positive definiteness of order $k' = k + (n/2)$, $p \in \pi_{k'-1}$ and sums of shifts of Φ, minimizes the seminorm

$$|g|^2 = \int_{\mathcal{R}^n} \sum_{|\alpha|=k'} \binom{k'}{\alpha} \left(D^\alpha g\right)^2 dx$$

over all suitably smooth interpolating functions.[2] This condition, and the scaling of derivatives, clearly implies that polyharmonic spline interpolation is scale independent. However, we shall see that the numerical methods commonly used for fitting such splines are not scale independent. This is because the chosen basis, or spanning set, behaves badly with respect to scale. We will propose a basis which behaves much better with respect to scale, and which will be very useful for the direct solution of small- to moderately-sized radial basis function interpolation problems.

One particular application for our basis is the direct solution of subproblems arising in a domain decomposition code for large problems. This is the subject of section 4. We wish to suggest that the domain decomposition iterative method of solving a linear system is an effective strategy for the systems considered here. Section 3 introduces a form of the domain decomposition algorithm. This may be viewed as multiplicative Schwarz, or as the alternating projection algorithm of von Neumann [13]. The main benefits of the latter viewpoint are that there is a lot of information available about the convergence of the alternating algorithm and the fact that in favorable circumstances the rate of convergence is linear. We will establish linear convergence of our domain decomposition method, and give some theoretical estimates of the rate, based on the idea of an angle between two subspaces in a Hilbert space.

The description of the domain decomposition method in section 3 is quite general and abstract. In section 4 we present some details of the functioning domain decomposition code and some numerical results.

We conclude this introduction with some remarks about notation. The sign $|\cdot|$ appears frequently throughout the paper and has a number of different meanings. For

[2]It should be clear to the discerning reader that a number of technicalities have been omitted here. Furthermore, it is not really necessary to impose the restriction that n is even, but we wanted to keep the discussion simple.

example, if $x \in \mathcal{R}^n$, then $|x|$ means the Euclidean norm of x. If α is a nonnegative multi-integer, that is, $\alpha \in \mathcal{Z}_+^n$, then $|\alpha|$ is the "length" of α—the sum of its coordinates. Frequent use will be made of the subspace π_{k-1} of $C(\mathcal{R}^n)$ consisting of all algebraic polynomials of total degree at most $k-1$.

2. Alternative bases for the interpolation problem. If the data values $d_1, \ldots, d_m$ are given on $x_1, \ldots, x_m \in \mathcal{R}^n$, then the radial basis function interpolant we seek is a function drawn from the subspace

$$\mathcal{S} = \left\{ s : s = p + \sum_{j=1}^{m} \lambda_j \phi(|\cdot - x_j|) : \ p \in \pi_{k-1}, \lambda_1, \ldots, \lambda_m \in \mathcal{R} \right.$$
$$\left. \text{and } \sum_{j=1}^{m} \lambda_j q(x_j) = 0 \text{ for all } q \in \pi_{k-1} \right\}.$$

Because the interpolation problem is well posed when Φ is strictly conditionally positive definite of order k, it must be that this space has dimension m. An alternative approach is therefore to develop a basis consisting of only m functions. The first results in this direction were obtained by Sibson and Stone [23], who used the thin-plate spline radial basic function $\Phi(x,y) = |x-y|^2 \log |x-y|$ together with linear polynomials in $\mathcal{R}^2$. As pointed out in the introduction, this choice of radial basic function is a very popular one that has been very successful in applications. In this section we present an alternative approach which is simple to follow both conceptually and numerically and can be carried out for any of the radial basis functions in common usage. It will turn out that our approach is equivalent to that of Sibson and Stone in the case of thin-plate splines in $\mathcal{R}^2$.

The technical fact on which our approach is based is that every basic function which is strictly conditionally positive definite of some order generates a reproducing kernel Hilbert space. That is, there is a Hilbert space $\mathcal{H}$ consisting of continuous functions on $\mathcal{R}^n$ and such that for each $x \in \mathcal{R}^n$ the mapping $f \mapsto f(x)$ for $f \in \mathcal{H}$ is bounded. It then follows easily from the Riesz representation theorem that there is a unique kernel $K : \mathcal{R}^n \times \mathcal{R}^n \to \mathcal{R}$ with the following properties.

(i) $K(x,y) = K(y,x)$ for all $x, y \in \mathcal{R}^n$.
(ii) $K(\cdot, x) \in \mathcal{H}$ for all $x \in \mathcal{R}^n$.
(iii) K is strictly positive definite. That is, for any choice of points $x_1, \ldots, x_m \in \mathcal{R}^n$ and numbers $\lambda_1, \ldots, \lambda_m \in \mathcal{R}$, $\sum_{i,j=1}^{m} \lambda_i \lambda_j K(x_i, x_j) \geq 0$, with equality only when $\boldsymbol{\lambda} = 0$.
(iv) $f(y) = \langle f, K(\cdot, y) \rangle$ for all $f \in \mathcal{H}$ and $y \in \mathcal{R}^n$.

Some other results about reproducing kernels help us in our analysis. Once Φ is given, and the degree of conditional positive definiteness is known, it is possible to write down the reproducing kernel for $\mathcal{H}$ explicitly. To do this we must first make a choice about how to define $\mathcal{H}$. Suppose Φ is strictly conditionally positive definite of order k on $\mathcal{R}^n$. Let $\dim(\pi_{k-1}) = \ell$, and let $\{x_1, \ldots, x_\ell\} \subset \mathcal{R}^n$ be a set of points which is unisolvent with respect to π_{k-1}. The conditional positive definiteness of Φ allows us to define a seminorm $|\cdot|$, with $\mathcal{H}$ then being the set of all functions for which this seminorm is finite. The seminorm has kernel π_{k-1}. The seminorm is modified into a norm simply by choosing an appropriate norm on π_{k-1} and adding it to the seminorm. We always make the choice

$$\|f\|^2 = \sum_{i=1}^{\ell} \big(f(x_i)\big)^2 + |f|^2 \qquad \text{for } f \text{ in } \mathcal{H}.$$

Let $p_1, \ldots, p_\ell$ be the Lagrange basis for π_{k-1} with respect to the points $x_1, \ldots, x_\ell$. That is, the p_j are chosen such that $p_j(x_i)$ is 1 if $i = j$ and is zero otherwise, $1 \le i, j \le \ell$. Then the reproducing kernel K is given by

$$\begin{aligned} K(x,y) = \Phi(x,y) &- \sum_{j=1}^{\ell} p_j(y)\Phi(x,x_j) - \sum_{i=1}^{\ell} p_i(x)\Phi(x_i,y) \\ &+ \sum_{i=1}^{\ell}\sum_{j=1}^{\ell} p_i(x)p_j(y)\Phi(x_i,x_j) + \sum_{i=1}^{\ell} p_i(x)p_i(y) \quad \text{for } x, y \in \mathcal{R}^n. \end{aligned} \tag{2.1}$$

Here we have tried to give a very rapid sketch of matters which are peripheral to our concerns in this paper. Full details may be found in [8]. Our reason for giving a little of the detail here is that we need the following observations. First, it is easily calculated from (2.1) that $K(\cdot, x_j) = p_j$ for $j = 1, \ldots, \ell$. Second, because of the form of the induced inner product in $\mathcal{H}$ (see (3.1)), $p_1, \ldots, p_\ell$ are an orthonormal basis for π_{k-1}. This follows from the observation that

$$\langle p_\mu, p_\nu \rangle = \sum_{i=1}^{\ell} p_\nu(x_i)p_\mu(x_i)$$

for $\nu, \mu = 1, \ldots, \ell$. Let $P : C(\mathcal{R}^n) \to \pi_{k-1}$ be defined by $Pf = \sum_{j=1}^{\ell} f(x_j)p_j$ for each $f \in C(\mathcal{R}^n)$. The above remarks then show that

$$Pf = \sum_{j=1}^{\ell} f(x_j)p_j = \sum_{j=1}^{\ell} \langle f, K(\cdot, x_j)\rangle p_j = \sum_{j=1}^{\ell} \langle f, p_j\rangle p_j,$$

and so P is the orthogonal projection from $\mathcal{H}$ onto π_{k-1}. It is now possible to compress (2.1) significantly:

$$K = (I \otimes I - P \otimes I)(I \otimes I - I \otimes P)\,\Phi + \sum_{j=1}^{\ell} p_j \otimes p_j. \tag{2.2}$$

The operator $P \otimes I + I \otimes P - P \otimes P$ is the Boolean sum $(P \otimes I) \oplus (I \otimes P)$. The radial basis interpolant defined in (1.1) and (1.2) can be expressed in the form $s = \sum_{j=1}^{m} \mu_j K(\cdot, x_j)$ as long as the interpolation points $x_1, \ldots, x_m$ are assumed to contain the set $x_1, \ldots, x_\ell$ which is unisolvent for π_{k-1}. This allows us to immediately achieve our first objective: $\{K(\cdot, x_1), \ldots, K(\cdot, x_m)\}$ is a basis for the subspace $\mathcal{S}$, and the interpolant $s = \sum_{j=1}^{m} \mu_j K(\cdot, x_j)$ is specified by the equations

$$\sum_{j=1}^{m} \mu_j K(x_i, x_j) = f_i, \quad 1 \le i \le m. \tag{2.3}$$

The matrix K with elements

$$\mathsf{K}_{ij} = K(x_i, x_j) \quad \text{for } 1 \le i, j \le m \tag{2.4}$$

is positive definite, and so (2.3) gives a potentially very attractive alternative to (1.3) for solving the interpolation equations. Table 1 compares the conditioning of the matrix K with that of the matrix

$$\mathsf{B} = \begin{pmatrix} \mathsf{A} & \mathsf{P} \\ \mathsf{P}^T & \mathsf{O} \end{pmatrix} \tag{2.5}$$

TABLE 1

Two-norm condition numbers of various matrices corresponding to different formulations of the thin-plate spline interpolation problem. The model problem is on a uniform grid including corners of the unit square with interpoint spacing h along each edge.

Spacing h	Conventional matrix B	Reproducing kernel matrix K	Homogeneous matrix C
1/8	3.5158(3)	1.8930(4)	7.5838(3)
1/16	3.8938(4)	2.6514(5)	1.1086(5)
1/32	5.1363(5)	4.0007(6)	1.6864(6)
1/64	7.6183(6)	6.2029(7)	2.6264(7)

TABLE 2

Two-norm condition numbers of various matrices corresponding to different formulations of the thin-plate spline interpolation problem. The model problem is on a uniform 5×5 grid in $[0, \alpha]^2$.

Scale parameter α	Conventional matrix B	Reproducing kernel matrix K	Homogeneous matrix C
0.001	2.4349(8)	8.4635(8)	5.4938(2)
0.01	2.4364(6)	8.4640(6)	5.4938(2)
0.1	2.5179(4)	8.5134(4)	5.4938(2)
1.0	3.6458(2)	1.3660(3)	5.4938(2)
10	1.8742(6)	1.2609(3)	5.4938(2)
100	1.1520(11)	1.1396(5)	5.4938(2)
1000	3.4590(15)	1.1386(7)	5.4938(2)

of the usual system (1.3), in which the monomials are used as a basis for the polynomials. Also included is a comparison with a different formulation of the interpolation problem to be developed later. The comparison is over sets of uniformly distributed points in $[0, 1]^2$, using the radial basis function $\phi(r) = r^2 \log r$ with the order of conditional positive definiteness equal to 2. In this table an entry of the form $d_0.d_1d_2d_3(e)$, with d_0, d_1, d_2, d_3 decimal digits, represents the number $d_0.d_1d_2d_3 \times 10^e$. It can be seen that the conditioning of the matrices corresponding to all three approaches is roughly the same, when the problem is posed on the unit square. However, the spectrum of the matrix K of (2.4) and also of the matrix (2.5) are affected by the scale of the problem. For example, Table 2 shows calculated condition numbers for the matrices B of (2.5), the matrix K of (2.4), and the matrix C of the different formulation to be developed later (see Theorem 2.4). These matrices correspond to different ways of casting the interpolation problem. The model problem considered is thin-plate spline interpolation on a uniform 5×5 grid in $[0, \alpha]^2$ with interpoint spacing of $\alpha/4$. The table shows the effect of varying the scale parameter α.

Recall from the introduction that the problem itself is scale independent, so that the scale dependence displayed in Table 2 is not intrinsic to the problem. Rather, it is a consequence of the approach taken to the solving the interpolation system. Some years ago Newsam [15] pointed out to one of us that in the case of the usual interpolation system (1.3) the problem is that the matrix A and the matrix P (using the monomial basis for π_{k-1}) scale differently. We will see later that a similar phenomena is happening in the reproducing kernel setting.

If the solution function we seek is scale independent, then the supplying of data at $hx_1, \ldots, hx_m$, $h > 0$, instead of $x_1, \ldots, x_m$ ought not to change the numerical difficulty of the fitting task. The ideal situation would be if K were homogeneous, that is, if for all $h > 0$ and $x, y \in \mathcal{R}^n$, there were some number $\gamma > 0$ such that $K(hx, hy) = h^\gamma K(x, y)$. This would have the effect that the eigenvalues of the scaled interpolation problem would be scaled versions of the eigenvalues of the unscaled interpolation

matrix. Thus, conditioning and eigenvalue clustering would be unaffected by changes of scale. We investigate this possibility next.

LEMMA 2.1. *Let $x_1, \ldots, x_\ell \in \mathcal{R}^n$ be unisolvent with respect to π_{k-1}. Let $p_1, \ldots, p_\ell$ in π_{k-1} be such that $p_i(x_j)$ is 1 if $i = j$ and is zero otherwise, $1 \le i, j \le \ell$. Define $T : C(\mathcal{R}^n \times \mathcal{R}^n) \to C(\mathcal{R}^n \times \mathcal{R}^n)$ by*

$$(Tg)(x,y) = g(x,y) - \sum_{i=1}^{\ell} g(x_i,y)p_i(x) - \sum_{j=1}^{\ell} g(x,x_j)p_j(y) + \sum_{i=1}^{\ell}\sum_{j=1}^{\ell} p_i(x)p_j(y)g(x_i,x_j)$$

for $x, y \in \mathcal{R}^n$ and $g \in C(\mathcal{R}^n \times \mathcal{R}^n)$. Then T annihilates any function g of the form $g(x,y) = p(x-y)$ with $p \in \pi_{2k-1}$.

Proof. It will suffice to establish the result in the case $p(x) = p_\alpha(x) = x^\alpha$, where $x \in \mathcal{R}^n$ and $\alpha \in \mathcal{Z}_+^n$ with $|\alpha| < 2k$. The binomial theorem provides constants a_β, $\beta \in \mathcal{Z}_+^n$, $0 \le \beta \le \alpha$ such that

$$p(x-y) = \sum_{0\le\beta\le\alpha} a_\beta x^{\alpha-\beta}y^\beta = \sum_{0\le\beta\le\alpha} a_\beta p_{\alpha-\beta}(x)p_\beta(y), \quad x, y \in \mathcal{R}^n.$$

Define $g_{\alpha\beta}$ by $g_{\alpha\beta}(x,y) = p_{\alpha-\beta}(x)p_\beta(y)$. Then,

$$Tp_\alpha = \sum_{0\le\beta\le\alpha} a_\beta T g_{\alpha\beta}. \tag{2.6}$$

Now recall the operator $P : C(\mathcal{R}^n) \to \pi_{k-1}$ given by $Pf = \sum_{j=1}^{\ell} f(x_j)p_j$. Then

$$\begin{aligned} Tg_{\alpha\beta} &= g_{\alpha\beta} - (Pp_{\alpha-\beta})\,p_\beta - p_{\alpha-\beta}Pp_\beta + Pp_{\alpha-\beta}Pp_\beta \\ &= (p_{\alpha-\beta} - Pp_{\alpha-\beta}) \otimes (p_\beta - Pp_\beta)\,. \end{aligned}$$

Now since $|\alpha| \le 2k-1$, it follows that either $|\beta| \le k-1$ or $|\alpha-\beta| \le k-1$ in (2.6). Because the operator $I - P$ annihilates π_{k-1}, the result follows. ∎

THEOREM 2.2. *Let the symmetric function $\Phi \in C(\mathcal{R}^n \times \mathcal{R}^n)$ be such that $\Phi(hx, hy) = h^\lambda \Phi(x,y) + q_h(x-y)$ for all $h > 0$ and $x, y \in \mathcal{R}^n$, where $\lambda \in \mathcal{R}$ and $q_h \in \pi_{2k-1}$. Let $x_1, \ldots, x_\ell$ be a unisolvent set of points with respect to π_{k-1}, and let $p_1, \ldots, p_\ell$ be the associated Lagrange basis for π_{k-1}: $p_i(x_j) = \delta_{ij}$, $1 \le i, j \le \ell$, where δ_{ij} is the Kronecker delta. Define the symmetric function $H : \mathcal{R}^n \times \mathcal{R}^n \to \mathcal{R}$ by*

$$H(x,y) = \Phi(x,y) - \sum_{i=1}^{\ell} p_i(x)\Phi(x_i,y) - \sum_{j=1}^{\ell} p_j(y)\Phi(x,x_j) + \sum_{i=1}^{\ell}\sum_{j=1}^{\ell} p_i(x)p_j(y)\Phi(x_i,x_j) \quad \text{for } x, y \in \mathcal{R}^n.$$

For $h > 0$, let H^h be defined in the same way except using points $hx_1, \ldots, hx_\ell$ and the associated Lagrange basis for π_{k-1}, $p_1^h, \ldots, p_\ell^h$. Then $H^h(hx, hy) = h^\lambda H(x,y)$ for all $x, y \in \mathcal{R}^n$.

Proof. From the definition

$$H^h(x,y) = \Phi(x,y) - \sum_{i=1}^{\ell} p_i^h(x)\Phi(hx_i,y) - \sum_{j=1}^{\ell} p_j^h(y)\Phi(x,hx_j) + \sum_{i=1}^{\ell}\sum_{j=1}^{\ell} p_i^h(x)p_j^h(y)\Phi(hx_i,hx_j).$$

Hence,

$$H^h(hx,hy) = \Phi(hx,hy) - \sum_{i=1}^{\ell} p_i^h(hx)\Phi(hx_i,hy) - \sum_{j=1}^{\ell} p_j^h(hy)\Phi(hx,hx_j) + \sum_{i=1}^{\ell}\sum_{j=1}^{\ell} p_i^h(hx)p_j^h(hy)\Phi(hx_i,hx_j).$$

Clearly $p_j^h(hx) = p_j(x)$ for all $x \in \mathcal{R}^n$ and $1 \le j \le \ell$. Thus,

$$H^h(hx,hy) = h^\lambda \left\{ \begin{array}{l} \Phi(x,y) - \displaystyle\sum_{i=1}^{\ell} p_i(x)\Phi(x_i,y) - \sum_{j=1}^{\ell} p_j(y)\Phi(x,x_j) \\ \qquad\qquad + \displaystyle\sum_{i=1}^{\ell}\sum_{j=1}^{\ell} p_i(x)p_j(y)\Phi(x_i,x_j) \end{array} \right\} + (Tv)(x,y),$$

where $v(x,y) = q_h(x-y)$ for some $q_h \in \pi_{2k-1}$, and T is as in Lemma 2.1. By that lemma, $Tv = 0$ and so $H^h(hx,hy) = h^\lambda H(x,y)$ for all $h > 0$ and $x, y \in \mathcal{R}^n$. ∎

We now give an application of Theorem 2.2 to the popular polyharmonic splines.

COROLLARY 2.3. *The function Φ defined by $\Phi(x,y) = (-1)^{k+1}|x-y|^{2k}\ln|x-y|$ is strictly conditionally positive definite of order k' for any $k' \ge k+1$. The corresponding function H satisfies $H^h(hx,hy) = h^{2k}H(x,y)$ for all $x, y \in \mathcal{R}^n$.*

Proof. The conditional positive definiteness can be found in Micchelli [11]. It follows that the reproducing kernel K, and hence the kernel H, are built using points $x_1, \ldots, x_m \in \mathcal{R}^n$ which are unisolvent with respect to $\pi_{k'-1}$. Furthermore,

$$\begin{aligned} \Phi(hx,hy) = (-1)^{k+1}|hx-hy|^{2k}\ln|hx-hy| &= (-1)^{k+1}h^{2k}|x-y|^{2k}\left(\ln|x-y| + \ln h\right) \\ &= h^{2k}\Phi(x,y) + (-1)^{k+1}h^{2k}|x-y|^{2k}\ln h \\ &= h^{2k}\Phi(x,y) + q_h(x-y), \end{aligned}$$

where $q_h \in \pi_{2k} \subset \pi_{2k'-1}$. Theorem 2.2 now applies and $H^h(hx,hy) = h^{2k}H(x,y)$. ∎

The problem with the reproducing kernel matrix is now clearly highlighted. We can write

$$K^h(x,y) = H^h(x,y) + \sum_{i=1}^{\ell} p_i^h(x)p_i^h(y) \quad \text{for } x, y \in \mathcal{R}^n. \tag{2.7}$$

In many cases the H^h function scales at a rate controlled by the basis function with which it is associated. As we just saw in Corollary 2.3, in the thin-plate case with

$\phi(r) = (-1)^{k+1} r^{2k} \ln r$, the kernel H^h scales as a function of h^{2k}. The last term $\sum_{i=1}^{\ell} p_i^h(x) p_i^h(y)$ does not scale at all. Consequently, the general position is that the reproducing kernel is made up of two pieces which scale in fundamentally different ways.

An alternative approach consists of trying to use $\{H(\cdot, x_j)\}_{j=1}^{m}$ as a basis for $\mathcal{S}$. This is, of course, doomed to failure, because we somehow need the terms $\sum_{i=1}^{\ell} p_i^h(x) p_i^h(y)$ which differentiate between K and H. Indeed, one simply checks that $H(\cdot, x_j) = 0$ for $j = 1, \ldots, \ell$. Consequently, we choose as a potential basis the functions $\{p_1, \ldots, p_\ell, H(\cdot, x_{\ell+1}), \ldots, H(\cdot, x_m)\}$. To solve the interpolation problem using this basis we must find $c_1, \ldots, c_\ell, \gamma_{\ell+1}, \ldots, \gamma_m$ such that

$$\sum_{j=1}^{\ell} c_j p_j(x_i) + \sum_{j=\ell+1}^{m} \gamma_j H(x_i, x_j) = d_i, \quad i = 1, \ldots, m.$$

This system of equations has some special structure which we now investigate. First, from the remark above that $H(\cdot, x_j) = 0$ for $1 \le j \le \ell$ and symmetry, it follows in particular that $H(x_i, x_j) = 0$ for $1 \le i \le \ell$ and all j. Thus our system of equations can be written

$$\begin{pmatrix} \mathsf{I} & \mathsf{O} \\ \mathsf{E}^T & \mathsf{C} \end{pmatrix} \begin{pmatrix} c \\ \gamma \end{pmatrix} = d, \tag{2.8}$$

where I is the $\ell \times \ell$ identity, C is the matrix $\big(H(x_i, x_j)\big)_{i,j=\ell+1}^{m}$, E^T is the $(m-\ell) \times \ell$ matrix $\big(p_j(x_i)\big)_{i=\ell+1, j=1}^{m, \ \ell}$, and c, γ, d are the vectors $(c_1, \ldots, c_\ell)^T$, $(\gamma_{\ell+1}, \ldots, \gamma_m)^T$, and $(d_1, \ldots, d_m)^T$, respectively.

THEOREM 2.4. *The matrix* $\mathsf{C} = \big(H(x_i, x_j)\big)_{i,j=\ell+1}^{m}$ *is strictly positive definite.*

Proof. As before, define $P : C(\mathcal{R}^n) \to \pi_{k-1}$ by $Pf = \sum_{j=1}^{\ell} f(x_j) p_j$. Then P is a projection onto π_{k-1}. Furthermore, from (2.2),

$$\begin{aligned} PK(\cdot, x_j) &= \left[(P \otimes I)\, K\right](\cdot, x_j) \\ &= \left[(P \otimes I)\,(I \otimes I - P \otimes I)\,(I \otimes I - I \otimes P)\,\Phi\right](\cdot, x_j) + \sum_{i=1}^{\ell} p_i(x_j) P p_i. \end{aligned}$$

Because $P \otimes I$ is a projection, $(P \otimes I)\,(I \otimes I - P \otimes I) = 0$ and so $PK(\cdot, x_j) = \sum_{i=1}^{\ell} p_i(x_j) p_i$. This calculation shows that for $j = \ell + 1, \ldots, m$,

$$H(\cdot, x_j) = K(\cdot, x_j) - \sum_{i=1}^{\ell} p_i(x_j) p_i = K(\cdot, x_j) - PK(\cdot, x_j). \tag{2.9}$$

Now let $v = (v_{\ell+1}, \ldots, v_m) \in \mathcal{R}^{m-\ell}$. Then,

$$\begin{aligned} v^T \mathsf{C} v &= \sum_{i,j=\ell+1}^{m} v_i v_j H(x_i, x_j) \\ &= \sum_{i,j=\ell+1}^{m} v_i v_j \big\langle H(\cdot, x_i), K(\cdot, x_j) \big\rangle \end{aligned}$$

$$
\begin{aligned}
&= \sum_{i,j=\ell+1}^{m} v_i v_j \big\langle H(\cdot,x_i), H(\cdot,x_j) + PK(\cdot,x_j)\big\rangle \\
&= \sum_{i,j=\ell+1}^{m} v_i v_j \left\{ \big\langle H(\cdot,x_i), H(\cdot,x_j)\big\rangle + \big\langle H(\cdot,x_i), PK(\cdot,x_j)\big\rangle \right\} \\
&= \sum_{i,j=\ell+1}^{m} v_i v_j \left\{ \big\langle H(\cdot,x_i), H(\cdot,x_j)\big\rangle + \big\langle K(\cdot,x_i) - PK(\cdot,x_i), PK(\cdot,x_j)\big\rangle \right\} \\
&= \sum_{i,j=\ell+1}^{m} v_i v_j \big\langle H(\cdot,x_i), H(\cdot,x_j)\big\rangle = \left\| \sum_{j=\ell+1}^{m} v_j H(\cdot,x_j) \right\|^2 \geq 0.
\end{aligned}
$$

In the second line we have used the reproducing property of K; in the third line we have used (2.9); and in the last line we have used the fact that $K(\cdot,x_j) - PK(\cdot,x_j)$ is orthogonal to polynomials of degree at most $k-1$. Furthermore, by the linear independence of $\{K(\cdot,x_j)\}_{j=1}^{m}$ and the fact that $K(\cdot,x_j) = p_j$, $1 \leq j \leq \ell$, the functions $\{H(\cdot,x_j), j = \ell+1, \ldots, m\}$ are linearly independent in the Hilbert space. Hence, $\sum_{j=\ell+1}^{m} v_j H(\cdot,x_j)$ is nonzero unless $v = 0$. It follows that $v^T \mathsf{C} v > 0$ unless $v = 0$ and so C is strictly positive definite. □

To summarize, we have shown that using a basis of the Lagrange polynomials, $\{p_j\}_{j=1}^{\ell}$, supplemented by the functions $\{H(\cdot,x_j) : \ell+1 \leq j \leq m\}$ casts the interpolation problem, Problem 1.1, in the form of (2.8) in which C is a strictly positive definite, symmetric matrix. One way to solve this system is as follows. First "solve" for the coefficients of the Lagrange polynomials setting $c = (d_1, \ldots, d_\ell)^T$. Next substitute the value for c back into the second row of the partitioned system (2.8), obtaining with $d^* = (d_{\ell+1}, \ldots, d_m)^T$ the equation

$$\mathsf{C}\gamma = d^* - \mathsf{E}^T c = \begin{pmatrix} -\mathsf{E}^T & \mathsf{I} \end{pmatrix} d.$$

In this system the right-hand side $\begin{pmatrix} -\mathsf{E}^T & \mathsf{I} \end{pmatrix} d$ is cheap to calculate as each row of E^T has at most ℓ nonzero entries. Once the right-hand side is calculated, the system can be efficiently solved for γ via Cholesky decomposition of C followed by forward and back substitution. This of course takes approximately half the operations of any Gauss elimination routine that does not exploit symmetry, such as might be applied to the usual system (1.3). Also, throughout this whole process one needs to store only E^T and the lower triangle of C, so that the method is efficient in storage as well as operations.

We conclude this section with a brief exploration of the connection between our work and that of Sibson and Stone [23]. Sibson and Stone examined the radial basis function $\phi(r) = r^2 \ln r$ supplemented by linear ploynomials, using data specified at points of $\mathcal{R}^2$. An extension of their approach to general dimension d, and general order k of polynomial, is as follows. Pre- and postmultiply the usual system

$$\begin{pmatrix} \mathsf{A} & \mathsf{P} \\ \mathsf{P}^T & \mathsf{O} \end{pmatrix} \begin{pmatrix} \lambda \\ c \end{pmatrix} = \begin{pmatrix} d \\ 0 \end{pmatrix} \tag{2.10}$$

by an $m \times (m-\ell)$ matrix Q whose columns span the orthogonal complement of the column space of P. This implies in particular that Q has full rank. Then any $\lambda \in \mathcal{R}^m$ satisfying $\mathsf{P}^T\lambda = 0$ can be uniquely expressed in the form $\lambda = \mathsf{Q}\gamma$, where $\gamma \in \mathcal{R}^{m-\ell}$. Thus, the constraints on λ are automatically satisfied, and we need only solve the

unconstrained system $\mathsf{A}\mathsf{Q}\gamma + \mathsf{P}c = d$. Premultiplying by Q^T gives

$$(\mathsf{Q}^T\mathsf{A}\mathsf{Q})\gamma = \mathsf{Q}^T d. \tag{2.11}$$

Observe that for any $u \in \mathcal{R}^{m-\ell}$ with $u \neq 0$, $u^T\left(\mathsf{Q}^T\mathsf{A}\mathsf{Q}\right)u = (\mathsf{Q}u)^T\,\mathsf{A}\,(\mathsf{Q}u) > 0$, since Q has full rank and Φ is strictly conditionally positive definite of order k. Hence, $\mathsf{Q}^T\mathsf{A}\mathsf{Q}$ is a strictly positive definite, symmetric matrix. Therefore, (2.11) uniquely specifies γ and $\lambda = \mathsf{Q}\gamma$. Note also that (2.11) can be rewritten as

$$0 = \mathsf{Q}^T\left(d - \mathsf{A}\mathsf{Q}\gamma\right) = \mathsf{Q}^T\left(d - \mathsf{A}\lambda\right).$$

Hence, $d - \mathsf{A}\lambda$ is in the column space of P, and there is a unique c such that

$$\mathsf{P}c = d - \mathsf{A}\lambda.$$

Thus, we may solve the interpolation problem, Problem 1.1, in the following way.

- Choose an $m \times (m-\ell)$ matrix Q whose columns span the orthogonal complement of the column space of P.
- Find γ by solving the $(m-\ell) \times (m-\ell)$ system (2.11), for example, via Cholesky factorization.
- Set $\lambda = \mathsf{Q}\gamma$.
- Find the polynomial from π_{k-1} interpolating to the following residual function

$$r = d - \sum_{j=1}^{m} \lambda_j \Phi(x, x_j),$$

  at the nodes $\{x_1, \ldots, x_\ell\}$.

A possible choice for Q is

$$\mathsf{Q} = \begin{pmatrix} -p_1(x_{\ell+1}) & -p_1(x_{\ell+2}) & -p_1(x_{\ell+3}) & \ldots & -p_1(x_m) \\ \vdots & \vdots & \vdots & & \vdots \\ -p_\ell(x_{\ell+1}) & -p_\ell(x_{\ell+2}) & -p_\ell(x_{\ell+3}) & \ldots & -p_\ell(x_m) \\ 1 & 0 & 0 & \ldots & 0 \\ 0 & 1 & 0 & \ldots & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & 0 & 0 & \ldots & 1 \end{pmatrix}.$$

This choice of Q is clearly of full rank. Also, we claim that the columns of Q are orthogonal to the columns of P, that is, $\mathsf{P}^T\mathsf{Q} = \mathsf{O}$. Since each row of P^T is of the form $\big(q(x_1), \ldots, q(x_m)\big)$, for some $q \in \pi_{k-1}$, a typical element of $\mathsf{P}^T\mathsf{Q}$ has the form

$$q(x_{\ell+i}) - \sum_{j=1}^{\ell} q(x_j)p_j(x_{\ell+i}) = q(x_{\ell+i}) - q(x_{\ell+i}) = 0,$$

by the properties of the Lagrange basis.

For the particular choice of Q given above, the matrix C in (2.8) obtained via homogenizing the reproducing kernel turns out to be identical with the matrix $\mathsf{Q}^T\mathsf{A}\mathsf{Q}$ obtained via the generalized Sibson and Stone approach.

Powell [16], in the setting of thin-plate splines in $\mathcal{R}^2$, and building on the Sibson and Stone approach, proposes a different choice of Q. His choice is based upon a Householder QR factorization of P.

Our choice of Q reduces the operation count for the setup of the lower triangle of the $\mathsf{Q}^T\mathsf{A}\mathsf{Q}$, and the modified right-hand side $\mathsf{Q}^T d$. When the iterative solution of the large system requires solution of a large number of different small systems, the accumulated setup cost can be a considerable part of the overall cost of a solve. A further convenience of our choice of Q is that it does not intermix the terms $\Phi(\cdot, x_j)$ as much. In certain situations this can allow one to curtail Cholesky back substitutions early, thus saving additional operations.

3. Convergence and the alternating algorithm. This section discusses the convergence of a domain decomposition method for solving the interpolation equations. In the language of the domain decomposition literature the method analyzed is a multiplicative Schwarz type method. It can be viewed equally well as an instance of the von Neumann alternating projection algorithm. We will show that the method presented converges linearly under very weak assumptions on the choice of the subdomains.

The key to the analysis is to view the algorithm as a sequence of orthogonal projections onto overlapping subspaces. Such an approach is standard in the domain decomposition literature [24, 2] and in the analysis of the von Neumann alternating algorithm [13, 14, 25]. In the radial basis function literature, related Hilbert space ideas are used to advantage by Madych and Nelson [9], Faul and Powell [4], Schaback [18, 19] and Schaback and Wendland [20]. After completing this work we learned of some work on different projection methods by Wenz [26].

As was already emphasized in the previous section, interpolation problems involving translates of a basic function can be solved with the aid of the reproducing kernel $K : \mathcal{R}^n \times \mathcal{R}^n \to \mathcal{R}$. It has been known for some time now that this reproducing kernel can be used to define an inner product on a suitable space of continuous functions. It is not necessary for us to operate at the full level of generality of this result. We can simply suppose that f and g mapping $\mathcal{R}^n \to \mathcal{R}$ have the form

$$f = \sum_{i=1}^{m} a_i K(\cdot, x_i) \quad \text{and} \quad g = \sum_{j=1}^{m} b_j K(\cdot, x_j).$$

Define

$$\langle f, g \rangle = \sum_{i,j=1}^{m} a_i b_j K(x_i, x_j). \tag{3.1}$$

There are a number of minor observations at this point. First of all, there is no loss of generality in our requirement that f and g are both linear combinations of $K(\cdot, x_1), \ldots, K(\cdot, x_m)$. If this is not the case, then one simply coalesces the sets on which f and g are based, and sets the appropriate coefficients amongst the a_i and b_j to zero. Second, the symmetry of the reproducing kernel produces the symmetry of the inner product. Third, since K is a reproducing kernel, the matrix $\big(K(x_i, x_j)\big)_{i,j=1}^{m}$ is positive definite. In all cases of interest to us this matrix is, in fact, strictly positive definite,[3] and so $\langle \cdot, \cdot \rangle$ defines an inner product on the set

$$\left\{ \sum_{j=1}^{m} a_j K(\cdot, x_j) : m \in \mathcal{N}, a_1, \ldots, a_m \in \mathcal{R}, x_1, \ldots, x_m \text{ are distinct points of } \mathcal{R}^n \right\}.$$

[3]For some discussion of what can happen in the exceptional cases, see Shapiro [22, p. 71].

Let $\mathcal{H}$ denote the completion of this space with respect to our inner product. In all cases of interest to us, $\mathcal{H} \subset C(\mathcal{R}^n)$, and so we shall assume this throughout what follows.

Throughout this section we will employ the following notation. Given a finite set of distinct points in $\mathcal{R}^n$, $X = \{x_1, \ldots, x_m\}$, we will use the symbol $\mathcal{X}$ to denote the subspace of functions in $\mathcal{H}$ "carried" by X. That is,

$$\mathcal{X} = \left\{ f \in \mathcal{H} : f = \sum_{x \in X} \lambda_x K(\cdot, x), \text{ where } \lambda_x \in \mathcal{R} \right\}.$$

This notation will chiefly be employed with sets of points X or Y, and corresponding sets of functions $\mathcal{X}$ and $\mathcal{Y}$. All these sets will on occasion carry integer subscripts.

Given such a finite set of distinct points $X \subset \mathcal{R}^n$ we consider interpolation problems of the following form:

Given $f \in \mathcal{H}$, find the element of $\mathcal{X}$ that interpolates to f on X.

One version of the domain decomposition method is to let $X_1, X_2, \ldots, X_k$ be subsets of X, such that $\cup_{i=1}^k X_i = X$. Given $f \in \mathcal{H}$, we let $P_i f$, $i = 1, \ldots, k$, denote the function from $\mathcal{X}_i$ that interpolates to f on X_i. Thus,

$$P_i f = \sum_{x \in X_i} \lambda_x K(\cdot, x) \quad \text{and} \quad (P_i f)(y) = f(y) \quad \text{for all } y \in X_i.$$

The domain decomposition algorithm now generates the sequence of residual vectors $\{f_{nk+r}\}$, where $n \in \mathcal{N}_0$ and $r = 1, 2, \ldots, k$ via the relationships

$$f_0 = f \text{ and } f_{nk+r} = f_{nk+r-1} - P_r f_{nk+r-1}.$$

The corresponding sequence of approximations $\{s_{nk+r}\}$ is generated via the relationships

$$s_0 = 0 \text{ and } s_{nk+r} = s_{nk+r-1} + P_r f_{nk+r-1}.$$

The next two results identify interpolation on a subset as the orthogonal projection onto the function space "carried" by that subset. They are key observations utilized by Faul and Powell [4] and Schaback and Wendland [20].

LEMMA 3.1. *Let X be a finite set of distinct points in $\mathcal{R}^n$. Let $f \in \mathcal{H}$ be such that $f(x) = 0$ for all $x \in X$. Let $g \in \mathcal{X}$. Then $\langle f, g \rangle = 0$.*

Proof. Let $X = \{x_1, \ldots, x_m\}$ and $g = \sum_{j=1}^m a_j K(\cdot, x_j)$. First suppose that f has the form $f = \sum_{j=1}^n b_j K(\cdot, x_j)$, where $n \geq m$. Set $a_{m+1} = \cdots = a_n = 0$. Then

$$\begin{aligned} \langle f, g \rangle &= \sum_{i,j=1}^{n} a_i b_j K(x_i, x_j) \\ &= \sum_{i=1}^{m} a_i \sum_{j=1}^{n} b_j K(x_i, x_j) \\ &= \sum_{i=1}^{m} a_i f(x_i) = 0. \end{aligned}$$

If $f \in \mathcal{H}$ does not have the form above, then we can find a sequence $\{h_\ell\} \subset \mathcal{H}$ such that each h_ℓ has the form $h_\ell = \sum_{j=1}^{n_\ell} b_{j\ell} K(\cdot, x_j)$ and $\langle f - h_\ell, f - h_\ell \rangle \to 0$ as $\ell \to \infty$. The above calculations show that

$$\langle h_\ell, g \rangle = \sum_{i=1}^{m} a_i h_\ell(x_i).$$

Since the point evaluation functionals lie in $\mathcal{H}^*$ (because $\mathcal{H}$ is a reproducing kernel Hilbert space) it follows that $h_\ell(x_i) \to 0$ as $\ell \to \infty$ for $i = 1, \ldots, m$. Hence,

$$\langle f, g \rangle = \lim_{\ell \to \infty} \langle h_\ell, g \rangle = \sum_{i=1}^{m} a_i \lim_{\ell \to \infty} h_\ell(x_i) = 0. \qquad \square$$

COROLLARY 3.2. *Let X be a finite set of distinct points of $\mathcal{R}^n$, and let $\mathcal{X} = \left\{ f \in \mathcal{H} : f = \sum_{x \in X} \lambda_x K(\cdot, x), \text{ where } \lambda_x \in \mathcal{R} \right\}$ be the space "carried" by X. Then*

$$\mathcal{X}^\perp = \{ f \in \mathcal{H} : f(x) = 0 \text{ for all } x \in X \}.$$

Furthermore, let P denote the operator defined by interpolation from $\mathcal{X}$ at the nodes in X. Then P is the orthogonal projection from $\mathcal{H}$ onto $\mathcal{X}$.

Proof. That functions $f \in \mathcal{H}$ zero everywhere on X are in $\mathcal{X}^\perp$ follows directly from Lemma 3.1. In the other direction if f is nonzero at a point $y \in X$, then consider the function $g := K(\cdot, y) \in \mathcal{X}$. By an argument similar to that of the lemma we find $\langle f, g \rangle = f(y) \neq 0$. Hence $f \notin \mathcal{X}^\perp$. This establishes the first claim of the corollary.

Now turn to the second claim. From the interpolation conditions, $f - Pf$ is zero at every point of $x \in X$. Hence, by the lemma $f - Pf$ is orthogonal to $\mathcal{X}$. The second claim now follows from the characterization of the orthogonal projection of $f \in \mathcal{H}$ onto a closed subspace $\mathcal{G}$ of $\mathcal{H}$, as the unique $g \in \mathcal{G}$ such that the error $f - g$ is orthogonal to $\mathcal{G}$. $\square$

It is now possible to understand the domain decomposition algorithm as a version of the alternating projection algorithm of von Neumann [13, 14] or the Kacmarz procedure [6]. Each operator P_i in the domain decomposition algorithm is, in fact, the orthogonal projection from $\mathcal{H}$ onto the subspace

$$\mathcal{X}_i = \left\{ f \in \mathcal{H} : f = \sum_{x \in X_i} \lambda_x K(\cdot, x), \text{ where } \lambda_x \in \mathcal{R} \right\}.$$

If we define $Q_i = I - P_i$, then Q_i is the orthogonal projection onto $\mathcal{X}_i^\perp$, and ℓ complete cycles of the domain decomposition algorithm computes $(Q_k \cdots Q_1)^\ell f$.

The convergence of the alternating algorithm is well understood in terms of the angles between these subspaces $\mathcal{X}_i^\perp$.

DEFINITION 3. *Let $\mathcal{U}_1$ and $\mathcal{U}_2$ be closed subspaces of a Hilbert space $\mathcal{H}$ with $\mathcal{U} = \mathcal{U}_1 \cap \mathcal{U}_2$. Then the angle α between $\mathcal{U}_1$ and $\mathcal{U}_2$ is given by*

$$\cos \alpha = \sup \left\{ \langle u, v \rangle : u \in \mathcal{U}_1 \cap \mathcal{U}^\perp, v \in \mathcal{U}_2 \cap \mathcal{U}^\perp \text{ and } \|u\|, \|v\| \leq 1 \right\}.$$

The following theorem of Smith, Solmon, and Wagner [25] then shows that the alternating algorithm usually converges linearly.[4]

[4]It is well known that the rate estimate of Theorem 3.3 is often pessimistic. However, our interest at this point is to guarantee linear convergence, rather than to obtain the best rate. Rate estimates for the alternating algorithm which are often improvements on the estimate of Theorem 3.3 can be found in [5, 7].

THEOREM 3.3. *Let $Q_1, \ldots, Q_k$ be orthogonal projections onto closed subspaces $\mathcal{U}_1, \ldots, \mathcal{U}_k$ in a Hilbert space $\mathcal{H}$. Let $\mathcal{U} = \cap_{i=1}^k \mathcal{U}_i$. Let $Q : \mathcal{H} \to \mathcal{U}$ be the orthogonal projection onto $\mathcal{U}$, and let α_j be the angle between $\mathcal{U}_j$ and $\mathcal{A}_{j+1} = \cap_{i=j+1}^k \mathcal{U}_j$. Then for any $f \in \mathcal{H}$,*

$$\left\| (Q_k \cdots Q_1)^\ell f - Qf \right\|^2 \leq c^{2\ell} \left\| f - Qf \right\|^2,$$

where

$$c^2 \leq 1 - \Pi_{j=1}^{k-1} \sin^2 \alpha_j.$$

Of course it is possible that the angle between some pair $\mathcal{U}_j$ and A_{j+1} may be zero so that $c = 1$, thus denying us the chance of verifying convergence. However, we will soon see that for our application some very weak conditions on the intersections of the node sets are sufficient to rule out this possibility.

DEFINITION 4. *Let $X_1, \ldots, X_k$ be nonempty subsets of $\mathcal{R}^n$ and let $Y_j = \cup_{i=j}^k X_i$, $j = 1, \ldots k$. The sets $X_1, \ldots, X_k$ will be called* weakly distinct *if for each $j = 1, \ldots, k-1$, $X_j \neq Y_j$ and $Y_{j+1} \neq Y_j$.*

Note that if each X_j has a point which belongs only to it, and to no other X_i, then the sets $X_1, \ldots, X_k$ are weakly distinct.

LEMMA 3.4. *Let $X_1, \ldots, X_k$ be finite subsets of $\mathcal{R}^n$, and let $Y_j = \cup_{i=j}^k X_i$, $j = 1, \ldots, k$. Then,*

(i) *$\mathcal{Y}_j^\perp = \cap_{i=j}^k \mathcal{X}_i^\perp$ for $j = 1, \ldots, k$.*

(ii) *If the sets $X_1, \ldots, X_k$ are weakly distinct, then the subspaces $\mathcal{X}_j^\perp \cap \mathcal{Y}_j$ and $\mathcal{Y}_{j+1}^\perp \cap \mathcal{Y}_j$ contain nonzero functions for $j = 1, \ldots, k-1$.*

Proof. First,

$$\begin{aligned}
\bigcap_{i=j}^k \mathcal{X}_i^\perp &= \bigcap_{i=j}^k \{f \in \mathcal{H} : f(x) = 0 \text{ for all } x \in X_i\} \\
&= \left\{ f \in \mathcal{H} : f(x) = 0 \text{ for all } x \in \bigcup_{i=j}^k X_i \right\} \\
&= \left\{ f \in \mathcal{H} : f = \sum_{x \in Y_j} \theta_x K(\cdot, x) : \theta_x \in \mathcal{R} \right\}^\perp \\
&= \mathcal{Y}_j^\perp.
\end{aligned}$$

This shows the first claim of the lemma.

Since $X_1, \ldots, X_k$ are weakly distinct, there is a point $z \in Y_j$ such that $z \notin X_j$. Now consider an element $u \in \mathcal{Y}_j$ of the form

$$u = \sum_{x \in X_j} \theta_x K(\cdot, x) + K(\cdot, z).$$

We claim there is a choice of the θ_x's such that $u \in \mathcal{X}_j^\perp$. In order that $u \in \mathcal{X}_j^\perp$, we have to satisfy the equations

$$\sum_{x \in X_j} \theta_x K(a, x) = -K(a, z) \text{ for all } a \in X_j.$$

Since the matrix $\big(K(a,x)\big)_{a,x\in X_j}$ is invertible, this set of equations has a unique solution for $\{\theta_x : x \in X_j\}$. The corresponding u is nontrivial since the coefficient of $K(\cdot,z)$ in its expansion is 1. The proof that $\mathcal{Y}_{j+1}^{\perp} \cap \mathcal{Y}_j$ contains a nonzero function is similar and will be omitted. ∎

THEOREM 3.5. *Let $f \in \mathcal{H}$ and let $X_1, \ldots, X_k$ be weakly distinct finite subsets of $\mathcal{R}^n$. Set $Y_j = \cup_{i=j}^{k} X_i$, $j = 1, \ldots, k$. Let $Q : \mathcal{H} \to \mathcal{Y}_1^{\perp}$ be the orthogonal projection onto $\mathcal{Y}_1^{\perp}$. Let f_ℓ be the element of $\mathcal{H}$ obtained by applying ℓ complete cycles of the domain decomposition algorithm to f on $X_1, \ldots, X_k$. Then there exists a number $0 \le c < 1$ such that*

$$\|f_\ell - Qf\|^2 \le c^{2\ell}\|f - Qf\|^2 \le c^{2\ell}\|f\|^2, \qquad \ell = 1, 2, \ldots .$$

Proof. We know from our earlier discussion that $f_\ell = (Q_k \cdots Q_1)^\ell f$, where $Q_i : \mathcal{H} \to \mathcal{X}_i^{\perp}$ is the orthogonal projection. An application of Theorem 3.3 shows that

$$\left\|(Q_k \cdots Q_1)^\ell f - Qf\right\|^2 \le c^{2\ell}\|f - Qf\|^2, \qquad \ell = 1, 2, \ldots ,$$

where c may be estimated in terms of the angles between subspaces.

It remains to show that $c < 1$. In the notation of Theorem 3.3, this involves examining α_j, the angle between the subspaces $\mathcal{U}_j = \mathcal{X}_j^{\perp}$ and $\mathcal{A}_{j+1} = \cap_{i=j+1}^{k} \mathcal{X}_i^{\perp}$. A critical ingredient of this analysis is the orthogonal complement of the intersection of these two spaces, namely the subspace

$$\mathcal{B}_j = \left(\mathcal{X}_j^{\perp} \bigcap \left(\bigcap_{i=j+1}^{k} \mathcal{X}_i^{\perp}\right)\right)^{\perp} = \left(\bigcap_{i=j}^{k} \mathcal{X}_i^{\perp}\right)^{\perp}.$$

In Lemma 3.4 it was shown that $\mathcal{Y}_j^{\perp} = \big(\cap_{i=j}^{k} \mathcal{X}_i^{\perp}\big)$ for $j = 1, \ldots, k$. It follows that $\mathcal{A}_{j+1} = \mathcal{Y}_{j+1}^{\perp}$ and $\mathcal{B}_j = \mathcal{Y}_j$. Lemma 3.4 also guarantees that the subspaces $\mathcal{X}_j^{\perp} \cap \mathcal{Y}_j$ and $\mathcal{Y}_{j+1}^{\perp} \cap \mathcal{Y}_j$ contain nonzero elements. Now suppose that $\cos\alpha_j = 1$. That is,

$$\sup\big\{\langle u, v\rangle : u \in \mathcal{X}_j^{\perp} \cap \mathcal{Y}_j, v \in \mathcal{Y}_{j+1}^{\perp} \cap \mathcal{Y}_j, \|u\|, \|v\| \le 1\big\} = 1.$$

For the purposes of the convergence of the algorithm, we can consider ourselves to be working in the finite-dimensional space spanned by f and the elements of $\{K(\cdot, y) : y \in Y_1\}$. Hence, by compactness, we can find elements $u^* \in \mathcal{X}_j^{\perp} \cap \mathcal{Y}_j$ and $v^* \in \mathcal{Y}_{j+1}^{\perp} \cap \mathcal{Y}_j$ with $\|u^*\| = \|v^*\| = 1$ such that $\langle u^*, v^*\rangle = 1$. By the condition for equality in the Cauchy–Schwarz inequality, it follows that $u^* = v^*$. Thus, $u^* \in \mathcal{X}_j^{\perp} \cap \mathcal{Y}_{j+1}^{\perp} = \mathcal{Y}_j^{\perp}$. However, u^* is also in $\mathcal{Y}_j$. Hence $u^* = 0$. This contradicts the fact that $\|u^*\| = 1$, and so $\sin\alpha_j \ne 0$ for $j = 1, \ldots, k$. ∎

We now complement the analysis of Theorem 3.5 by showing how to calculate the angles between subspaces employed in the estimate of the constant c and in the alternative estimates of [5, 7].

Recall from the proof of Theorem 3.5 that α_j is the angle between $\mathcal{X}_j^{\perp}$ and $\mathcal{Y}_{j+1}^{\perp}$. The following result can be found in [3].

LEMMA 3.6. *Let $\mathcal{U}$, $\mathcal{V}$ be closed subspaces of a Hilbert space $\mathcal{H}$. Then the angle between $\mathcal{U}$ and $\mathcal{V}$ is the same as the angle between $\mathcal{U}^{\perp}$ and $\mathcal{V}^{\perp}$.*

This allows us to compute the α_j as the angle between

$$\mathcal{X}_j = \left\{ f \in \mathcal{H} : f = \sum_{x \in X_j} \lambda_x K(\cdot, x) \right\}$$

and

$$\mathcal{Y}_{j+1} = \left\{ f \in \mathcal{H} : f = \sum_{y \in Y_{j+1}} \lambda_y K(\cdot, y) \right\}.$$

Relabeling if necessary, we enumerate X_j as $\{x_1, \ldots, x_{m_2}\}$ and Y_{j+1} as $\{x_{m_1+1}, \ldots, x_{m_3}\}$, where $1 \le m_1 < m_2 \le m_3$. This enumeration is made so that $X_j \cap Y_{j+1} = \{x_{m_1+1}, \ldots, x_{m_2}\}$. We now work entirely in the finite-dimensional Hilbert space

$$\mathcal{H} = \left\{ \sum_{j=1}^{m_3} \lambda_j K(\cdot, x_j) : \lambda_j \in \mathcal{R}, j = 1, \ldots, m_3 \right\}.$$

We adopt the ";" notation for stacking rows of partitioned vectors. Then any vector $\boldsymbol{\lambda} \in \mathcal{R}^{m_3}$ of coefficients can be partitioned into $\boldsymbol{\lambda} = (\boldsymbol{\lambda}_1; \boldsymbol{\lambda}_2; \boldsymbol{\lambda}_3)$, where $\boldsymbol{\lambda}_1 \in \mathcal{R}^{m_1}$, $\boldsymbol{\lambda}_2 \in \mathcal{R}^{m_2-m_1}$, and $\boldsymbol{\lambda}_3 \in \mathcal{R}^{m_3-m_2}$. The corresponding partitioning of the matrix $\mathsf{K} = \left(K(x_i, x_j)\right)_{i,j=1}^{m_3}$ is

$$\begin{array}{c} \begin{array}{ccc} m_1 & m_2 - m_1 & m_3 - m_2 \end{array} \\ \begin{pmatrix} \mathsf{D}_1 & \mathsf{E}_1 & \mathsf{F}_1 \\ \mathsf{C}_2 & \mathsf{D}_2 & \mathsf{E}_2 \\ \mathsf{B}_3 & \mathsf{C}_3 & \mathsf{D}_3 \end{pmatrix} \begin{array}{c} m_1 \\ m_2 - m_1 \\ m_3 - m_2 \end{array} \end{array}.$$

Also the inner product defined in (3.1) of two functions u, v in $\mathcal{H}$ is $\mu^T \mathsf{K} \nu$, where μ and ν are the coefficient vectors of u and v with respect to the reproducing kernel basis.

Now let $u \in \mathcal{X}_j \cap (\mathcal{X}_j \cap \mathcal{Y}_{j+1})^\perp$. Then u has a coefficient vector of the form $(\boldsymbol{\mu}_1; \boldsymbol{\mu}_2; \mathbf{0})$ because $u \in \mathcal{X}_j$. However, since $u \in (\mathcal{X}_j \cap \mathcal{Y}_{j+1})^\perp$ we have $u(x_i) = 0$, $i = m_1 + 1, \ldots, m_2$. Hence, u satisfies interpolation equations of the form $\mathsf{K}(\boldsymbol{\mu}_1; \boldsymbol{\mu}_2; \mathbf{0}) = (\boldsymbol{a}; \mathbf{0}; \boldsymbol{b})$, which imply

$$\mathsf{C}_2 \boldsymbol{\mu}_1 + \mathsf{D}_2 \boldsymbol{\mu}_2 = \mathbf{0}.$$

Since D_2 is invertible, $\boldsymbol{\mu}_2 = -\mathsf{D}_2^{-1} \mathsf{C}_2 \boldsymbol{\mu}_1$. Thus any element $u \in \mathcal{X}_j \cap (\mathcal{X}_j \cap \mathcal{Y}_{j+1})^\perp$ has coefficient vector $(\boldsymbol{\mu}_1; \boldsymbol{\mu}_2; \mathbf{0})$, where

$$\begin{pmatrix} \boldsymbol{\mu}_1 \\ \boldsymbol{\mu}_2 \end{pmatrix} = \begin{pmatrix} \mathsf{I} \\ -\mathsf{D}_2^{-1} \mathsf{C}_2 \end{pmatrix} \boldsymbol{\mu}_1,$$

$\boldsymbol{\mu}_1 \in \mathcal{R}^{m_1}$, $\boldsymbol{\nu}_2 \in \mathcal{R}^{m_2-m_1}$, and I is the $m_1 \times m_1$ identity. Note in particular that the matrix on the right of the equation above has full rank.

In our computation of the angle between $\mathcal{X}_j$ and $\mathcal{Y}_{j+1}$, we need to restrict attention to $u \in \mathcal{X}_j$ with $\|u\| = 1$. Now,

$$\begin{aligned} \|u\|^2 &= \begin{pmatrix} \boldsymbol{\mu}_1^T & \boldsymbol{\mu}_2^T & \mathbf{0}^T \end{pmatrix} \begin{pmatrix} \mathsf{D}_1 & \mathsf{E}_1 & \mathsf{F}_1 \\ \mathsf{C}_2 & \mathsf{D}_2 & \mathsf{E}_2 \\ \mathsf{B}_3 & \mathsf{C}_3 & \mathsf{D}_3 \end{pmatrix} \begin{pmatrix} \boldsymbol{\mu}_1 \\ \boldsymbol{\mu}_2 \\ \mathbf{0} \end{pmatrix} \\ &= \boldsymbol{\mu}_1^T \begin{pmatrix} I & \left(-\mathsf{D}_2^{-1} \mathsf{C}_2\right)^T \end{pmatrix} \begin{pmatrix} \mathsf{D}_1 & \mathsf{E}_1 \\ \mathsf{C}_2 & \mathsf{D}_2 \end{pmatrix} \begin{pmatrix} \mathsf{I} \\ -\mathsf{D}_2^{-1} \mathsf{C}_2 \end{pmatrix} \boldsymbol{\mu}_1 \\ &= \boldsymbol{\mu}_1^T \left(\mathsf{D}_1 - \mathsf{E}_1 \mathsf{D}_2^{-1} \mathsf{C}_2\right) \boldsymbol{\mu}_1 \\ &= \boldsymbol{\mu}_1^T \left(\mathsf{D}_1 - \mathsf{C}_2^T \mathsf{D}_2^{-1} \mathsf{C}_2\right) \boldsymbol{\mu}_1 \, . \end{aligned}$$

The matrix $\left(\begin{smallmatrix}\mathsf{D}_1 & \mathsf{E}_1\\ \mathsf{C}_2 & \mathsf{D}_2\end{smallmatrix}\right)$ is the interpolation matrix for the set X_j and so is strictly positive definite and symmetric. Since $\begin{pmatrix}\mathsf{I} & (-\mathsf{D}_2^{-1}\mathsf{C}_2)^T\end{pmatrix}$ is of full rank we can write $\|u\|^2 = \boldsymbol{\mu}_1^T\mathsf{G}_1\boldsymbol{\mu}_1$, where G_1 is strictly positive definite and symmetric. Let the Cholesky decomposition of G_1 be $\mathsf{L}_1\mathsf{L}_1^T$. Then $\|u\|^2 = \boldsymbol{\mu}_1^T\mathsf{L}_1\mathsf{L}_1^T\boldsymbol{\mu}_1$. Letting $\boldsymbol{\gamma}_1 = \mathsf{L}_1^T\boldsymbol{\mu}_1$ gives $\|u\|^2 = \boldsymbol{\gamma}_1^T\boldsymbol{\gamma}_1$.

Similar arguments show that if $v \in \mathcal{Y}_{j+1}\cap(\mathcal{X}_j \cap \mathcal{Y}_{j+1})^\perp$, then v has coefficient vector $(\mathbf{0};\boldsymbol{\nu}_2;\boldsymbol{\nu}_3)$. Furthermore, $\|v\|^2$ can be realized via the expression $\|v\|^2 = \boldsymbol{\gamma}_3^T\boldsymbol{\gamma}_3 = \boldsymbol{\nu}_3\mathsf{L}_3\mathsf{L}_3^T\boldsymbol{\nu}_3$, where $\mathsf{L}_3\mathsf{L}_3^T$ is the Cholesky decomposition of the matrix

$$\begin{pmatrix}\left(-\mathsf{D}_2^{-1}\mathsf{E}_2\right)^T & \mathsf{I}\end{pmatrix}\begin{pmatrix}\mathsf{D}_2 & \mathsf{E}_2\\ \mathsf{C}_3 & \mathsf{D}_3\end{pmatrix}\begin{pmatrix}-\mathsf{D}_2^{-1}\mathsf{E}_2\\ \mathsf{I}\end{pmatrix} = \mathsf{D}_3 - \mathsf{C}_3\mathsf{D}_2^{-1}\mathsf{C}_3^T.$$

Thus the calculation of the cosine of the angle between $\mathcal{X}_j$ and $\mathcal{Y}_{j+1}$ involves finding the supremum over vectors $\boldsymbol{\gamma}_1 \in \mathcal{R}^{m_1}$ and $\boldsymbol{\gamma}_3 \in \mathcal{R}^{m_3-m_2}$ with $\|\boldsymbol{\gamma}_1\|_2 = \|\boldsymbol{\gamma}_3\|_2 = 1$ of the expression

$$\begin{aligned}\mu^T\mathsf{K}\nu &= \boldsymbol{\mu}_1^T\begin{pmatrix}\mathsf{I} & \left(-\mathsf{D}_2^{-1}\mathsf{C}_2\right)^T & \mathsf{O}\end{pmatrix}\begin{pmatrix}\mathsf{D}_1 & \mathsf{E}_1 & \mathsf{F}_1\\ \mathsf{C}_2 & \mathsf{D}_2 & \mathsf{E}_2\\ \mathsf{B}_3 & \mathsf{C}_3 & \mathsf{D}_3\end{pmatrix}\begin{pmatrix}\mathsf{O}\\ -\mathsf{D}_2^{-1}\mathsf{E}_2\\ \mathsf{I}\end{pmatrix}\boldsymbol{\nu}_3\\ &= \boldsymbol{\mu}_1^T\begin{pmatrix}\mathsf{I} & \left(-\mathsf{D}_2^{-1}\mathsf{C}_2\right)^T & \mathsf{O}\end{pmatrix}\begin{pmatrix}-\mathsf{E}_1\mathsf{D}_2^{-1}\mathsf{E}_2+\mathsf{F}_1\\ \mathsf{O}\\ -\mathsf{C}_3\mathsf{D}_2^{-1}\mathsf{E}_2+\mathsf{D}_3\end{pmatrix}\boldsymbol{\nu}_3\\ &= \boldsymbol{\mu}_1^T(\mathsf{F}_1 - \mathsf{E}_1\mathsf{D}_2^{-1}\mathsf{E}_2)\boldsymbol{\nu}_3\\ &= \boldsymbol{\gamma}_1^T\mathsf{L}_1^{-1}(\mathsf{F}_1 - \mathsf{E}_1\mathsf{D}_2^{-1}\mathsf{E}_2)\left(\mathsf{L}_3^T\right)^{-1}\boldsymbol{\gamma}_3.\end{aligned}$$

We have therefore established the following result.

THEOREM 3.7. *Adopt the notation prior to the theorem. Then the cosine of the angle between* $\mathcal{X}_j^\perp$ *and* $\cap_{i=j+1}^k\mathcal{X}_i^\perp$ *is given by the* ℓ_2 *norm of the matrix* $\mathsf{L}_1^{-1}(\mathsf{F}_1 - \mathsf{E}_1\mathsf{D}_2^{-1}\mathsf{E}_2)\left(\mathsf{L}_3^T\right)^{-1}$.

An easy adjustment needs to be made to the above theorem if there is no overlap between the sets X_j and Y_{j+1}. In this case, we write the block matrix as

$$\mathsf{K} = \begin{pmatrix}\mathsf{D}_1 & \mathsf{F}_1\\ \mathsf{B}_3 & \mathsf{D}_3\end{pmatrix},$$

and find that the angle between the subspaces $\mathcal{X}_j$ and $\mathcal{Y}_{j+1}$ is the ℓ_2 norm of the matrix $\mathsf{L}_1^{-1}\mathsf{F}_1(\mathsf{L}_3^T)^{-1}$, where $\mathsf{L}_1\mathsf{L}_1^T = \mathsf{D}_1$ and $\mathsf{L}_3\mathsf{L}_3^T = \mathsf{D}_3$.

4. Numerical results of a domain decomposition code. In this section we will briefly discuss some aspects of the implementation and performance of a domain decomposition interpolation code employing some of the previous mathematics. As in section 3, it will be convenient to denote a finite set of distinct points by X and the space of functions "carried" by that set by $\mathcal{X}$.

We list below what we see as the essential ingredients of a domain decomposition code for fitting a globally supported radial basis function by interpolation.

Essential ingredients for a domain decomposition interpolatory fitter.

(i) A method for subdividing space.

(ii) An efficient and scale independent method for solving small interpolation subproblems. The solutions to the small problems will be used to precondition, or approximately solve, the large problem.

(iii) A fast method for computing the action of the large interpolation matrices that occur at various scales.
(iv) An outer iteration.

The method for subdividing space of item (i) above will be used to form the subdomains. Currently we are using a point ordering related to a balanced nD-tree [17] to subdivide space into rectangular boxes. Given that the solution of the large system is to be built upon the solution of many small systems, the need for item (ii) above is clear. This need is filled by, and indeed partly motivated, section 2 of this paper. All matrix iterative methods of which we are aware require at least one matrix-vector product per iteration. Therefore, a prerequisite for the overall method to achieve convergence in $\mathcal{O}(N)$, or $\mathcal{O}(N \log N)$, operations is that a single matrix-vector product costs at most $\mathcal{O}(N)$ or $\mathcal{O}(N \log N)$ operations. Thus, for large interpolation problems with globally supported radial basis functions, item (iii) in the list above, a fast way of computing the action of the interpolation matrix on a vector is absolutely essential. Fortunately, such fast evaluators are becoming available for more and more functions Φ. Item (iv) of the list consists of any suitable means of updating the current approximation to the interpolant using the current residuals and quantities derived from the residuals by interpolation on subdomains. It may be as simple as incrementing the current approximation to the interpolant by an approximate interpolant to the current residual.

A simplified two-level code built upon the parts specified in the ingredients list is given below. The method described is a variant of additive Schwarz as opposed to the multiplicative Schwarz method analyzed in section 3.

A simplified domain decomposition interpolation code.

INPUT

Input the finite node set X, the right-hand side f, and the desired accuracy ϵ.

SETUP

(1) Subdivide space (that is, X) into overlapping subdomains $\{X_j : j = 1, \ldots, m\}$. For each subdomain classify some points as inner points and some as outer, such that the union of all the inner points is the whole node set.
(2) Choose a coarse grid $Y^{(2)}$ containing some points from each inner subdomain.
(3) Form and factor the matrices required to solve the radial basis function (RBF) interpolation problems on the subdomains. Throughout, r_g, s_g, λ_g and c_g will denote the current residual, the current approximation to the interpolant and parameters of the current approximation to the interpolant at the finest, or global, level.
(5) $r_g \leftarrow f$, $s_g \leftarrow 0$, $\lambda_g \leftarrow 0$, and $c_g \leftarrow 0$.

ITERATIVE SOLUTION

(1) **while** $\|r_g\| > \epsilon$
(2) $\lambda \leftarrow 0$. λ is the coefficient vector of the fine level correction.

(3) **for** $j = 1$ **to** m
(4) Set the coefficients λ_i corresponding to inner points of X_j to the coefficients of the interpolant from the spline space $\mathcal{X}_j$ to $r_g|_{X_j}$.
(5) **end for**
(6) Correct λ to be orthogonal to π_{k-1}.
(7) $s_1 \leftarrow \sum \lambda_j \Phi(\cdot, x_j)$.
(8) Evaluate the residual $r_1 \leftarrow r_g - s_1$ at the coarse grid points.
(9) Interpolate to r_1 at the coarse grid points $Y^{(2)}$ using a spline s_2 (including the polynomial part) from $\mathcal{Y}^{(2)}$.
(10) $s_g \leftarrow s_g + s_1 + s_2$.
(11) Reevaluate the global residuals $r_g \leftarrow f - s_g$ at all the points of X.
(12) **end while**

Clearly there are many choices here. For example, one can use a multiplicative Schwarz (block Gauss–Seidel) like update, rather than the additive Schwarz (block Jacobi) like update of the coefficient vector λ. The analysis of section 3 guarantees that such a multiplicative Schwarz updated version will converge. Note for the analysis to apply one must at step 4 of the iterative solution update all the λ_i's corresponding to $\mathcal{X}_j$, not just those for inner points. As is well known [24], an advantage of additive Schwarz is that it is easier to parallelize as the changes to blocks of coefficient vectors can be made in parallel. Furthermore, in our application, the evaluation of the residuals, typically performed with a variant of the fast multipole method, is also easily parallelized.

Other choices occur in the evaluation of the residuals. There is usually more than one applicable fast evaluation algorithm, and even when the overall fast evaluation algorithm is fixed, there are many detailed implementation decisions to make. Another possibility is to view the forming of the correction function $s_1 + s_2$ as the action of a preconditioning matrix B on the current residual vector r_g. It is then natural to employ some matrix iterative method, for example GMRES, in the outer iteration. Furthermore, for really large data sets one clearly employs more than two levels.

A single pass through the main while loop of the iterative process sketched above can be viewed as a linear process on the input residuals. Therefore, one can represent the process as a mapping $r_g \to \mathsf{R} r_g$. The convergence of the iteration will then be characterized by the spectral radius of this *residual matrix* R. Numerical experiments show that if coarse grid correction is absent, then even with subdomain overlap, the spectral radius of R can be large. Further, these experiments show that the eigenvectors corresponding to the largest magnitude eigenvalue typically represent a low frequency oscillation. Correction by interpolation at coarse grid points can be expected to damp out such low frequency oscillations. Experiments involving calculation of the spectral radius of R, with coarse grid correction, show that this expectation is realized. Specifically calculations of the spectral radius of R have been performed for small model problems and the thin-plate spline in two dimensions. The results are tabulated in Table 3. They show that, as expected, the spectral radius of R is generally small and becomes smaller as the amount of overlap, or the size of the correction grid, is increased.

TABLE 3

Spectral radius of the residual matrix for four subdomains and various values of overlap and correction size. The grids are all uniform subdivisions of the unit square. The interpolant considered is the thin-plate spline.

Correction grid	Fine grid	Number of overlap rows/columns		
		2	3	4
3×3	10×10	1.391(-1)	9.152(-2)	5.290(-2)
	20×20	1.769(-1)	1.463(-1)	1.355(-1)
	40×40	3.338(-1)	1.823(-1)	1.755(-1)
4×4	10×10	5.606(-2)	4.186(-2)	3.374(-2)
	20×20	2.479(-1)	5.758(-2)	5.145(-2)
	40×40	6.975(-1)	1.556(-1)	6.404(-2)

TABLE 4

Performance of the current C domain decomposition implementation. The table gives times in seconds to fit the Franke 1 *function until the residual is of magnitude less than* 1.0×10^{-6}. *All computations were carried out using doubles (*64 *bit reals). The node sets were randomly distributed in* $[0,1]^2$ *and the timings performed on a generic machine with an Intel Celeron processor running at* 450 *MHz.*

Number of nodes	Number of level 1 iterations	Time taken in seconds
10,000	8	7.0
20,000	8	17.5
40,000	6	35.5
80,000	6	105.7
160,000	7	407.8

The current C implementation is specialized to fitting 2- and 3-dimensional polyharmonic splines. This was due mostly to the suitability of these splines for some large geophysical and image processing problems motivating our work. A subsidiary reason was that we had in place fast evaluation codes for these functions. However, the ideas and the mathematics of the previous sections apply in the much more general setting of interpolants based upon sums of translates of any strictly conditionally positive definite function Φ. Furthermore, fast evaluation schemes are now being developed for a wide variety of basic functions. Thus we expect the domain decomposition approach to have wide applicability. The existing code has been successfully used to fit data sets of up to 5 million points in 2 dimensions, and up to 250,000 points in 3 dimensions. It is being steadily improved in sophistication, speed, and memory footprint.

The $\mathcal{O}(N^2)$ storage requirements, and $\mathcal{O}(N^3)$ operation counts of direct methods, imply that direct solution of problems with greater than 10,000 centers will be very time consuming, even on very well-endowed machines. Indeed, as late as 1992, many authors have commented on the impracticality of direct, or even iterative, solution of such large RBF interpolation problems. See the quotes in [1]. Bearing this history in mind, the timings in Table 4 are highly satisfactory. They are for interpolation to the

Franke 1 function

$$F^{(1)}(\xi,\eta) = \frac{3}{4}\exp\left(-\frac{(9\xi-2)^2+(9\eta-2)^2}{4}\right) + \frac{3}{4}\exp\left(-\frac{(9\xi+1)^2}{49} - \frac{9\eta+1}{10}\right)$$
$$+ \frac{1}{2}\exp\left(-\frac{(9\xi-7)^2+(9\eta-3)^2}{4}\right) - \frac{1}{5}\exp\left(-(9\xi-4)^2-(9\eta-7)^2\right).$$

To the best of our knowledge these timings are superior at the time of writing to those achieved by any competing RBF fitting code. The closest competitors in terms of performance on large problems are the preconditioned GMRES iterative methods discussed in [1]. These are slower but much easier to implement. Numerical experience with the domain decomposition code seems to indicate an approximately $\mathcal{O}(N \log N)$ complexity.

Acknowledgment. We are grateful to Tim Mitchell of the University of Canterbury for his careful efforts in coding various versions of the method.

REFERENCES

[1] R.K. Beatson, J.B. Cherrie, and C.T. Mouat, *Fast fitting of radial basis functions: Methods based on preconditioned GMRES iteration*, Adv. Comput. Math., 11 (1999), pp. 253–270.
[2] T.F. Chan and T.P. Mathew, *Domain decomposition algorithms*, in Acta Numer. 1994, Cambridge University Press, 1994, pp. 61–143.
[3] F. Deutsch and H. Hundal, *The rate of convergence of Dykstra's cyclic projections algorithm: The polyhedral case*, Numer. Funct. Anal. Optim., 15 (1994), pp. 537–565.
[4] A.C. Faul and M.J.D. Powell, *Proof of the Convergence of an Iterative Technique for Thin Plate Spline Interpolation in Two Dimensions*, Numerical Analysis Technical Report DAMTP 1998/NA08, Department of Applied Mathematics and Theoretical Physics, University of Cambridge, Cambridge, UK, 1998.
[5] C. Hamaker and D.C. Solmon, *The angles between the null spaces of X-rays*, J. Math. Anal. Appl., 62 (1978), pp. 1–23.
[6] S. Kacmarz, *Angenaherte auflosung von systemen linearer gleichungen*, Bull. Acad. Polon. Sci. Lett., A (1937), pp. 355–357.
[7] S. Kayalar and H.L. Weinert, *Error bounds for the method of alternating projections*, Math. Control Signal Systems, 1 (1988), pp. 43–59.
[8] W.A. Light and H.S.J. Wayne, *Spaces of distributions and conditionally positive definite functions*, Numer. Math., 81 (1999), pp. 415–450.
[9] W. Madych and S. Nelson, *Multivariate interpolation and conditionally positive definite functions*, Approx. Theory Appl., 4 (1988), pp. 77–89.
[10] W. Madych, *Miscellaneous error bounds for multiquadric and related interpolators*, Comput. Math. Appl., 24 (1992), pp. 121–138.
[11] C.A. Micchelli, *Interpolation of scattered data: Distance matrices and conditionally positive definite functions*, Constr. Approx., 2 (1986), pp. 11–22.
[12] F. Narcowich and J.D. Ward, *Norms of inverses and condition numbers for matrices associated with scattered data*, J. Approx. Theory, 64 (1991), pp. 69–94.
[13] J. von Neumann, *On rings of operators. Reduction theory*, Ann. of Math. (2), 50 (1949), pp. 401–485.
[14] J. von Neumann, *Functional Operators, Vol.* II*: The Geometry of Orthogonal Spaces*, Princeton University Press, Princeton, NJ, 1950.
[15] G.N. Newsam, *private communication*, Adelaide, Australia, 1996.
[16] M.J.D. Powell, *Some algorithms for thin plate spline interpolation to functions of two variables*, in Adv. Comput. Math., H.P. Dikshit and C.A. Micchelli, eds., World Scientific Publishing Co., New Delhi, India, 1993, pp. 303–319.
[17] H. Samet, *The Design and Analysis of Spatial Data Structures*, Addison–Wesley, New York, 1989.
[18] R. Schaback, *Comparison of radial basis function interpolants*, in Multivariate Approximations: From CAGD to Wavelets, K. Jetter and F. Utreras, eds., World Scientific, Singapore, 1993, pp. 293–305.

[19] R. Schaback, *Multivariate interpolation and approximation by translates of a basis function*, in Approximation Theory VIII, Vol. 1, Approximation and Interpolation, C. K. Chui and L. L. Schumacker, eds., World Scientific, Singapore, 1995, pp. 491–514.

[20] R. Schaback and H. Wendland, *Numerical techniques based on radial basis functions*, in Curves and Surface Fitting: Saint-Malo 1999, A. Cohen, C. Rabut, and L.L. Schumaker, eds., Vanderbilt University Press, Nashville, TN, 2000, pp. 359–374.

[21] I.J. Schoenberg, *Metric spaces and completely monotone functions*, Ann. Math. (2), 39 (1938), pp. 811–841.

[22] H.S. Shapiro, *Topics in Approximation Theory*, Lecture Notes in Math., 187, Springer-Verlag, New York, 1971.

[23] R. Sibson and G. Stone, *Computation of thin-plate splines*, SIAM J. Sci. Statist. Comput., 12 (1991), pp. 1304–1313.

[24] B. Smith, P. Bjorstad, and W. Gropp, *Domain Decomposition: Parallel Multilevel Methods for Partial Differential Equations*, Cambrige University Press, Cambridge, UK, 1996.

[25] K.T. Smith, D.C. Solmon, and S.L. Wagner, *Practical and mathematical aspects of the problem of reconstructing objects from radiographs*, Bull Amer. Math. Soc., 83 (1977), pp. 1227–1270.

[26] H.J. Wenz, *Projection methods for solving interpolation problems with radial basis functions*, presented to Curves and Surfaces, Saint-Malo, France, 1999.

SIAM J. SCI. COMPUT.
Vol. 22, No. 5, pp. 1741–1761
© 2001 Society for Industrial and Applied Mathematics

PRECONDITIONERS FOR ILL-CONDITIONED TOEPLITZ SYSTEMS CONSTRUCTED FROM POSITIVE KERNELS*

DANIEL POTTS† AND GABRIELE STEIDL‡

Abstract. In this paper, we are interested in the iterative solution of ill-conditioned Toeplitz systems generated by continuous nonnegative real-valued functions f with a finite number of zeros. We construct new w-circulant preconditioners without explicit knowledge of the generating function f by approximating f by its convolution $f * K_N$ with a suitable positive reproducing kernel K_N. By the restriction to positive kernels we obtain positive definite preconditioners. Moreover, if f has only zeros of even order $\leq 2s$, then we can prove that the property $\int_{-\pi}^{\pi} t^{2k} K_N(t)\,\mathrm{d}t \leq CN^{-2k}$ $(k = 0, \ldots, s)$ of the kernel is necessary and sufficient to ensure the convergence of the PCG method in a number of iteration steps independent of the dimension N of the system. Our theoretical results were confirmed by numerical tests.

Key words. ill-conditioned Toeplitz matrices , CG method, preconditioners, reproducing kernels

AMS subject classifications. 65F10, 65F15, 65T10

PII. S1064827599351428

1. Introduction.

In this paper, we are concerned with the iterative solution of sequences of "mildly" ill-conditioned Toeplitz systems

$$\boldsymbol{A}_N \boldsymbol{x}_N = \boldsymbol{b}_N\,,$$

where $\boldsymbol{A}_N \in \mathbb{C}^{N,N}$ are positive definite Hermitian Toeplitz matrices generated by a continuous nonnegative function f which has only a finite number of zeros. Often these systems are obtained by discretization of continuous problems (partial differential equation, integral equation with weakly singular kernel) and the dimension N is related to the grid parameter of the discretization. For further applications, see [12] and the references therein.

Iterative solution methods for Toeplitz systems, in particular the conjugate gradient method (CG method), have attained much attention during the last years. The reason for this is that the essential computational effort per iteration step, namely, the multiplication of a vector with the Toeplitz matrix $\boldsymbol{A}_N$, can be reduced to $\mathcal{O}(N \log N)$ arithmetical operations by fast Fourier transforms (FFTs). However, the number of iteration steps depends on the distribution of the eigenvalues of $\boldsymbol{A}_N$. If we allow the generating function f to have isolated zeros, then the condition numbers of the related Toeplitz matrices grow polynomial with N and the CG method converges very slowly [8, 28, 45]. Therefore, the real task consists in the construction of suitable preconditioners $\boldsymbol{M}_N$ of $\boldsymbol{A}_N$ so that the number of iteration steps of the corresponding preconditioned CG method (PCG method) becomes independent of N. Here it is useful to recall a result of Axelsson [1, p. 573] relating the spectrum of the coefficient matrix to the number of iteration steps to achieve a prescribed precision.

*Received by the editors February 5, 1999; accepted for publication (in revised form) July 31, 2000; published electronically January 5, 2001.
http://www.siam.org/journals/sisc/22-5/35142.html
†Medical University of Lübeck, Institut of Mathematics, Wallstr. 40, D–23560 Lübeck, Germany (potts@math.mu-luebeck.de).
‡University of Mannheim, Institut of Mathematics, D–68131 Mannheim, Germany (steidl@math.uni-mannheim.de).

THEOREM 1.1. *Let $\boldsymbol{A}$ be a positive definite Hermitian (N,N)-matrix which has p and q isolated large and small eigenvalues, respectively:*

$$0 < \lambda_1 \le \lambda_2 \le \cdots \le \lambda_q < a \le \lambda_{q+1} \le \cdots \le \lambda_{N-p} \le b < \lambda_{N-p+1} \le \lambda_{N-p+2} \le \cdots \le \lambda_N \quad (0 < a < b < \infty).$$

Let $\lceil x \rceil$ denote the smallest integer $\ge x$. Then the CG method for the solution of $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ requires at most

$$n = \left\lceil \left(\ln \frac{2}{\tau} + \sum_{k=1}^{q} \ln \frac{b}{\lambda_k} \right) \Big/ \ln \frac{1 + (\frac{a}{b})^{1/2}}{1 - (\frac{a}{b})^{1/2}} \right\rceil + p + q$$

iteration steps to achieve precision τ, i.e.,

$$\frac{||\boldsymbol{x}_n - \boldsymbol{x}||_A}{||\boldsymbol{x}_0 - \boldsymbol{x}||_A} \le \tau \ ,$$

where $||\boldsymbol{x}||_A := \sqrt{\bar{\boldsymbol{x}}' \boldsymbol{A}\, \boldsymbol{x}}$ and where $\boldsymbol{x}_n$ denotes the numerical solution after n iteration steps.

In literature two kinds of preconditioners were mainly exploited, namely banded Toeplitz matrices and matrices arising from a matrix algebra $\mathcal{A}_{O_N} := \{\bar{\boldsymbol{O}}'_N(\operatorname{diag} \boldsymbol{d})\boldsymbol{O}_N : \boldsymbol{d} \in \mathbb{C}^N\}$, where $\boldsymbol{O}_N$ denotes a unitary matrix.

For another approach by multigrid methods see, for example, [23].

Various banded Toeplitz preconditioners were examined [10, 5, 40, 36, 41]. It was proved that the corresponding PCG methods converge in a number of iteration steps independent of N. However, there is the significant constraint that the cost per iteration of the proposed procedure should be upper-bounded by $\mathcal{O}(N \log N)$. This implies some conditions on the growth of the bandwidth of the banded Toeplitz preconditioners [41].

The above constraint is trivially fulfilled if we chose preconditioners from matrix algebras, where the unitary matrix $\boldsymbol{O}_N$ has to allow an efficient multiplication with a vector in $\mathcal{O}(N \log N)$ arithmetical operations. Up to now, the only preconditioners of the matrix algebra class which ensure the desired convergence of the corresponding PCG method are the preconditioners proposed in [31, 25]. Unfortunately, the construction of these preconditioners requires the explicit knowledge of the generating function f.

Extensive examinations were done with natural and optimal Tau preconditioners [6, 3]. Only for sufficiently smooth functions, where the necessary smoothness depends on the order of the zeros of f, the natural Tau preconditioners become positive definite and lead to the desired location of the eigenvalues of the preconditioned matrices. The optimal Tau preconditioner is in general a bad choice if f has zeros of order > 2 (cf. [6]).

In this paper, we combine our approach in [31] with the approximation of f by its convolution with a reproducing kernel K_N. The kernel approach was given in [15] for positive generating functions. Interesting tests with B-spline kernels were performed by Chan, Tso, and Sun in [14]. The advantage of the kernel approach is that it does not require the explicit knowledge of the generating function. However, for our theoretical proofs we need some knowledge about the location of the zeros of the generating function f. See the remarks at the end of this section. We restrict our attention to positive kernels. This ensures that our preconditioners are positive

definite. Suppose that f has only zeros of even order $\leq 2s$. Then we prove that under the *moment condition*

$$\int_{-\pi}^{\pi} t^{2k} K_N(t)\,\mathrm{d}t \leq CN^{-2k} \quad (k = 0, \ldots, s)$$

on the kernels K_N, the eigenvalues of $\boldsymbol{M}_N^{-1}\boldsymbol{A}_N$ are contained in some interval $[a, b]$ $(0 < a \leq b < \infty)$ except for a fixed number (independent of N) of eigenvalues falling into $[b, \infty)$ so that PCG converges in $\mathcal{O}(1)$ steps.

Note that the above kernel property with $s = 1$ implies for sufficiently smooth f the Jackson result

$$||f - K_N * f||_\infty \leq N^{-2}\omega(f^{(2)}, 1/N)\,,$$

where ω denotes the modulus of continuity. On the other hand, the classical saturation result of Korovkin [29, 21] states that we cannot expect a convergence speed of $||f - K_N * f||_\infty$ better than N^{-2} even in the presence of very regular functions f.

This paper is organized as follows: In section 2, we introduce our w-circulant positive definite preconditioners. We show how the corresponding PCG method can be implemented with only $\mathcal{O}(N)$ arithmetical operations per step more than the original CG method. Section 3 is concerned with the location of the eigenvalues of the preconditioned matrices. We will see that under some assumptions on the kernel the number of CG iterations is independent of N. Special kernels as Jackson kernels and B-spline kernels are considered in section 5. In section 6, we sketch how our ideas can be extended to (real) symmetric Toeplitz matrices with trigonometric preconditioners and to doubly symmetric block Toeplitz matrices with Toeplitz blocks (BTTB matrix). Finally, section 7 contains numerical results.

After sending our manuscript to the *SIAM Journal of Scientific Computing*, R. H. Chan informed us that his group produced results similar to those in our preprint. See [16], and for a refined version, see [17]. The construction of circulant preconditioners of Chan et al. is only based on Jackson kernels and the proofs are different from ours. By a trick (see [16, Theorem 4.2]), which can also be applied to our w-circulant preconditioners, the authors need no knowledge about the location of the zeros of f.

2. Preconditioners from kernels. Let $C_{2\pi}$ denote the Banach space of 2π-periodic real-valued continuous functions with norm

$$\|f\|_\infty := \max_{x\in[-\pi,\pi]} |f(x)|\,.$$

We are interested in the solution of Hermitian Toeplitz systems

$$\boldsymbol{A}_N\boldsymbol{x} = \boldsymbol{b}, \quad \boldsymbol{A}_N = \boldsymbol{A}_N(f) := (a_{j-k})_{j,k=0}^{N-1}, \tag{2.1}$$

$$a_k = a_k(f) := \frac{1}{2\pi}\int_0^{2\pi} f(x)\,\mathrm{e}^{-\mathrm{i}kx}\,\mathrm{d}x$$

generated by a nonnegative function $f \in C_{2\pi}$ which has only a finite number of zeros. By [10], the matrices $\boldsymbol{A}_N(f)$ are positive definite such that (2.1) can be solved by the CG method. Unfortunately, since the generating function $f \in C_{2\pi}$ has zeros, the related Toeplitz matrices are asymptotically ill-conditioned and the CG method converges very slowly. To accelerate the convergence of the CG method, we are looking

for suitable preconditioners of $\boldsymbol{A}_N$, where we do not suppose the explicit knowledge of the generating function f. To reach our aim, we use reproducing kernels. This method was originally proposed for Toeplitz matrices arising from positive functions $f \in C_{2\pi}$ in [15].

In [14] Chan, Tso, and Sun showed by numerical tests that preconditioners from special kernels related to B-splines can improve the convergence of the CG method also if $f \geq 0$ has zeros of various order. A theoretical proof of R. H. Chan's results was open up to now.

In this paper, we restrict our attention to even trigonometric polynomials

$$K_N(x) := c_{N,0} + 2 \sum_{k=1}^{N-1} c_{N,k} \cos kx \,, \quad c_{N,k} = a_k(K_N) \,. \tag{2.2}$$

If

$$\frac{1}{2\pi} \int_0^{2\pi} K_N(x) dx = c_{N,0} = 1 \tag{2.3}$$

and $K_N \geq 0$, then K_N is called a *positive* (trigonometric) *kernel*. As main examples of such kernels, we consider generalized Jackson polynomials and B-spline kernels in section 4. For $f \in C_{2\pi}$, let f_N denote the *convolution of f with K_N*, i.e.,

$$f_N(x) = (f * K_N)(x) := \frac{1}{2\pi} \int_0^{2\pi} f(t) K_N(x-t) \, \mathrm{d}t, \tag{2.4}$$

or equivalently, in the Fourier domain

$$f_N(x) = \sum_{k=-(N-1)}^{N-1} a_k(f) c_{N,k} \, \mathrm{e}^{\mathrm{i}kx} \,. \tag{2.5}$$

We consider so-called *reproducing kernels* K_N $(N \in \mathbb{N})$ with the property that

$$\lim_{N \to \infty} \|f - f_N\|_\infty = 0 \tag{2.6}$$

for all $f \in C_{2\pi}$.

We chose grids G_N $(N \in \mathbb{N})$ consisting of equispaced nodes

$$x_{N,l} := w_N + \frac{2\pi l}{N} \quad \left(l = 0, \ldots, N-1; \, w_N \in \left[0, \frac{2\pi}{N} \right) \right) \tag{2.7}$$

such that $f(x_{N,l}) \neq 0$ for all $l = 0, \ldots, N-1$. Note that the choice of the grids requires some preliminary information about the location of the zeros of f. By a trick (cf. [16]) this restriction can be neglected if we accept some more outliers. We consider matrices of the form

$$\boldsymbol{M}_N(f) := \boldsymbol{W}_N \boldsymbol{F}_N \boldsymbol{D}_N(f) \bar{\boldsymbol{F}}_N \bar{\boldsymbol{W}}_N \tag{2.8}$$

with

$$\boldsymbol{F}_N := \frac{1}{\sqrt{N}} \left(\mathrm{e}^{-2\pi \mathrm{i} jk/N} \right)_{j,k=0}^{N-1} \,, \quad \boldsymbol{W}_N := \operatorname{diag} \left(\mathrm{e}^{-\mathrm{i}k w_N} \right)_{k=0}^{N-1},$$

$$\boldsymbol{D}_N(f) \, = \, \operatorname{diag} \left(f(x_{N,l}) \right)_{l=0}^{N-1}.$$

Obviously, the matrices $\boldsymbol{M}_N$ can be written as

$$\boldsymbol{M}_N(f) = \begin{pmatrix} \tilde{a}_0 & \tilde{a}_{N-1}\,\mathrm{e}^{\mathrm{i}Nw_N} & \cdots & \tilde{a}_1\,\mathrm{e}^{\mathrm{i}Nw_N} \\ \tilde{a}_1 & \tilde{a}_0 & & \\ \vdots & & \ddots & \vdots \\ \tilde{a}_{N-1} & \cdots & \cdots & \tilde{a}_0 \end{pmatrix}$$

with

$$\tilde{a}_k = \tilde{a}_k(f) := \frac{1}{N}\sum_{l=0}^{N-1} f(x_{N,l})\ \mathrm{e}^{-ikw_N}\ \mathrm{e}^{-2\pi\mathrm{i}kl/N}\,. \tag{2.9}$$

These are $(\mathrm{e}^{\mathrm{i}Nw_N})$-*circulant matrices* (see [20]). In particular, we obtain circulant matrices for $w_N = 0$ and skew-circulant matrices for $w_N = \frac{\pi}{N}$.

As preconditioners for (2.1), we suggest matrices of the form

$$\boldsymbol{M}_N := \boldsymbol{M}_N(f_N) \tag{2.10}$$

with suitable positive reproducing kernels K_N. By (2.5), the construction of these preconditioners requires only the knowledge of the Toeplitz matrices $\boldsymbol{A}_N$. It is not necessary to know the generating function f explicitly. However, for the theoretical results in this paper, we must have some information about the location of the zeros of f. Note that by a trick in [16] this information is also superfluous. Here we point out that the auxiliary nontrivial problem of finding some crucial analytic properties of the generating function f has been treated and partially solved in [40].

Moreover, our preconditioners have the following desirable properties.

1. Since $f \geq 0$ with a finite number of zeros and K_N is a positive kernel, it follows by (2.4) that $f_N > 0$. Thus, the matrices $\boldsymbol{M}_N(f_N)$ are positive definite.
2. In the following section, we will prove that under certain conditions on the kernels K_N, the eigenvalues of $\boldsymbol{M}_N^{-1}\boldsymbol{A}_N$ are bounded from below by a positive constant independent of N and that the number of isolated eigenvalues of $\boldsymbol{M}_N^{-1}\boldsymbol{A}_N$ is independent of N. Then, by Theorem 1.1, the number of PCG steps to achieve a fixed precision is independent of N.
3. By construction (2.8), the multiplication of $\boldsymbol{M}_N$ with a vector requires only $\mathcal{O}(N\log N)$ arithmetical operations by using FFT techniques. By a technique presented in [26] it is possible to implement a PCG method with preconditioner $\boldsymbol{M}_N$ which takes only $\mathcal{O}(N)$ instead of $\mathcal{O}(N\log N)$ arithmetical operations per iteration step more than the original CG method with respect to $\boldsymbol{A}_N$.

3. Eigenvalues of $\boldsymbol{M}_N^{-1}\boldsymbol{A}_N$. In this section, we prove that under certain assumptions on the kernels K_N the eigenvalues of $\boldsymbol{M}_N^{-1}\boldsymbol{A}_N$ are bounded from below by a positive constant independent of N and that the number of isolated eigenvalues of $\boldsymbol{M}_N^{-1}\boldsymbol{A}_N$ is independent of N. For the proof of our main result, we need some preliminary lemmas.

LEMMA 3.1. *Let $p \in C_{2\pi}$ be a nonnegative function which has only a finite number of zeros. Let $h \in C_{2\pi}$ be a positive function with*

$$h_{\min} := \min_{x\in[0,2\pi]} h(x)\,, \quad h_{\max} := \max_{x\in[0,2\pi]} h(x)\,.$$

Then, for $f := ph$ and any $N \in \mathbb{N}$, the eigenvalues of $\boldsymbol{A}_N^{-1}(p)\boldsymbol{A}_N(f)$ lie in the interval $[h_{\min}, h_{\max}]$.

The proof can be found, for example, in [5, 10, 31]. A more sophisticated version for $f, g \in L^1$ was proved in [38, 37].

LEMMA 3.2. *Let p be a real-valued nonnegative trigonometric polynomial of degree $\le s$. Let $N \ge 2s$. Then at most $2s$ eigenvalues of $\boldsymbol{M}_N(p)^{-1}\boldsymbol{A}_N(p)$ differ from 1.*

Proof. For arbitrary $f \in C_{2\pi}$ with pointwise convergent Fourier series, we obtain by replacing $f(x_{N,l})$ in (2.9) by the Fourier series of f at $x_{N,l}$

$$\begin{aligned}
\tilde{a}_k &= \frac{1}{N} \sum_{l=0}^{N-1} \sum_{j\in\mathbb{Z}} a_j \, \mathrm{e}^{\mathrm{i}jx_{N,l}} \, \mathrm{e}^{-2\pi\mathrm{i}lk/N} \, \mathrm{e}^{-\mathrm{i}kw_N} \\
&= \sum_{j=0}^{N-1} a_j \, \mathrm{e}^{-\mathrm{i}w_N k} \, \mathrm{e}^{\mathrm{i}w_N j} \left(\frac{1}{N} \sum_{l=0}^{N-1} \mathrm{e}^{-2\pi\mathrm{i}lk/N} \, \mathrm{e}^{2\pi\mathrm{i}lj/N} \right) \\
&\quad + \sum_{j=0}^{N-1} \sum_{r\in\mathbb{Z}\setminus\{0\}} a_{j+rN} \mathrm{e}^{-\mathrm{i}w_N k} \, \mathrm{e}^{\mathrm{i}w_N (j+rN)} \left(\frac{1}{N} \sum_{l=0}^{N-1} \mathrm{e}^{-2\pi\mathrm{i}lk/N} \, \mathrm{e}^{2\pi\mathrm{i}lj/N} \right) \\
&= a_k \quad + \sum_{r\in\mathbb{Z}\setminus\{0\}} a_{k+rN} \, \mathrm{e}^{\mathrm{i}w_N rN} \, .
\end{aligned}$$

This is well known as *aliasing effect*. Then it follows that

$$\boldsymbol{A}_N(f) = \boldsymbol{M}_N(f) - \boldsymbol{B}_N(f) \, , \tag{3.1}$$

where

$$\boldsymbol{B}_N(f) := (b_{j-k}(f))_{j,k=0}^{N-1} \, , \quad b_k(f) := \sum_{r\in\mathbb{Z}\setminus\{0\}} a_{k+rN}(f) \, \mathrm{e}^{\mathrm{i}w_N rN}.$$

We consider $f = p$. Since p is of degree smaller than $s \le \frac{N}{2}$, we have that $b_k(p) = 0$ for $|k| \le N - 1 - s$. Consequently, $\boldsymbol{B}_N(p)$ is of rank $\le 2s$. Now the assertion follows by (3.1). ∎

In what follows, we restrict our attention to Toeplitz matrices having a nonnegative generating function $f \in C_{2\pi}$ with a zero of even order $2s$ ($s \in N$) at $x = 0$.

We use the trigonometric polynomial

$$p_s(x) := (2 - 2\cos x)^s \; = \; \left(2 \sin \frac{x}{2}\right)^{2s} \; = \; \sum_{k=0}^{s} \alpha_k \cos kx \quad (s \ge 1) \tag{3.2}$$

of degree s which also has a zero of order $2s$ at $x = 0$.

The convergence of our PCG method is related to the behavior of the grid functions

$$q_{s,N}(x) := \frac{p_{s,N}(x)}{p_s(x)} \quad (x \in G_N) \, , \tag{3.3}$$

where $p_{s,N}(x) := (p_s * K_N)(x)$. More precisely, for the proof of our main theorem, we need that $\{q_{s,N}(x)\}_{N\in\mathbb{N}}$ is bounded for all $x \in G_N$ from above and below by positive constants independent of N. This will be the content of the following lemmas.

First, we see that the above property follows immediately for all grid points $x \in G_N$ having some distance independent of N from the zero of f.

LEMMA 3.3. *Let G_N be defined by (2.7) with $w_N \neq 0$. Let $\{K_N\}_{N\in\mathbb{N}}$ be a sequence of positive even reproducing kernels, and let $q_{s,N}$ be given by (3.3). Then, for $x_N \in G_N \cap [a,b]$ ($[a,b] \subset (0,2\pi)$) and for every $\varepsilon > 0$ there exists $N(\varepsilon)$ such that*

$$1 - \varepsilon \leq q_{s,N}(x_N) \leq 1 + \varepsilon$$

for all $N \geq N(\varepsilon)$.

Proof. Since $x_N \in [a,b]$ ($N \in \mathbb{N}$) for some $a > 0, b < 2\pi$, we have that

$$p_s(x_N) \geq \min\{p_s(a), p_s(b)\} > 0\,.$$

Further, we obtain by (2.6) that for every $\varepsilon > 0$ there exists $N(\varepsilon)$ such that

$$|p_s(x) - p_{s,N}(x)| \leq \varepsilon \min\{p_s(a), p_s(b)\} \quad (x \in [0, 2\pi))$$

for all $N \geq N(\varepsilon)$. By rewriting (3.3) in the form

$$q_{s,N}(x_N) \;=\; 1 + \frac{p_{s,N}(x_N) - p_s(x_N)}{p_s(x_N)}$$

we obtain the assertion. ∎

By Lemma 3.3, it remains to consider the sequences $\{q_{s,N}(x_N)\}_{N\in\mathbb{N}}$ for $x_N \in G_N$ with $x_N \to 0$ for $N \to \infty$ or with $x_N \to 2\pi$ for $N \to \infty$. Since both cases require the same ideas, we consider $x_N \in G_N$ with

$$\lim_{N\to\infty} x_N \;=\; 0\,.$$

The existence of a lower bound of $\{q_{s,N}(x_N)\}_{N\in\mathbb{N}}$ does not also require additional properties of the kernel K_N.

LEMMA 3.4. *Let G_N be defined by (2.7) with $w_N \neq 0$. Let $\{K_N\}_{N\in\mathbb{N}}$ be a sequence of positive even reproducing kernels, and let $q_{s,N}$ be given by (3.3). Then, for $x_N \in G_N$ with $\lim_{N\to\infty} x_N = 0$, there exists a constant $\alpha > 0$ independent of N such that*

$$\alpha \leq q_{s,N}(x_N).$$

Proof. By definition of $q_{s,N}$ and $p_{s,N}$, we have that

$$q_{s,N}(x_N) = \frac{1}{2\pi} \int_0^{2\pi} \frac{p_s(t)}{p_s(x_N)}\, K_N(x_N - t)\,\mathrm{d}t$$

and since $p_s \geq 0$ and $K_N \geq 0$, we obtain for $x_N < \pi$ that

$$q_{s,N}(x_N) \geq \frac{1}{2\pi} \int_{x_N}^{\pi} \frac{p_s(t)}{p_s(x_N)}\, K_N(x_N - t)\,\mathrm{d}t\,.$$

The polynomial p_s is monotonely increasing on $[0,\pi]$. Thus

$$q_{s,N}(x_N) \geq \frac{1}{2\pi} \int_{x_N}^{\pi} K_N(x_N - t)\,\mathrm{d}t\,.$$

Since K_N is even and fulfills (2.3), we get for any sequence $x_N \in G_N$ ($x_N < \pi$) with $\lim_{N\to\infty} x_N = 0$ that

$$q_{s,N}(x_N) \geq \frac{1}{2\pi} \int_0^{\pi - x_N} K_N(t)\,\mathrm{d}t \geq \text{const}\,. \qquad ∎$$

It remains to examine if

$$q_{s,N}(x_N) = \frac{p_{s,N}(x_N)}{p_s(x_N)} \le \beta$$

for any $x_N \in G_N$ with $\lim_{N\to\infty} x_N = 0$. Here the "moment property" comes into play.

LEMMA 3.5. *Let G_N $(n \in \mathbb{N})$ be defined by (2.7) with*

$$0 < w \le w_N N \le \tilde{w} < 2\pi\,. \tag{3.4}$$

Let $\{K_N\}_{N\in\mathbb{N}}$ be a sequence of positive even kernels, and let $q_{s,N}$ $(s \ge 1)$ be given by (3.3). Then there exists a constant $\beta < \infty$ independent of N such that

$$q_{s,N}(x_N) \le \beta$$

for all $x_N \in G_N$ with $\lim_{N\to\infty} x_N = 0$ if and only if K_N fulfills the "moment property"

$$\int_{-\pi}^{\pi} t^{2k} K_N(t)\,dt = \mathcal{O}(N^{-2k}) \quad (k = 0, \dots, s)\,. \tag{3.5}$$

Note that the restriction (3.4) on the grids G_N means that we have for any $x_N \in G_N$ that $w/N \le x_N$.

Proof. Since $\sin^2 x \le x^2$ for all $x \in \mathbb{R}$, we obtain by (3.2) that

$$p_s(x) \le x^{2s} \quad (x \in \mathbb{R})\,. \tag{3.6}$$

Similarly, we have for any fixed $0 \le y \le \pi/2$ that

$$\sin^2 x \ge \left(\frac{2}{\pi}\,\frac{\frac{\pi}{2} - y}{\frac{\pi}{2} + y}\right)^2 x^2 \quad \left(x \in \left[-\frac{\pi}{2} - y, \frac{\pi}{2} + y\right]\right)$$

and hence

$$p_s(x) \ge \left(\frac{2}{\pi}\right)^{2s} \left(\frac{\frac{\pi}{2} - y}{\frac{\pi}{2} + y}\right)^{2s} x^{2s} \qquad (x \in [-\pi - 2y, \pi + 2y])\,. \tag{3.7}$$

Using (3.6), we conclude by $K_N \ge 0$ that

$$\begin{aligned} p_{s,N}(x) &= \frac{1}{2\pi} \int_{-\pi}^{\pi} p_s(x - t)\, K_N(t)\,\mathrm{d}t \\ &\le \frac{1}{2\pi} \int_{-\pi}^{\pi} (x - t)^{2s}\, K_N(t)\,\mathrm{d}t \\ &= \frac{1}{2\pi} \sum_{k=0}^{2s} \binom{2s}{k} (-1)^k x^{2s-k} \int_{-\pi}^{\pi} t^k\, K_N(t)\,\mathrm{d}t \quad (x \in [-\pi, \pi)) \end{aligned}$$

and since K_N is even

$$p_{s,N}(x) \le \frac{1}{2\pi} \sum_{k=0}^{s} \binom{2s}{2k} x^{2s-2k} \int_{-\pi}^{\pi} t^{2k}\, K_N(t)\,\mathrm{d}t\,.$$

Let K_N satisfy (3.5). Then

$$p_{s,N}(x) \le \frac{c}{2\pi} \sum_{k=0}^{s} \binom{2s}{2k} x^{2s-2k} N^{-2k}\,.$$

By (3.4), we have for any grid sequence $x_N \in G_N$ that $x_N \geq w/N$. Consequently,

$$p_{s,N}(x_N) \leq C\, x_N^{2s}\,.$$

By (3.7) this implies that there exists $\beta < \infty$ independent of N so that $q_{s,N}(x_N) \leq \beta$.

On the other hand, we see by (3.7) with $y := \pi/4$ that

$$\begin{aligned} p_{s,N}(x) &= \frac{1}{2\pi}\int_{-\pi}^{\pi} p_s(x-t)K_N(t)\,\mathrm{d}t \\ &\geq \frac{1}{2\pi}\left(\frac{2}{3\pi}\right)^{2s}\sum_{k=0}^{s}\binom{2s}{2k}x^{2s-2k}\int_{-\pi}^{\pi} t^{2k}K_N(t)\,\mathrm{d}t \qquad \left(x \in \left[-\frac{\pi}{2},\frac{\pi}{2}\right]\right)\,. \end{aligned}$$

By definition of G_N, there exists a grid sequence $\{x_N\}_{N\in\mathbb{N}}$ so that x_N approaches zero as N^{-1} $(N \to \infty)$. Assume that K_N does not fulfill (3.5). Then we obtain for the above sequence that $p_{s,N}(x_N) \geq c\,N^{-2s+\varepsilon}$ $(\varepsilon > 0)$, while we have by (3.6) that $p_s(x_N) = \mathcal{O}(N^{-2s})$. Thus $q_{s,N}(x_N)$ cannot be bounded from above. This completes the proof. □

By Lemmas 3.3–3.5, we obtain that for grids G_N defined by (2.7) and (3.4) and for even positive reproducing kernels with (3.5) there exist

$$\begin{aligned} 0 < \alpha &:= \inf\{q_{s,N}(x) : x \in G_N; N \in \mathbb{N}\}, \\ \infty > \beta &:= \sup\{q_{s,N}(x) : x \in G_N; N \in \mathbb{N}\}. \end{aligned} \tag{3.8}$$

Now we can prove our main theorem.

THEOREM 3.6. *Let $\{\boldsymbol{A}_N(f)\}_{N\in\mathbb{N}}$ be a sequence of Toeplitz matrices generated by a nonnegative function $f \in C_{2\pi}$ which has only a zero of order $2s$ $(s \in \mathbb{N})$ at $x = 0$. Let the grids G_N be defined by* (2.7) *and* (3.4). *Assume that $\{K_N\}_{N\in\mathbb{N}}$ is a sequence of even positive reproducing kernels satisfying* (3.5). *Finally, let $\boldsymbol{M}_N(f_N)$ be defined by* (2.10). *Then we have the following results:*

(i) *The eigenvalues of $\boldsymbol{M}_N(f_N)^{-1}\boldsymbol{A}_N(f)$ are bounded from below by a positive constant independent of N.*

(ii) *For $N \geq 2s$, at most $2s$ eigenvalues of $\boldsymbol{M}_N(f_N)^{-1}\boldsymbol{A}_N(f)$ are not contained in the interval $[\frac{h_{\min}}{\beta\, h_{\max}}, \frac{h_{\max}}{\alpha\, h_{\min}}]$. Here α, β are given by* (3.8) *and $h_{\min}, h_{\max}$ are defined as in Lemma* 3.1, *where $h := f/p_s$.*

Proof. 1. To show (ii), we consider the Rayleigh quotient

$$\frac{\bar{\boldsymbol{u}}'\,\boldsymbol{A}_N(f)\,\boldsymbol{u}}{\bar{\boldsymbol{u}}'\,\boldsymbol{M}_N(f_N)\,\boldsymbol{u}} \;=\; \frac{\bar{\boldsymbol{u}}'\,\boldsymbol{A}_N(f)\,\boldsymbol{u}}{\bar{\boldsymbol{u}}'\,\boldsymbol{A}_N(p_s)\,\boldsymbol{u}}\;\frac{\bar{\boldsymbol{u}}'\,\boldsymbol{A}_N(p_s)\,\boldsymbol{u}}{\bar{\boldsymbol{u}}'\,\boldsymbol{M}_N(f_N)\,\boldsymbol{u}} \quad (\boldsymbol{u} \neq \boldsymbol{o}_N)\,. \tag{3.9}$$

By Lemma 3.1, we have that

$$\frac{\bar{\boldsymbol{u}}'\,\boldsymbol{A}_N(f)\,\boldsymbol{u}}{\bar{\boldsymbol{u}}'\,\boldsymbol{A}_N(p_s)\,\boldsymbol{u}} \in [h_{\min}, h_{\max}]$$

and thus, since the second factor on the right-hand side of (3.9) is positive,

$$h_{\min}\,\frac{\bar{\boldsymbol{u}}'\,\boldsymbol{A}_N(p_s)\,\boldsymbol{u}}{\bar{\boldsymbol{u}}'\,\boldsymbol{M}_N(f_N)\,\boldsymbol{u}} \;\leq\; \frac{\bar{\boldsymbol{u}}'\,\boldsymbol{A}_N(f)\,\boldsymbol{u}}{\bar{\boldsymbol{u}}'\,\boldsymbol{M}_N(f_N)\,\boldsymbol{u}} \;\leq\; h_{\max}\,\frac{\bar{\boldsymbol{u}}'\,\boldsymbol{A}_N(p_s)\,\boldsymbol{u}}{\bar{\boldsymbol{u}}'\,\boldsymbol{M}_N(f_N)\,\boldsymbol{u}}\,. \tag{3.10}$$

By Lemma 3.2, we know that

$$\boldsymbol{A}_N(p_s) \;=\; \boldsymbol{M}_N(p_s) \;+\; \boldsymbol{R}_N(2s)$$

with a matrix $\boldsymbol{R}_N(2s)$ of rank $2s$ and consequently

$$\frac{\bar{\boldsymbol{u}}'\,\boldsymbol{A}_N(f)\,\boldsymbol{u}}{\bar{\boldsymbol{u}}'\,\boldsymbol{M}_N(f_N)\,\boldsymbol{u}} \;\le\; h_{\max}\,\frac{\bar{\boldsymbol{u}}'\,\boldsymbol{M}_N(p_s)\,\boldsymbol{u}}{\bar{\boldsymbol{u}}'\,\boldsymbol{M}_N(f_N)\,\boldsymbol{u}} \;+\; \frac{\bar{\boldsymbol{u}}'\,h_{\max}\boldsymbol{R}_N(2s)\,\boldsymbol{u}}{\bar{\boldsymbol{u}}'\,\boldsymbol{M}_N(f_N)\,\boldsymbol{u}}\,,$$

and

$$\frac{\bar{\boldsymbol{u}}'\,(\boldsymbol{A}_N(f)-h_{\max}\boldsymbol{R}_N(2s))\,\boldsymbol{u}}{\bar{\boldsymbol{u}}'\,\boldsymbol{M}_N(f_N)\,\boldsymbol{u}} \;\le\; h_{\max}\,\frac{\bar{\boldsymbol{u}}'\,\boldsymbol{M}_N(p_s)\,\boldsymbol{u}}{\bar{\boldsymbol{u}}'\,\boldsymbol{M}_N(f_N)\,\boldsymbol{u}}\,.$$

Since K_N and p_s are nonnegative, we obtain by (2.4) and by definition of h that

$$h_{\min}\,p_{s,N}(x) \;\le\; f_N(x) \;\le\; h_{\max}\,p_{s,N}(x) \quad x\in[0,2\pi]\,.$$

This implies by definition of $\boldsymbol{M}_N(f_N)$ that

$$\frac{\bar{\boldsymbol{u}}'\,(\boldsymbol{A}_N(f)-h_{\max}\boldsymbol{R}_N(2s))\,\boldsymbol{u}}{\bar{\boldsymbol{u}}'\,\boldsymbol{M}_N(f_N)\,\boldsymbol{u}} \;\le\; \frac{h_{\max}}{h_{\min}}\,\frac{\bar{\boldsymbol{u}}'\,\boldsymbol{M}_N(p_s)\,\boldsymbol{u}}{\bar{\boldsymbol{u}}'\,\boldsymbol{M}_N(p_{s,N})\,\boldsymbol{u}}$$

and further by (3.3), (3.8) and since $0<\alpha\le\beta<\infty$ that

$$\frac{\bar{\boldsymbol{u}}'\,(\boldsymbol{A}_N(f)-h_{\max}\boldsymbol{R}_N(2s))\,\boldsymbol{u}}{\bar{\boldsymbol{u}}'\,\boldsymbol{M}_N(f_N)\,\boldsymbol{u}} \;\le\; \frac{h_{\max}}{\alpha\,h_{\min}}$$

for all $\boldsymbol{u}\ne\boldsymbol{o}_N$. Assume that $\boldsymbol{R}_N(2s)$ has s_1 positive eigenvalues. Then, by properties of the Rayleigh quotient and by Weyl's theorem [24, p. 184] at most s_1 eigenvalues of $\boldsymbol{M}_N(f_N)^{-1}\boldsymbol{A}_N(f)$ are larger than $\frac{h_{\max}}{\alpha\,h_{\min}}$. Similarly, we obtain by consideration of the left-hand inequality of (3.10) that at most $2s-s_1$ eigenvalues of $\boldsymbol{M}_N(f_N)^{-1}\boldsymbol{A}_N(f)$ are smaller than $\frac{h_{\min}}{\beta\,h_{\max}}$.

2. To show (i), we rewrite (3.9) as

$$\frac{\bar{\boldsymbol{u}}'\,\boldsymbol{A}_N(f)\,\boldsymbol{u}}{\bar{\boldsymbol{u}}'\,\boldsymbol{M}_N(f_N)\,\boldsymbol{u}} \;=\; \frac{\bar{\boldsymbol{u}}'\,\boldsymbol{A}_N(f)\,\boldsymbol{u}}{\bar{\boldsymbol{u}}'\,\boldsymbol{A}_N(p_s)\,\boldsymbol{u}}\;\frac{\bar{\boldsymbol{u}}'\,\boldsymbol{M}_N(p_s)\,\boldsymbol{u}}{\bar{\boldsymbol{u}}'\,\boldsymbol{M}_N(f_N)\,\boldsymbol{u}}\;\frac{\bar{\boldsymbol{u}}'\,\boldsymbol{A}_N(p_s)\,\boldsymbol{u}}{\bar{\boldsymbol{u}}'\,\boldsymbol{M}_N(p_s)\,\boldsymbol{u}} \quad (\boldsymbol{u}\ne\boldsymbol{o}_N)\,.$$

As in the first part of the proof, we see that this implies

$$\frac{\bar{\boldsymbol{u}}'\,\boldsymbol{A}_N(f)\,\boldsymbol{u}}{\bar{\boldsymbol{u}}'\,\boldsymbol{M}_N(f_N)\,\boldsymbol{u}} \;\ge\; \frac{h_{\min}}{\beta\,h_{\max}}\;\frac{\bar{\boldsymbol{u}}'\,\boldsymbol{A}_N(p_s)\,\boldsymbol{u}}{\bar{\boldsymbol{u}}'\,\boldsymbol{M}_N(p_s)\,\boldsymbol{u}}\,.$$

Consequently, it remains to show that there exists a constant $0<c<\infty$ such that

$$\frac{\bar{\boldsymbol{u}}'\,\boldsymbol{A}_N(p_s)\,\boldsymbol{u}}{\bar{\boldsymbol{u}}'\,\boldsymbol{M}_N(p_s)\,\boldsymbol{u}} \;\ge\; \frac{1}{c}\,.$$

By (3.1), this is equivalent to

$$1 \;+\; \frac{\bar{\boldsymbol{u}}'\,\boldsymbol{B}_N(p_s)\,\boldsymbol{u}}{\bar{\boldsymbol{u}}'\,\boldsymbol{A}_N(p_s)\,\boldsymbol{u}} \;\le\; c\,.$$

By the special structure of $\boldsymbol{B}_N(p_s)$ and $\boldsymbol{A}_N(p_s)$, assertion (i) follows as in the proof of Theorem 4.3 in [3]. This completes the proof. □

By the following theorem, the "moment property" (3.5) of the kernel is also *necessary* to obtain good preconditioners.

THEOREM 3.7. *Let $\{\boldsymbol{A}_N(f)\}_{N\in\mathbb{N}}$ be a sequence of Toeplitz matrices generated by a nonnegative function $f \in C_{2\pi}$ which has only a zero of order $2s$ ($s \in \mathbb{N}$) at $x = 0$. Let the grids G_N be defined by* (2.7) *and* (3.4). *Assume that $\{K_N\}_{N\in\mathbb{N}}$ is a sequence of even positive reproducing kernels which do not fulfill* (3.5). *Finally, let $\boldsymbol{M}_N(f_N)$ be defined by* (2.10). *Then, for arbitrary $\varepsilon > 0$ and arbitrary $c \in \mathbb{N}$, there exist $N(\varepsilon, c)$ such that for all $N \geq N(\varepsilon, c)$ at least c eigenvalues of $\boldsymbol{M}_N(f_N)^{-1}\boldsymbol{A}_N(f)$ are contained in $(0, \varepsilon)$.*

The proof follows again the lines of the fundamental paper of Di Benedetto [3, Theorem 5.4]. We include the short proof with respect to our background.

Proof. By the proof of Theorem 3.6, we have for all $\boldsymbol{u} \neq \boldsymbol{o}$ that

$$\frac{\bar{\boldsymbol{u}}'\,\boldsymbol{A}_N(f)\,\boldsymbol{u}}{\bar{\boldsymbol{u}}'\,\boldsymbol{M}_N(f_N)\,\boldsymbol{u}} \;\leq\; \frac{h_{\max}}{h_{\min}}\,\frac{\bar{\boldsymbol{u}}'\,\boldsymbol{A}_N(p_s)\,\boldsymbol{u}}{\bar{\boldsymbol{u}}'\,\boldsymbol{M}_N(p_{s,N})\,\boldsymbol{u}}\,.$$

Hence it remains to show that $\boldsymbol{M}_N(p_{s,N})^{-1}\boldsymbol{A}_N(p_s)$ has an arbitrary number of eigenvalues in $(0, \varepsilon)$ for N sufficiently large. By (3.2) and [32, Theorem 3.1], we have that

$$\begin{aligned}
\boldsymbol{T}_{N+2s-2} &:= \boldsymbol{S}^I_{N+2s-2}\,\mathrm{diag}\left(\left(2\,\sin\frac{j\pi}{2(N+2s-1)}\right)^{2s}\right)_{j=1}^{N+2s-2}\boldsymbol{S}^I_{N+2s-2}\\
&= \boldsymbol{S}^I_{N+2s-2}\,\mathrm{diag}\left(\sum_{k=0}^{s}\alpha_k\cos\frac{jk\pi}{N+2s-1}\right)_{j=1}^{N+2s-2}\boldsymbol{S}^I_{N+2s-2}\\
&= \frac{1}{2}\,\mathrm{stoep}(2\alpha_0,\,\ldots\,,\alpha_s,0,\,\ldots\,,0) \;-\; \frac{1}{2}\,\mathrm{shank}(\alpha_2,\,\ldots\,,\alpha_s,0,\,\ldots\,,0)\,,
\end{aligned}$$

where $\boldsymbol{S}^I_{N-1} := (2/N)^{1/2}(\sin\frac{(j+1)(k+1)\pi}{N})_{j,k=0}^{N-2}$ is an orthogonal matrix and where $\mathrm{stoep}\,\boldsymbol{a}'$ and $\mathrm{shank}\,\boldsymbol{a}'$ denote the symmetric Toeplitz matrix and the persymmetric Hankel matrix with first row $\boldsymbol{a}'$, respectively. Deleting the first $s-1$ and the last $s-1$ rows and columns of $\boldsymbol{T}_{N+2s-2}$ we obtain $\boldsymbol{A}_N(p_s)$. Thus, we have by Courant's minimax theorem for the eigenvalues $\lambda_1(\boldsymbol{A}_N(p_s)) \leq \cdots \leq \lambda_N(\boldsymbol{A}_N(p_s))$ of $\boldsymbol{A}_N(p_s)$ that

$$\lambda_j(\boldsymbol{A}_N(p_s)) \leq \lambda_{j+2s-2}(\boldsymbol{T}_{N+2s-2}) = \left(2\,\sin\frac{j+2s-2}{2(N+2s-1)}\right)^{2s} \leq \left(\frac{j+2s-2}{N+2s-1}\right)^{2s}.$$

The later result is due to a technique of Bini and Capovani [7, Proposition 4.2]. Consider $\boldsymbol{A}_N(p_s) - t\boldsymbol{M}_N(p_{s,N})$. For $t = 0$, this matrix has positive eigenvalues, while we have for arbitrary $\varepsilon > 0$ that

$$\begin{aligned}
\lambda_j(\boldsymbol{A}_N(p_s) - \varepsilon\,\boldsymbol{M}_N(p_{s,N})) &\leq \lambda_j(\boldsymbol{A}_N(p_s)) \;-\; \varepsilon\,\lambda_{\min}(\boldsymbol{M}_N(p_{s,N}))\\
&\leq \left(\frac{j+2s-2}{N+2s-1}\right)^{2s} \;-\; \varepsilon\,p_{s,N}(w_N)\\
&= N^{-2s}\left(\left(\frac{j+2s-2}{1+\frac{2s-1}{N}}\right)^{2s} \;-\; \varepsilon\,\frac{p_{s,N}(w_N)}{N^{2s}}\right).
\end{aligned}$$

Since K_N does not fulfill (3.5), we have by Lemma 3.5 that

$$\lim_{N\to\infty}\frac{p_{s,N}(w_N)}{N^{2s}} = \infty\,.$$

Thus, for $j \leq c$ independent of N and for sufficiently large $N \geq N(\varepsilon, c)$ the values $\lambda_j(\boldsymbol{A}_N(p_s) - \varepsilon \boldsymbol{M}_N(p_{s,N}))$ become negative. The eigenvalues of $\boldsymbol{A}_N(p_s) - t\,\boldsymbol{M}_N(p_{s,N})$ are continuous functions of t. Since the smallest c eigenvalues pass from a positive value for $t = 0$ to a negative value for $t = \varepsilon$, there exist $\varepsilon_1, \ldots, \varepsilon_c \in (0, \varepsilon)$ such that $\boldsymbol{A}_N(p_s) - \varepsilon_j \boldsymbol{M}_N(p_{s,N})$ has eigenvalue zero. This is equivalent to the fact that $\boldsymbol{M}_N(p_{s,N})^{-1}\boldsymbol{A}_N(p_s)$ has an eigenvalue $\varepsilon_j \in (0, \varepsilon)$, and we are done with the proof. □

The generalization of the above results for generating functions with different zeros of even order

$$f(x) \;=\; (x - y_1)^{2s_1} \ldots (x - y_m)^{2s_m} \tilde{f}(x) \quad (\tilde{f} > 0)$$

is straightforward (see [18]). By applying the polynomial

$$p(x) := \prod_{i=1}^{m} p_{s_i}(x - y_i)$$

instead of p_s and following the above lines, we can show that for grids G_N of the form (2.7) with $x_{N,l} \neq y_i$ $(l = 0, \ldots, N-1;\ i = 1, \ldots, m)$ and for kernels K_N fulfilling (3.5) with $s := \max\{s_j : j = 1, \ldots, m\}$, there exist constants $0 < \alpha \leq \beta < \infty$ such that for all $x \in G_N$

$$\alpha \leq \frac{(p * K_N)(x)}{p(x)} \leq \beta\,.$$

4. Jackson polynomials and *B*-spline kernels. In this section, we consider concrete positive reproducing kernels K_N with property (3.5).

The *generalized Jackson polynomials* of degree $\leq N - 1$ are defined by

$$J_{m,N}(x) = \lambda_{m,N} \left(\frac{\sin(nx/2)}{\sin x/2} \right)^{2m} \qquad (m \in \mathbb{N})\,,$$

where $n := \lfloor \frac{N-1}{m} \rfloor + 1$ and where $\lambda_{m,N}$ is determined by (2.3) [22, p. 203]. It is well known [22, p. 204] that the generalized Jackson polynomials $J_{m,N}$ are even positive reproducing kernels which satisfy property (3.5) for

$$m \geq s + 1\,.$$

In particular, $J_{1,N}$ is the *Fejér kernel* which is related to the optimal circulant preconditioner [19, 15]. However, the Fejér kernel does not fulfill (3.5) for $s \geq 1$ such that we cannot expect a fast convergence of our PCG method if f has a zero of order ≥ 2. Our numerical tests confirm this result.

By Theorem 3.6, the generalized Jackson polynomials $K_N = J_{m,N}$ with $m \geq s+1$ can be used for the construction of preconditioners. Note that preconditioners related to Jackson kernels were also suggested in [39]. However, the construction of the Fourier coefficients of $J_{m,N}$ seems to be rather complicated. See also [10]. Therefore we prefer the following B-spline kernels.

The "B-spline kernels" were introduced by Chan, Tso, and Sun in [14]. The authors showed by numerical tests that preconditioners from B-spline kernels of certain order seem to be good candidates for the PCG method. Applying the results of the previous section, we are able to show the theoretical reasons for these results, at least for the positive B-spline kernels.

Let $\chi_{[0,1)}$ denote the characteristic function of $[0,1)$. The *cardinal B-splines* N_m $(m \geq 1)$ *of order* m are defined by

$$N_1 := \chi_{[0,1)}\,,\ N_{m+1} := \int_0^1 N_1(t)\ N_m(\cdot - t)\ \mathrm{d}t$$

and their centered version by

$$M_m := N_m\left(\cdot + \frac{m}{2}\right)\ .$$

Note that M_m is an even function with $\operatorname{supp} M_m = [-\frac{m}{2}, \frac{m}{2}]$ and that

$$\int_{-\infty}^{\infty} M_m(t)\ \mathrm{e}^{-\mathrm{i}xt}\ \mathrm{d}t = \left(\operatorname{sinc}\frac{x}{2}\right)^m\ , \tag{4.1}$$

where

$$\operatorname{sinc} x := \begin{cases} \dfrac{\sin x}{x}, & x \neq 0\ , \\ 1, & x = 0\ . \end{cases}$$

Let the *B-spline kernels* $B_{m,N}$ be defined by [14]

$$B_{m,N}(x) := 1 + \frac{2}{M_{2m}(0)} \sum_{k=1}^{N-1} M_{2m}\left(\frac{mk}{N}\right)\ \cos kx\ .$$

Note that $B_{1,N}$ again coincides with the Fejér kernel.

For the construction of the preconditioner, it is important that the Fourier coefficient $c_{N,k} = M_{2m}(\frac{mk}{N})/M_{2m}(0)$ can be computed in a simple way, for example, by applying a simplified version of de Boor's algorithm [9, p. 54].

By (4.1), it is easy to check that $B_{m,N}$ is a dilated, 2π-periodized version of $(\operatorname{sinc}\frac{x}{2})^{2m}$, i.e.,

$$B_{m,N}(x) = \frac{N}{m}\ \frac{1}{M_{2m}(0)}\ \sum_{r\in\mathbb{Z}} \left(\operatorname{sinc}\ \left(\frac{N}{m}\left(\frac{x+2\pi r}{2}\right)\right)\right)^{2m}\ . \tag{4.2}$$

Thus

$$B_{m,N} \geq 0 \quad (m \in \mathbb{N})\ .$$

Moreover, we obtain similar to the generalized Jackson polynomials the following lemma.

LEMMA 4.1. *The B-spline kernels* $B_{m,N}$ *satisfy* (3.5) *if and only if* $m \geq s+1$.

Proof. By (4.2), we obtain that

$$\begin{aligned}
\int_{-\pi}^{\pi} t^{2k}\, B_{m,N}(t)\,\mathrm{d}t &= \frac{N}{m}\ \frac{1}{M_{2m}(0)} \int_{-\pi}^{\pi} t^{2k} \sum_{r\in\mathbb{Z}} \left(\operatorname{sinc}\ \left(\frac{N}{m}\left(\frac{t+2\pi r}{2}\right)\right)\right)^{2m}\ \mathrm{d}t \\
&\leq \frac{N}{m}\ \frac{1}{M_{2m}(0)} \int_{-\infty}^{\infty} t^{2k} \left(\frac{\sin\left(\frac{N}{m}\frac{t}{2}\right)}{\frac{N}{m}\frac{t}{2}}\right)^{2m}\ \mathrm{d}t \\
&= \frac{2}{M_{2m}(0)} \int_{-\infty}^{\infty} \left(\frac{2mu}{N}\right)^{2k} \left(\frac{\sin u}{u}\right)^{2m}\ \mathrm{d}u \\
&\leq c\, N^{-2k} \int_{-\infty}^{\infty} u^{2k-2m}\,\mathrm{d}u \leq C\, N^{-2k}
\end{aligned}$$

for $m \geq k+1$. Thus, for $m \geq s+1$ the kernels $B_{m,N}$ satisfy property (3.5).

On the other hand, we have that

$$\begin{aligned}\int_{-\pi}^{\pi} t^{2k}\, B_{m,N}(t)\,\mathrm{d}t &\geq \frac{N}{m}\,\frac{1}{M_{2m}(0)}\int_{-\pi}^{\pi} t^{2k}\left(\frac{\sin\frac{N}{m}\left(\frac{t}{2}\right)}{\frac{N}{m}\left(\frac{t}{2}\right)}\right)^{2m}\mathrm{d}t\\ &= \frac{2}{M_{2m}(0)}\int_{-N\pi/(2m)}^{N\pi/(2m)}\left(\frac{2mu}{N}\right)^{2k}\left(\frac{\sin u}{u}\right)^{2m}\mathrm{d}u\\ &= \frac{2\,(2m)^{2k}}{M_{2m}(0)}\,N^{-2k}\int_{-N\pi/(2m)}^{N\pi/(2m)}\frac{(\sin u)^{2m}}{u^{2m-2k}}\,\mathrm{d}u\,.\end{aligned}$$

If $m \leq k$, then the last integral is not bounded for $N \to \infty$. Thus, for $m \leq s$, the kernel $B_{m,N}$ does not fulfill property (3.5). ∎

By Theorem 3.6, the B-spline kernels $K_N = B_{m,N}$ with $m \geq s+1$ produce good preconditioners.

5. Generalizations of the preconditioning technique. In this section, we sketch how our preconditioners can be generalized to (real) symmetric Toeplitz matrices and to BTTB matrices. We will do this in a very short way since both cases do not require new ideas. However, we have to introduce some notation to understand the numerical tests in section 7.

Symmetric Toeplitz matrices. First, we suppose in addition to section 2 that the Toeplitz matrices $\boldsymbol{A}_N \in \mathbb{R}^{N,N}$ are symmetric, i.e., the generating function $f \in C_{2\pi}$ is even. Note that in this case, the multiplication of a vector with $\boldsymbol{A}_N$ can be realized using *fast trigonometric transforms* instead of FFTs (see [32]). In this way, complex arithmetic can be completely avoided in the iterative solution of (2.1). This is one of the reasons to look for preconditioners of type (2.8), where the Fourier matrix $\boldsymbol{F}_N$ is replaced by trigonometric matrices corresponding to fast trigonometric transforms. In practice, four discrete sine transforms (DST I–IV) and four discrete cosine transforms (DCT I–IV) were applied (see [46]). Any of these eight trigonometric transforms can be realized with $\mathcal{O}(N \log N)$ arithmetical operations (see, for example, [2, 44]). Likewise, we can define preconditioners with respect to any of these transforms. In this paper, we restrict our attention to the DST–II and DCT–II, which are determined by the following transform matrices:

$$\begin{aligned}\text{DCT–II}: \quad \boldsymbol{C}_N^{II} &:= \left(\frac{2}{N}\right)^{1/2}\left(\varepsilon_j^N\,\cos\,\frac{j(2k+1)\pi}{2N}\right)_{j,k=0}^{N-1} \in \mathbb{R}^{N,N},\\ \text{DST–II}: \quad \boldsymbol{S}_N^{II} &:= \left(\frac{2}{N}\right)^{1/2}\left(\varepsilon_{j+1}^N\,\sin\,\frac{(j+1)(2k+1)\pi}{2N}\right)_{j,k=0}^{N-1} \in \mathbb{R}^{N,N},\end{aligned}$$

where $\varepsilon_k^N := 2^{-1/2}$ $(k = 0, N)$ and $\varepsilon_k^N := 1$ $(k = 1, \ldots, N-1)$. Similar to (2.10), (2.8), we introduce the preconditioners (see [31])

$$\begin{aligned}\mathrm{DCT-II}: \qquad \boldsymbol{M}_N(f_N, \boldsymbol{C}_N^{II}) &:= (\boldsymbol{C}_N^{II})'\,\mathrm{diag}\left(f_N\left(\frac{l\pi}{N}\right)\right)_{l=0}^{N-1}\boldsymbol{C}_N^{II},\\ \mathrm{DST-II}: \qquad \boldsymbol{M}_N(f_N, \boldsymbol{S}_N^{II}) &:= (\boldsymbol{S}_N^{II})'\,\mathrm{diag}\left(f_N\left(\frac{l\pi}{N}\right)\right)_{l=1}^{N}\boldsymbol{S}_N^{II}\,.\end{aligned} \tag{5.1}$$

We recall that for the construction of these preconditioners no explicit knowledge of the generating function is required. Since f is even, the grids G_N are simply chosen as

$G_N := \{x_{N,l} := \frac{l\pi}{N} : l = 0, \dots, N-1\}$ and $G_N := \{x_{N,l} := \frac{(l+1)\pi}{N} : l = 0, \dots, N-1\}$ for the DCT–II and the DST–II preconditioners, respectively. If $f(x_{N,l}) \neq 0$ ($l = 0, \dots, N$), then we can prove Theorem 3.6 with respect to the preconditioners (5.1) in a completely similar way. We have only to replace the decomposition (3.1) by

$$\boldsymbol{A}_N(f) = \boldsymbol{M}_N(f, \boldsymbol{C}_N^{II}) - \operatorname{shank}(a_1, \dots, a_{N-1}, 0),$$
$$\boldsymbol{A}_N(f) = \boldsymbol{M}_N(f, \boldsymbol{S}_N^{II}) + \operatorname{shank}(a_1, \dots, a_{N-1}, 0)$$

for the DCT–II and for the DST–II, respectively. See also [31].

Remark. Let

$$\mathcal{A}_{O_N} := \{\bar{\boldsymbol{O}}_N'(\operatorname{diag} \boldsymbol{d})\boldsymbol{O}_N : \boldsymbol{d} \in \mathbb{R}^N\}$$

denote the matrix algebra with respect to the unitary matrix $\boldsymbol{O}_N$. Then the *optimal preconditioner* $\boldsymbol{M}_N \in \mathcal{A}_{O_N}$ *of* $\boldsymbol{A}_N$ *in* $\mathcal{A}_{O_N}$ is defined by

$$||\boldsymbol{M}_N - \boldsymbol{A}_N||_F = \min\{||\boldsymbol{P} - \boldsymbol{A}_N||_F : \boldsymbol{P} \in \mathcal{A}_{O_N}\},$$

where $\|\cdot\|_F$ denotes the Frobenius norm. As mentioned in the previous section, the optimal preconditioner in $\mathcal{A}_{F_N}$ coincides with our preconditioner (2.10) defined with respect to the Fejér kernel $B_{1,N}$ and with $w_N = 0$ in (2.7). It is easy to check (see [33]) that the optimal preconditioner in $\mathcal{A}_{O_N}$, where $\boldsymbol{O}_N \in \{\boldsymbol{C}_N^{IV}, \boldsymbol{S}_N^{IV}\}$, is equal to our preconditioner $\boldsymbol{M}_N(f_N, \boldsymbol{O}_N)$ in (5.1) defined with respect to $\boldsymbol{O}_N$ and with respect to the Fejér kernel. Unfortunately, the Fejér kernel preconditioners do not lead to a fast convergence of the PCG method if the generating function f of $\boldsymbol{A}_N$ has a zero of order $2s \geq 2$.

In contrast to these results, the optimal preconditioners in $\mathcal{A}_{O_N}$ with $\boldsymbol{O}_N$ defined by the DCT I–III or by the DST I–III do not coincide with the corresponding Fejér kernel preconditioner $\boldsymbol{M}_N(f_N, \boldsymbol{O}_N)$ in (5.1). In literature [6, 3], so-called optimal Tau preconditioners were of special interest. Using our notation, optimal Tau preconditioners are the optimal preconditioners with respect to the DST–I as unitary transform. The optimal Tau preconditioner realizes a fast convergence of the PCG method if the generating function f of $\boldsymbol{A}_N$ has only zeros of order $2s \leq 2$ [6].

Block Toeplitz matrices with Toeplitz blocks. Next we are interested in the solution of BTTB matrices. The construction of preconditioners with the help of reproducing kernels was applied to well-conditioned block Toeplitz systems in [27]. Following these lines, we generalize our univariate construction to ill-conditioned block Toeplitz systems with Toeplitz blocks. In the next section we will present good numerical results also for the block case. However, in general, it is not possible to prove the convergence of PCG in a number of iteration steps independent of N. Here we refer to [34].

Note that as in the univariate case there exist banded block Toeplitz preconditioners with banded Toeplitz blocks which ensure a fast convergence of the corresponding PCG method [35]. See also [4, 30].

We consider systems of linear equations

$$\boldsymbol{A}_{M,N}\boldsymbol{x} = \boldsymbol{b},$$

where $\boldsymbol{A}_{M,N}$ denotes a positive *definite doubly symmetric block Toeplitz matrix with Toeplitz blocks* (BTTB matrix), i.e.,

$$\boldsymbol{A}_{M,N} := (\boldsymbol{A}_{r-s})_{r,s=0}^{M-1} \quad \text{with} \quad \boldsymbol{A}_r := (a_{r,j-k})_{j,k=0}^{N-1}$$

and $a_{r,j} = a_{|r|,|j|}$. We assume that the matrices $\boldsymbol{A}_{M,N}$ are generated by a real-valued 2π-periodic continuous even function in two variables, i.e.,

$$a_{j,k} := \frac{1}{4\pi^2} \int_0^{2\pi} \int_0^{2\pi} \varphi(s,t)\, \mathrm{e}^{-\mathrm{i}(sj+tk)}\, \mathrm{d}s\, \mathrm{d}t\,.$$

Note that the multiplication of a vector with a BTTB matrix requires only $\mathcal{O}(MN \log(MN))$ arithmetical operations (see [33]). We define our so-called "*level*-2" *preconditioners* by

$$\begin{aligned}
\boldsymbol{M}_{M,N}(\varphi_{M,N}, \boldsymbol{C}_M^{II} \otimes \boldsymbol{C}_N^{II}) &:= (\boldsymbol{C}_M^{II} \otimes \boldsymbol{C}_N^{II})'\ \mathrm{diag}\left(\mathrm{col}\left(\varphi_{M,N}\left(\frac{r\pi}{M}, \frac{j\pi}{N}\right)\right)_{j,k=0}^{N-1,M-1}\right) \\
&\quad \times (\boldsymbol{C}_M^{II} \otimes \boldsymbol{C}_N^{II})\,, \\
\boldsymbol{M}_{M,N}(\varphi_{M,N}, \boldsymbol{S}_M^{II} \otimes \boldsymbol{S}_N^{II}) &:= (\boldsymbol{S}_M^{II} \otimes \boldsymbol{S}_N^{II})'\ \mathrm{diag}\left(\mathrm{col}\left(\varphi_{M,N}\left(\frac{r\pi}{M}, \frac{j\pi}{N}\right)\right)_{j,k=1}^{N,M}\right) \\
&\quad \times (\boldsymbol{S}_M^{II} \otimes \boldsymbol{S}_N^{II})\,, \qquad (5.2)
\end{aligned}$$

with $\varphi_{M,N} = \varphi * K_{M,N}$ and $K_{M,N}(x,y) := K_M(x) K_N(y)$. Here col: $\mathbb{R}^{N,M} \to \mathbb{R}^{MN}$ is defined by

$$\mathrm{col}\,(x_{j,k})_{j=0,k=0}^{N-1,M-1} := (x_r)_{r=0}^{MN-1} \quad \text{with} \quad x_{kN+j} := x_{j,k}\,.$$

6. Numerical examples. In this section, we confirm our theoretical results by various numerical examples. The fast computation of the preconditioners and the PCG method were implemented in MATLAB, where the C programs for the fast trigonometric transforms were included by cmex. The algorithms were tested on a Sun SPARCstation 20.

As transform length we choose $N = 2^n$ and as right-hand side $\boldsymbol{b}$ of (2.1) the vector consisting of N entries "1". The PCG method started with the zero vector and stopped if $\|\boldsymbol{r}^{(j)}\|_2 / \|\boldsymbol{r}^{(0)}\|_2 < 10^{-7}$, where $\boldsymbol{r}^{(j)}$ denotes the residual vector after j iterations.

We restrict our attention to preconditioners (2.10) and (5.1) constructed from B-spline kernels $K_N = B_{m,N}$. The following tables (Tables 1–6) show the number of iterations of the corresponding PCG method to achieve a fixed precision. The first row of each table contains the exponent n of the transform length $N = 2^n$ in the univariate case and the block length N in the block Toeplitz case. The kernels are listed in the first column and the applied unitary transform are listed in the second column of each table. Here $\boldsymbol{F}_N^w := \boldsymbol{W}_N \boldsymbol{F}_N$ with $\boldsymbol{W}_N := \mathrm{diag}(\,\mathrm{e}^{-\mathrm{i}k\pi/N})_{k=0}^{N-1}$, i.e., $w_N := \pi/N$ in (2.7). For comparison, the second row of each table contains the number of PCG steps with preconditioner $\boldsymbol{M}_N(f)$ defined by (2.8). These preconditioners, which can be constructed only if the generating function f is known, were examined in [31].

We begin with symmetric ill-conditioned Toeplitz matrices $\boldsymbol{A}_N(f)$ arising from the generating functions

(i) (see [13, 14, 33]): $f(x) := x^2 \quad (x \in [-\pi, \pi))$.

(ii) (see [3, 10, 11, 14, 31, 36]): $f(x) := x^4 \quad (x \in [-\pi, \pi))$.

Tables 1 and 2 present the number of iteration steps with different preconditioners.

As expected, for $f(x) = x^2$ it is not sufficient to choose a preconditioner based on the Fejér kernel $K_N = B_{1,N}$ and for $f(x) = x^4$ it is not sufficient to choose a preconditioner based on the cubic B-spline kernel $K_N = B_{2,N}$ in order to keep the number of iterations independent of N.

TABLE 1
$f(x) = x^2 \quad (x \in [-\pi, \pi))$.

K_N	$\boldsymbol{O}_N$	4	5	6	7	8	9	10	11	12
f	$\boldsymbol{F}_N^w$	4	4	4	5	6	6	6	6	6
$B_{1,N}$	$\boldsymbol{F}_N$	7	8	11	12	14	18	22	29	39
$B_{1,N}$	$\boldsymbol{F}_N^w$	7	8	9	11	13	17	20	26	37
$B_{1,N}$	$\boldsymbol{C}_N^{II}$	7	8	10	11	13	16	20	25	33
$B_{1,N}$	$\boldsymbol{S}_N^{II}$	7	8	9	11	14	17	21	27	38
$B_{2,N}$	$\boldsymbol{F}_N$	6	6	6	7	7	7	6	6	6
$B_{2,N}$	$\boldsymbol{F}_N^w$	6	6	5	5	5	6	6	6	6
$B_{2,N}$	$\boldsymbol{C}_N^{II}$	6	6	6	6	6	6	5	5	5
$B_{2,N}$	$\boldsymbol{S}_N^{II}$	6	6	5	5	5	7	7	7	7
$B_{3,N}$	$\boldsymbol{F}_N$	6	6	6	7	7	7	7	6	6
$B_{3,N}$	$\boldsymbol{F}_N^w$	6	6	6	6	5	6	6	6	6
$B_{3,N}$	$\boldsymbol{C}_N^{II}$	6	6	6	6	6	6	6	5	5
$B_{3,N}$	$\boldsymbol{S}_N^{II}$	6	6	5	7	6	7	7	7	7

TABLE 2
$f(x) = x^4 \quad (x \in [-\pi, \pi))$.

K_N	$\boldsymbol{O}_N$	4	5	6	7	8	9	10	11	12
f	$\boldsymbol{F}_N^w$	6	6	6	8	11	11	11	12	14
$B_{1,N}$	$\boldsymbol{F}_N$	8	15	23	36	61	153	391	> 800	> 800
$B_{1,N}$	$\boldsymbol{F}_N^w$	8	15	23	36	61	153	390	> 800	> 800
$B_{1,N}$	$\boldsymbol{C}_N^{II}$	8	13	20	32	53	129	319	> 800	> 800
$B_{1,N}$	$\boldsymbol{S}_N^{II}$	8	16	24	38	65	158	402	> 800	> 800
$B_{2,N}$	$\boldsymbol{F}_N$	9	9	11	11	13	15	18	22	27
$B_{2,N}$	$\boldsymbol{F}_N^w$	9	9	10	10	13	14	17	20	26
$B_{2,N}$	$\boldsymbol{C}_N^{II}$	8	8	9	9	9	11	13	14	16
$B_{2,N}$	$\boldsymbol{S}_N^{II}$	10	10	10	11	13	14	18	19	22
$B_{3,N}$	$\boldsymbol{F}_N$	9	11	11	12	12	12	13	15	14
$B_{3,N}$	$\boldsymbol{F}_N^w$	9	9	10	10	12	12	13	13	13
$B_{3,N}$	$\boldsymbol{C}_N^{II}$	8	9	9	9	9	9	10	10	9
$B_{3,N}$	$\boldsymbol{S}_N^{II}$	10	10	12	12	14	14	14	15	16

On the other hand, we have a similar convergence behavior for the different unitary transforms. This is no surprise for $\boldsymbol{F}_N^w$ and for $\boldsymbol{S}_N^{II}$. However, for $\boldsymbol{F}_N$ and for $\boldsymbol{C}_N^{II}$, the corresponding grids G_N contain the zero of f, namely, $x_{N,0} = 0$. This was excluded in Theorem 3.6. In our numerical tests it seems to play no rule that a grid point meets the zero of f.

TABLE 3
$f(x) = (x^2 - 1)^2 \quad (x \in [-\pi, \pi))$.

K_N	$\boldsymbol{O}_N$	4	5	6	7	8	9	10	11	12
f	$\boldsymbol{F}_N^w$	7	5	5	7	8	8	7	7	7
$B_{1,N}$	$\boldsymbol{F}_N$	7	13	15	20	27	34	46	63	86
$B_{1,N}$	$\boldsymbol{F}_N^w$	7	14	16	20	26	32	44	59	83
$B_{1,N}$	$\boldsymbol{C}_N^{II}$	8	13	15	18	25	30	41	55	75
$B_{1,N}$	$\boldsymbol{S}_N^{II}$	8	14	16	19	26	33	43	57	79
$B_{2,N}$	$\boldsymbol{F}_N$	8	9	9	9	9	10	9	9	9
$B_{2,N}$	$\boldsymbol{F}_N^w$	8	9	9	9	9	8	10	9	9
$B_{2,N}$	$\boldsymbol{C}_N^{II}$	8	8	8	8	9	10	10	9	9
$B_{2,N}$	$\boldsymbol{S}_N^{II}$	8	10	10	10	9	8	9	9	9
$B_{3,N}$	$\boldsymbol{F}_N$	8	10	10	10	10	9	9	11	11
$B_{3,N}$	$\boldsymbol{F}_N^w$	8	10	9	9	9	10	10	9	9
$B_{3,N}$	$\boldsymbol{C}_N^{II}$	8	9	9	9	9	8	9	10	10
$B_{3,N}$	$\boldsymbol{S}_N^{II}$	8	11	10	10	10	10	9	9	10

TABLE 4
$\varphi(s,t) = s^2 + t^2 + s^2 t^2 \quad (s,t \in [-\pi, \pi))$.

$K_{N,N}$	$\boldsymbol{O}_N$	8	16	32	64	128	256	512
φ	$\boldsymbol{S}_N^{II}$	8	9	9	10	10	10	10
$B_{1,N,N}$	$\boldsymbol{S}_N^{II}$	10	12	14	16	20	26	36
$B_{2,N,N}$	$\boldsymbol{S}_N^{II}$	10	10	11	11	11	11	11
$B_{3,N,N}$	$\boldsymbol{S}_N^{II}$	10	10	11	11	11	11	11

Our next example in Table 3 confirms our theoretical results for the function

(iii) $f(x) = (x^2 - 1)^2$

with zeros of order 2 at $x = \pm 1$.

Finally, let us turn to BTTB matrices $\boldsymbol{A}_{N,N}$. In our examples, the matrices $\boldsymbol{A}_{N,N}$ are generated by the functions

(iv) (see [4]): $\varphi(s,t) = s^2 + t^2 + s^2\, t^2 \quad (s,t \in [-\pi, \pi))$.

(v) (see [30, 31]): $\varphi(s,t) = s^2\, t^4 \quad (s,t \in [-\pi, \pi))$.

(vi) (see [30, 31]): $\varphi(s,t) = (s^2 + t^2)^2 \quad (s,t \in [-\pi, \pi))$.

These matrices are ill-conditioned and the CG method without preconditioning, with Strang-type–preconditioning or with optimal trigonometric preconditioning, converges very slow (see [30, 33, 4]). Our preconditioning (5.2) leads to the number of iterations in Tables 4–6. Here $B_{k,N,N}(x,y) := B_{k,N}(x)\, B_{k,N}(y)$.

In [34], we proved that the number of iteration steps of PCG is independent of N in Example (iv) and we explained the convergence behavior of PCG for the other examples. To our knowledge, at the present time there does not exist a faster PCG

TABLE 5
$\varphi(s,t) = s^2\, t^4 \quad (s,t \in [-\pi,\pi))$.

$K_{N,N}$	$\boldsymbol{O}_N$	8	16	32	64	128	256	512
φ	$\boldsymbol{S}_N^{II}$	13	16	22	29	36	43	52
$B_{1,N,N}$	$\boldsymbol{S}_N^{II}$	18	67	184	631	2363	> 3000	> 3000
$B_{2,N,N}$	$\boldsymbol{S}_N^{II}$	16	29	39	56	77	106	158
$B_{3,N,N}$	$\boldsymbol{S}_N^{II}$	17	29	34	48	63	79	91

TABLE 6
$\varphi(s,t) = (s^2+t^2)^2 \quad (s,t \in [-\pi,\pi))$.

$K_{N,N}$	$\boldsymbol{O}_N$	8	16	32	64	128	256	512
φ	$\boldsymbol{S}_N^{II}$	9	12	14	19	25	35	49
$B_{1,N,N}$	$\boldsymbol{S}_N^{II}$	10	19	31	63	144	381	1413
$B_{2,N,N}$	$\boldsymbol{S}_N^{II}$	10	13	15	18	26	39	62
$B_{3,N,N}$	$\boldsymbol{S}_N^{II}$	10	14	15	18	25	37	48

method if the generating function φ is unknown.

Note that by [42, 43] any multilevel preconditioner is not optimal in the sense that a cluster cannot be proper [45].

Summary. We suggested new positive definite w-circulant preconditioners for sequences of Toeplitz systems with polynomial increasing condition numbers. The construction of our preconditioners is based on the convolution of the generating function with positive reproducing kernels and, by working in the Fourier domain, does not require the explicit knowledge of the generating function. As our main result we proved that the quality of the preconditioner depends on a "moment property" of the corresponding kernel which is related to the order of the zeros of the generating function. This explains, e.g., why optimal circulant preconditioners arising from convolutions with the Fejér kernels fail to be good preconditioners if the generating function has zeros of order ≥ 2.

REFERENCES

[1] O. AXELSSON, *Iterative Solution Methods*, Cambridge University Press, Cambridge, 1996.
[2] G. BASZENSKI AND M. TASCHE, *Fast polynomial multiplication and convolution related to the discrete cosine transform*, Linear Algebra Appl., 252 (1997), pp. 1–25.
[3] F. DI BENEDETTO, *Analysis of preconditioning techniques for ill-conditioned Toeplitz matrices*, SIAM J. Sci. Comput., 16 (1995), pp. 682–697.
[4] F. DI BENEDETTO, *Preconditioning of block Toeplitz matrices by sine transforms*, SIAM J. Sci. Comput., 18 (1997), pp. 499–515.
[5] F. DI BENEDETTO, G. FIORENTINO, AND S. SERRA, *C.G. preconditioning for Toeplitz matrices*, Comput. Math. Appl., 25 (1993), pp. 35–45.
[6] F. DI BENEDETTO AND S. SERRA CAPIZZANO, *A unifying approach to abstract matrix algebra preconditioning*, Numer. Math., 82 (1999), pp. 57–90.
[7] D. BINI AND M. CAPOVANI, *Spectral and computational properties of band symmetric Toeplitz matrices*, Linear Algebra Appl., 52/53 (1983), pp. 99–126.

[8] A. Böttcher and S. M. Grudsky, *Toeplitz band matrices with exponentially growing condition numbers*, Electron. J. Linear Algebra, 5 (1999), pp. 104–125.
[9] C. de Boor, *Splinefunktionen*, Birkhäuser-Verlag, Basel, Switzerland, 1990.
[10] R. H. Chan, *Toeplitz preconditioners for Toeplitz systems with nonnegative generating functions*, IMA J. Numer. Anal., 11 (1991), pp. 333–345.
[11] R. H. Chan and M. K. Ng, *Toeplitz preconditioners for Hermitian Toeplitz systems*, Linear Algebra Appl., 190 (1993), pp. 181–208.
[12] R. H. Chan and M. K. Ng, *Conjugate gradient methods of Toeplitz systems*, SIAM Rev., 38 (1996), pp. 427–482.
[13] R. H. Chan, M. K. Ng, and C. K. Wong, *Sine transform based preconditioners for symmetric Toeplitz systems*, Linear Algebra Appl., 232 (1996), pp. 237–259.
[14] R. H. Chan, T. Tso, and H. Sun, *Circulant preconditioners from B-splines*, in Algorithms, Architectures, and Implementations, F. Luk, ed., San Diego, CA, 1997, pp. 338–347.
[15] R. H. Chan and M.-C. Yeung, *Circulant preconditioners constructed from kernels*, SIAM J. Numer. Anal., 29 (1992), pp. 1093–1103.
[16] R. H. Chan, A. M. Yip, and M. K. Ng, *Circulant preconditioners for ill-conditioned Hermitian Toeplitz matrices*, in Proceedings of the International Congress Chin. Math., Beijing, 1998, Chinese University, Hong Kong, 1998, pp. 165–174.
[17] R. H. Chan, A. M. Yip, and M. K. Ng, *The best circulant preconditioners for Hermitian Toeplitz systems*, SIAM J. Numer. Anal., 38 (2000), pp. 876–896.
[18] R. H. Chan, A. M. Yip, and M. K. Ng, *The Best Circulant Preconditioners for Hermitian Toeplitz Systems* II*: The Multiple-Zero Case*, Chinese University, Hong Kong, 1999, preprint.
[19] T. F. Chan, *An optimal circulant preconditioner for Toeplitz systems*, SIAM J. Sci. Statist. Comput., 9 (1988), pp. 766–771.
[20] P. Davis, *Circulant Matrices*, John Wiley and Sons, New York, 1979.
[21] R. A. DeVore, *The Approximation of Continuous Functions by Positive Linear Operators*, Lecture Notes in Math. 293, Springer, Berlin, New York, 1972.
[22] R. A. DeVore and G. G. Lorentz, *Constructive Approximation*, Springer, Berlin, 1993.
[23] G. Fiorentino and S. Serra, *Multigrid methods for Toeplitz matrices*, Calcolo, 28 (1991), pp. 283–305.
[24] R. A. Horn and C. R. Johnson, *Matrix Analysis*, Cambridge University Press, Cambridge, MA, 1985.
[25] T. Huckle, *Iterative methods for ill–conditioned Toeplitz matrices*, in Proceedings of the Cortona Workshop on Toeplitz Matrices, Cortona, Italy, 1996.
[26] T. Huckle, *Iterative Methods for Toeplitz–Like Matrices*, Report SCCM-94-05, Stanford University, Stanford, CA, 1994.
[27] X.-Q. Jin, *A note on construction of circulant preconditioners from kernels*, Appl. Math. Comput., 83 (1997), pp. 3–12.
[28] M. Kac, W. Murdock, and G. Szegö, *On the eigenvalues of certain Hermitian forms*, J. Ration. Mech. Anal., 2 (1953), pp. 767–800.
[29] P. P. Korovkin, *Linear Operators and Approximation Theory*, Hindustan Publishing, Delhi, India, 1960.
[30] M. K. Ng, *Band preconditioners for block–Toeplitz–Toeplitz–block–systems*, Linear Algebra Appl., 259 (1997), pp. 307–327.
[31] D. Potts and G. Steidl, *Preconditioners for ill–conditioned Toeplitz matrices*, BIT, 39 (1999), pp. 513–533.
[32] D. Potts and G. Steidl, *Optimal trigonometric preconditioners for nonsymmetric Toeplitz systems*, Linear Algebra Appl., 281 (1998), pp. 265–292.
[33] D. Potts, G. Steidl, and M. Tasche, *Trigonometric preconditioners for block Toeplitz systems*, in Multivariate Approximation and Splines, G. Nürnberger, J. W. Schmidt, and G. Walz, eds., Birkhäuser-Verlag, Basel, Switzerland, 1997, pp. 219–234.
[34] D. Potts and G. Steidl, *Preconditioning of Hermitian Block–Toeplitz–Toeplitz–Block Matrices by Level–1 Preconditioners*, Medical University of Lübeck, Lübeck, Germany, 1999, preprint.
[35] S. Serra, *Preconditioning strategies for asymptotically ill–conditioned block Toeplitz systems*, BIT, 34 (1994), pp. 579–594.
[36] S. Serra, *Optimal, quasi–optimal and superlinear band–Toeplitz preconditioners for asymptotically ill–conditioned positive definite Toeplitz systems*, Math. Comp., 66 (1997), pp. 651–665.
[37] S. Serra, *An ergodic theorem for classes of preconditioned matrices*, Linear Algebra Appl., 282 (1998), pp. 169–183.

[38] S. Serra, *On the extreme eigenvalues of Hermitian (block) Toeplitz matrices*, Linear Algebra Appl., 270 (1998), pp. 109–129.
[39] S. Serra Capizzano, *Korovkin theorems and linear positive gram matrix algebra approximations of Toeplitz matrices*, Linear Algebra Appl., 284 (1998), pp. 307–334.
[40] S. Serra, *How to choose the best iterative strategy for symmetric Toeplitz systems*, SIAM J. Numer. Anal., 36 (1999), pp. 1078–1103.
[41] S. Serra Capizzano, *Toeplitz preconditioners constructed from linear approximation processes*, SIAM J. Matrix Anal., 20 (1999), pp. 446–465.
[42] S. Serra Capizzano and E. E. Tyrtyshnikov, *Any circulant-like preconditioner for multilevel matrices is not superlinear*, SIAM J. Matrix Anal., 21 (1999), pp. 431–439.
[43] S. Serra Capizzano and E. E. Tyrtyshnikov, *How to Prove that a Preconditioner Can Not be Optimal*, preprint, 1998.
[44] G. Steidl and M. Tasche, *A polynomial approach to fast algorithms for discrete Fourier–cosine and Fourier–sine transforms*, Math. Comp., 56 (1991), pp. 281–296.
[45] E. E. Tyrtyshnikov, *Circulant preconditioners with unbounded inverses*, Linear Algebra Appl., 216 (1995), pp. 1–23.
[46] Z. Wang, *Fast algorithms for the discrete W transform and for the discrete Fourier transform*, IEEE Trans. Acoust. Speech Signal Process., 32 (1984), pp. 803–816.

SIAM J. SCI. COMPUT.
Vol. 22, No. 5, pp. 1762–1771
© 2001 Society for Industrial and Applied Mathematics

A NOTE ON PARALLEL MATRIX INVERSION*

ENRIQUE S. QUINTANA†, GREGORIO QUINTANA†, XIAOBAI SUN‡, AND ROBERT VAN DE GEIJN§

Dedicated to G. W. Stewart on the occasion of his 60th birthday

Abstract. We present one-sweep parallel algorithms for the inversion of general and symmetric positive definite matrices. The algorithms feature simple programming and performance optimization while maintaining the same arithmetic cost and numerical properties of conventional inversion algorithms. Our experiments on a Cray T3E-600 and a Beowulf cluster demonstrate high performance of implementations for distributed memory parallel computers.

Key words. matrix inversion, Gauss–Jordan elimination, parallel computers

AMS subject classifications. 65F05, 65Y05

PII. S1064827598345679

1. Introduction.

Despite the inexpedience of matrix inverse, as Higham summarizes in [13], "there are situations in which a matrix inverse must be computed." Examples arise in statistics [4, section 7.5], [15, section 2.3], [16, p. 342], [3]; in numerical integrations in superconductivity computations [12]; and in stable subspace computation in control theory [17]. The recent years have seen increasing interest in parallel solutions of large-scale applications [3, 7, 9]. Inversion algorithms for general full nonsingular matrices are mostly based on the availability of a complete LU factorization. The algorithm used by LINPACK [8] and LAPACK [2] proceeds as follows.

(1) LU factorization with partial pivoting, $PA = LU$, where P is a permutation matrix, and U, $L \in \Re^{n \times n}$ are upper triangular and unit lower triangular matrices, respectively.
(2) Triangular inversion of U (forward substitution).
(3) Triangular (system) solve for X: $XL = U^{-1}$ (backward substitution).
(4) Back permutation of columns, $A^{-1} = XP$.

The algorithm allows the inverse to be computed *in-place*, that is, the computed inverse overwrites the input matrix to be inverted. It sweeps three times across the array that houses the involved matrices for LU factorization, triangular inversion, and triangular solve. The algorithm is effective on uniprocessor computers and shared memory parallel computers. A parallel version of the algorithm is implemented in ScaLAPACK [6]. We present in this paper our study of inversion algorithms via Gauss–Jordan elimination (GJE) for general square matrices and a related algorithm for symmetric positive definite (SPD) matrices. Specifically, we show that these algorithms are more suitable for parallel computers with physically distributed

*Received by the editors October 9, 1998; accepted for publication (in revised form) August 16, 2000; published electronically January 16, 2001. This project was supported in part by the PRISM project (ARPA grant P-95006) and a grant from the Intel Research Council.
http://www.siam.org/journals/sisc/22-5/34567.html
† Departamento de Informática, Universidad Jaume I, 12080 Castellón, Spain (quintana@inf.uji.es, gquintan@inf.uji.es).
‡Department of Computer Science, Duke University, Durham, NC 27708-0129 (xiaobai@cs.duke.edu).
§Department of Computer Sciences, The University of Texas, Austin, TX 78712 (rdvg@cs.utexas.edu).

```
% Input  : n x n nonsingular matrix A.
% Output : matrix A overwritten by its inverse
                                          % pivoting
                                          ipivs = [1:n];
for k = 1 : n
                                          [abs_ipiv,ipiv] = max(abs(A(k:n,k)));
                                          ipiv = ipiv+k-1;
                                          [A(k,:),A(ipiv,:)]     =
                                                   swap(A(ipiv,:),A(k,:));
                                          [ipivs(k),ipivs(ipiv)] =
                                                   swap(ipivs(ipiv),ipivs(k));
  Akk    = A(k,k);
  % Jordan transformation
  A(:,k) = -[A(1:k-1,k); 0; A(k+1:n,k)]/Akk;
  A      = A + A(:,k) * A(k,:); \\
  A(k,:) =  [A(k,1:k-1), 1, A(k,k+1:n)]/Akk;
end
                                        A(:,ipivs) = A;
```

FIG. 1. MATLAB *code for matrix inversion via GJE. By also executing the steps on the right, pivoting is added.*

memory.

Matrix inversion using GJE is, in essence, a reordering of the computation performed by matrix inversion methods using Gaussian elimination (LU factorization) and hence requires the same arithmetic cost. Many classic references to inversion methods can be found in [13, 14]. Nevertheless, computation arrangements for matrix inversion via GJE (instead of a system solve with multiple right-hand sides) are rarely presented in the literature. A nonblocked parallel version of the in-place GJE-based inversion algorithm presented later is given in [10]. An in-place procedure for inversion of positive definite matrices is given by Bauer and Reinsch [5].

In Figure 1 we describe in MATLAB language a LEVEL-2 basic linear algebra subprogram (BLAS), in-place inversion algorithm via GJE for a general square matrix. The fact that the sweeps over the intermediate triangular matrices L and U are computationally eluded has multiple advantages in parallel computations, as we will describe in sections 5 and 6.

The rest of the paper is organized as follows. In section 2 we present an inversion algorithm via GJE with partial pivoting. The relation of this algorithm with traditional multistage algorithms is given in section 3. The approach is extended to SPD matrices in section 4. In section 5 we discuss parallel implementation issues. In section 6 we present performance results for our parallel implementations of the algorithm. Concluding remarks can be found in the final section.

2. An inversion algorithm via Gauss–Jordan elimination. Recall how to invert an $n \times n$ matrix A by creating an augmented system $\left(\begin{array}{c|c} A & B \end{array}\right)$ with $B = I_n$, the $n \times n$ identity matrix, and performing Gauss–Jordan elimination on this augmented system:

for $k = 1, n$

Partition $\left(\begin{array}{c|c} A & B \end{array}\right) \rightarrow \left(\begin{array}{c|c|c||c|c|c} I_{k-1} & a_{01} & A_{02} & B_{00} & 0 & 0 \\ \hline 0 & \alpha_{11} & a_{12}^T & b_{10}^T & 1 & 0 \\ \hline 0 & a_{21} & A_{22} & B_{20} & 0 & I_{n-k} \end{array}\right)$

Update $\left(\begin{array}{c|c||c} A & B \end{array}\right) \leftarrow$

$$\left(\begin{array}{c|c|c||c|c|c} I_{k-1} & 0 & A_{02}+u_{01}a_{12}^T & B_{00}+u_{01}b_{10}^T & u_{01}=-a_{01}/\alpha_{11} & 0 \\ \hline 0 & 1 & a_{12}^T/\alpha_{11} & b_{10}^T/\alpha_{11} & 1/\alpha_{11} & 0 \\ \hline 0 & 0 & A_{22}+l_{21}a_{12}^T & B_{20}+l_{21}b_{10}^T & l_{21}=-a_{21}/\alpha_{11} & I_{n-k} \end{array}\right)$$

endfor

Magically, upon completion B has been overwritten by A^{-1}.

2.1. An in-place algorithm. After k steps of the above algorithm, it suffices to store only the first k columns of B and the last $n-k$ columns of A. Thus, an algorithm that overwrites A with A^{-1} is given by:

Partition $A = \left(\begin{array}{c||c} A_L & A_R \end{array}\right)$ and $B = \left(\begin{array}{c||c} B_L & B_R \end{array}\right)$, where initially $A_R = A$ and $B_R = I_n$.

for $k = 1, n$

Partition $\left(\begin{array}{c||c} B_L & A_R \end{array}\right) \rightarrow \left(\begin{array}{c||c|c} B_{00} & a_{01} & A_{02} \\ \hline b_{10}^T & \alpha_{11} & a_{12}^T \\ \hline B_{20} & a_{21} & A_{22} \end{array}\right)$, where B_{00} is square.

Update $\left(\begin{array}{c||c} B_L & A_R \end{array}\right) \leftarrow$

$$\left(\begin{array}{c|c||c} B_{00}+u_{01}b_{10}^T & u_{01}=-a_{01}/\alpha_{11} & A_{02}+u_{01}a_{12}^T \\ \hline b_{10}^T/\alpha_{11} & 1/\alpha_{11} & a_{12}^T/\alpha_{11} \\ \hline B_{20}+l_{21}b_{10}^T & l_{21}=-a_{21}/\alpha_{11} & A_{22}+l_{21}a_{12}^T \end{array}\right)$$

endfor

Upon entering the loop $\left(\begin{array}{c||c} B_L & A_R \end{array}\right)$ equals A and after its completion it equals A^{-1}.

The above update can also be accomplished by the *rank*-1 *update*

$$\left(\begin{array}{c||c} B_L & A_R \end{array}\right) \leftarrow \left(\begin{array}{c|c||c} B_{00} & 0 & A_{02} \\ \hline 0 & 0 & 0 \\ \hline B_{20} & 0 & A_{22} \end{array}\right) + \frac{1}{\alpha_{11}} \left(\begin{array}{c} -a_{01} \\ \hline 1 \\ \hline -a_{21} \end{array}\right) \left(\begin{array}{c|c||c} b_{10}^T & 1 & a_{12}^T \end{array}\right).$$

The entire computation is in the rank-1 update at a cost of $2n^2$ floating point operations (flops) per iteration for a total cost of $2n^3$ flops. A compact MATLAB code is given in Figure 1.

2.2. A blocked algorithm. A blocked variant of the above algorithm can be developed in a straightforward fashion:

Partition $A = \left(\begin{array}{c||c} A_L & A_R \end{array}\right)$ and $B = \left(\begin{array}{c||c} B_L & B_R \end{array}\right)$, where initially $A_R = A$ and $B_R = I_n$.

for $k = 1, n$ in steps of b

Partition $\left(\begin{array}{c||c} B_L & A_R \end{array}\right) \rightarrow \left(\begin{array}{c||c|c} B_{00} & A_{01} & A_{02} \\ \hline B_{10} & A_{11} & A_{12} \\ \hline B_{20} & A_{21} & A_{22} \end{array}\right)$, where B_{00} is square and A_{11} is $b \times b$.

Update

$$\left(\begin{array}{c||c} B_L & A_R \end{array}\right) \leftarrow \left(\begin{array}{c|c||c} B_{00} & 0 & A_{02} \\ \hline 0 & 0 & 0 \\ \hline B_{20} & 0 & A_{22} \end{array}\right) + \left(\begin{array}{c} -A_{01}A_{11}^{-1} \\ \hline A_{11}^{-1} \\ \hline -A_{21}A_{11}^{-1} \end{array}\right) \left(\begin{array}{c|c||c} B_{10} & I_b & A_{12} \end{array}\right)$$

endfor

The benefit of the blocked algorithm is that most of the computation is now in a matrix-matrix multiply (rank-k update) which allows high performance to be achieved in a portable fashion.

2.3. Adding pivoting. As for the LU factorization, in order to improve stability for general square nonsingular matrices, it is necessary to include row pivoting in the algorithm. To add pivoting to our nonblocked algorithm, before the update step, the row with the largest absolute value among elements of α_{11} and a_{21} is swapped with the kth row. If the augmented system is taken to equal $\left(\begin{array}{c||c} A & B \end{array}\right)$ with $B = P^T$ instead, we can guarantee that after $k-1$ steps *and after swapping the rows for the current step* the system looks like

$$\left(\begin{array}{c|c|c||c|c|c} I_{k-1} & a_{01} & A_{02} & B_{00} & 0 & 0 \\ \hline 0 & \alpha_{11} & a_{12}^T & b_{10}^T & 1 & 0 \\ \hline 0 & a_{21} & A_{22} & B_{20} & 0 & P_{n-k} \end{array}\right),$$

where P_{n-k} is some permutation matrix determined by future iterations of the loop. The usual update step then proceeds. This time, $A^{-1}P^T$ is computed, so that it is necessary to complete the process by permuting the columns of the result: $A^{-1} = (A^{-1}P^T)P$. Figure 1 shows how pivoting can be added to the MATLAB code.

Likewise, pivoting is added to the above blocked algorithm by changing the body of the loop to

Partition (as before)

Compute $P\left(\begin{array}{c} A_{11} \\ \hline A_{21} \end{array}\right) \rightarrow \left(\begin{array}{c} L_{11} \\ \hline L_{21} \end{array}\right) U_{11}$ (LU factorization with pivoting)

Swap $\left(\begin{array}{c|c||c} B_{10} & A_{11} & A_{12} \\ \hline B_{20} & A_{21} & A_{22} \end{array}\right) \leftarrow P \left(\begin{array}{c|c||c} B_{10} & A_{11} & A_{12} \\ \hline B_{20} & A_{21} & A_{22} \end{array}\right)$

Update (as before)

There are a number of ways to compute the update. As for the unblocked algorithm, a final permutation of the columns is required.

The update in the blocked algorithm is equivalent to the update

$$\begin{aligned} \left(\begin{array}{c||c} B_L & A_R \end{array}\right) \leftarrow & \left(\begin{array}{c|c||c} B_{00} & 0 & A_{02} \\ \hline 0 & 0 & 0 \\ \hline B_{20} & 0 & A_{22} \end{array}\right) \\ & + \left(\begin{array}{c} -A_{01}U_{11}^{-1} \\ \hline U_{11}^{-1} \\ \hline -L_{21} \end{array}\right) \left(\begin{array}{c|c||c} L_{11}^{-1}B_{10} & L_{11}^{-1} & L_{11}^{-1}A_{12} \end{array}\right), \end{aligned} \tag{2.1}$$

where L_{11}, L_{21}, and U_{11} are as computed by the LU factorization with partial pivoting of the corresponding part of A. Here $-A_{01}U_{11}^{-1}$, $L_{11}^{-1}B_{10}$, and $L_{11}^{-1}A_{21}$ are computed using triangular solves rather than matrix inversion. Forming $A_{11}^{-1} = U_{11}^{-1}L_{11}^{-1}$ can be accomplished as in the traditional matrix inversion algorithm. Computing the update in this fashion has the benefit that since explicit use of A_{11}^{-1} is avoided, stability of the algorithm should be similar to that exhibited by the traditional approach. We elaborate on this next.

3. Relation with multistage algorithms. The explicit use of A_{11}^{-1} in the blocked algorithm with pivoting should raise eyebrows. In [18], we show that the

numerical stability of the overall algorithm is only mildly affected by the use of the inverse, with stability deteriorating as a function of the algorithmic block size b. By using instead the variant that updates the matrix given in (2.1) these instabilities can be avoided.

Consider yet another alternative for computing A^{-1}:

(1) LU factorization with partial pivoting, $PA = LU$.
(2) Triangular inversion of U.
(3) Triangular inversion of L.
(4) Multiplication of triangular matrices $X = U^{-1}L^{-1}$.
(5) Back permutation of columns, $A^{-1} = XP$.

For simplicity we ignore pivoting and give blocked algorithms for the four remaining stages in Figure 2. If one picks the block size b to be equal at each step of the algorithms, it becomes clear that all four algorithms can be implemented by merging the loops. Critical to this observation is the fact that the different algorithms update four different quadrants of the matrix. Once the four loops have been merged into one loop, a number of intermediate computations can be eliminated leaving only the computations in the boxes in Figure 2. In Figure 3 we show that these computations are exactly the updates of the nine submatrices as given in (2.1). The same conclusion holds when pivoting is added.

We conclude that the algorithm based on (2.1) is equivalent to this multistage algorithm and thus shares its stability properties. In [13] it is shown that the traditional approach and this alternative multistage algorithm are essentially equally stable.

4. SPD matrix inversion. The techniques described above can be used to derive a blocked algorithm for SPD matrix inversion.

Let A be SPD. Then its inverse can be computed by first computing its Cholesky factor L, where $A = LL^T$, after which $A^{-1} = L^{-T}L^{-1}$. Naturally, this can be accomplished in three stages: (1) Compute the Cholesky factorization, (2) invert the lower triangular matrix L, and (3) multiply $A^{-1} = L^{-1^T}L^{-1}$. By putting all these stages together, a one-sweep algorithm can be derived: Partition the current matrix $A^{(k)}$, the original matrix A, and the factor L like

$$A^{(k)} = \left(\begin{array}{c||c} A_{TL}^{(k)} & \star \\ \hline\hline A_{BL}^{(k)} & A_{BR}^{(k)} \end{array}\right), \quad A = \left(\begin{array}{c||c} A_{TL} & \star \\ \hline\hline A_{BL} & A_{BR} \end{array}\right), \text{ and } L = \left(\begin{array}{c||c} L_{TL} & 0 \\ \hline\hline L_{BL} & L_{BR} \end{array}\right),$$

where the $\star$ indicates the symmetric part not to be updated and $A_{TL}^{(k)}$, A_{TL}, and L_{TL} are $k \times k$. Repartition

$$A^{(k)} = \left(\begin{array}{c||c} A_{TL}^{(k)} & \star \\ \hline\hline A_{BL}^{(k)} & A_{BR}^{(k)} \end{array}\right) = \left(\begin{array}{c||c|c} A_{00} & \star & \star \\ \hline\hline A_{10} & A_{11} & \star \\ \hline A_{20} & A_{21} & A_{22} \end{array}\right),$$

and consider the update

$$A^{(k+b)} = \left(\begin{array}{c||c} A_{TL}^{(k+b)} & \star \\ \hline\hline A_{BL}^{(k+b)} & A_{BR}^{(k+b)} \end{array}\right) = \left(\begin{array}{c|c||c} A_{00} + \hat{L}_{10}^T\hat{L}_{10} & \star & \star \\ \hline\hline L_{11}^{-T}\hat{L}_{10} & L_{11}^{-T}L_{11}^{-1} & \star \\ \hline A_{20} - L_{21}\hat{L}_{10} & L_{21} & A_{22} - L_{21}L_{21}^T \end{array}\right),$$

where L_{11} equals the Cholesky factor of A_{11}, $L_{21} = A_{21}L_{11}^{-T}$ and $\hat{L}_{10} = L_{11}^{-1}A_{10}$. This

Factor $A \leftarrow L\backslash U = \mathrm{LU}(A)$**:**

Partition $A = \left(\begin{array}{c||c} A_{TL} & A_{TR} \\ \hline\hline A_{BL} & A_{BR} \end{array}\right)$
where A_{TL} is 0×0
do until A_{BR} is 0×0
Determine block size b
View
$\left(\begin{array}{c||c} A_{TL} & A_{TR} \\ \hline\hline A_{BL} & A_{BR} \end{array}\right) = \left(\begin{array}{c||c|c} L\backslash U_{00} & U_{01} & U_{02} \\ \hline\hline L_{10} & A_{11} & A_{12} \\ \hline L_{20} & A_{21} & A_{22} \end{array}\right)$
where A_{11} is $b \times b$

$A_{11} \leftarrow L\backslash U_{11} = \mathrm{LU}(A_{11})$
$A_{21} \leftarrow L_{21} = A_{21}U_{11}^{-1}$
$A_{12} \leftarrow U_{12} = L_{11}^{-1}A_{12}$
$A_{22} \leftarrow \hat{A}_{22} = A_{22} - L_{21}U_{12}$
$(= A_{22} - (A_{21}U_{11}^{-1})(L_{11}^{-1}A_{12}))$
Continue with
$\left(\begin{array}{c||c} A_{TL} & A_{TR} \\ \hline\hline A_{BL} & A_{BR} \end{array}\right) = \left(\begin{array}{c||c|c} L\backslash U_{00} & U_{01} & U_{02} \\ \hline\hline L_{10} & \mathbf{L\backslash U_{11}} & \mathbf{U_{12}} \\ \hline L_{20} & \mathbf{L_{21}} & \mathbf{\hat{A}_{22}} \end{array}\right)$
enddo

Invert $U \leftarrow \hat{U} = U^{-1}$**:**

Partition $U = \left(\begin{array}{c||c} U_{TL} & U_{TR} \\ \hline\hline 0 & U_{BR} \end{array}\right)$
where U_{TL} is 0×0
do until U_{BR} is 0×0
Determine block size b
View
$\left(\begin{array}{c||c} U_{TL} & U_{TR} \\ \hline\hline 0 & U_{BR} \end{array}\right) = \left(\begin{array}{c||c|c} \hat{U}_{00} & U_{01} & U_{02} \\ \hline\hline 0 & U_{11} & U_{12} \\ \hline 0 & 0 & U_{22} \end{array}\right)$
where U_{11} is $b \times b$

$U_{11} \leftarrow \hat{U}_{11} = U_{11}^{-1}$
$U_{01} \leftarrow \hat{U}_{01} = -U_{01}U_{11}^{-1}$
$U_{02} \leftarrow U_{02}^{\star} = U_{02} + \hat{U}_{01}U_{12}$
$(= U_{02} + (U_{01}U_{11}^{-1})(L_{11}^{-1}A_{12}))$
$U_{12} \leftarrow U_{12}^{\star} = U_{11}^{-1}U_{12}$
$(= U_{11}^{-1}(L_{11}^{-1}A_{12}))$
Continue with
$\left(\begin{array}{c||c} U_{TL} & U_{TR} \\ \hline\hline 0 & U_{BR} \end{array}\right) = \left(\begin{array}{c||c|c} \hat{U}_{00} & \mathbf{\hat{U}_{01}} & \mathbf{U_{02}^{\star}} \\ \hline\hline 0 & \mathbf{\hat{U}_{11}} & \mathbf{U_{12}^{\star}} \\ \hline 0 & 0 & U_{22} \end{array}\right)$
enddo

Invert $L \leftarrow \hat{L} = L^{-1}$**:**

Partition $L = \left(\begin{array}{c||c} L_{TL} & 0 \\ \hline\hline L_{BL} & L_{BR} \end{array}\right)$
where L_{TL} is 0×0
do until L_{BR} is 0×0
Determine block size b
View
$\left(\begin{array}{c||c} L_{TL} & 0 \\ \hline\hline L_{BL} & L_{BR} \end{array}\right) = \left(\begin{array}{c||c|c} \hat{L}_{00} & 0 & 0 \\ \hline\hline L_{10} & L_{11} & 0 \\ \hline L_{20} & L_{21} & L_{22} \end{array}\right)$
where L_{11} is $b \times b$

$L_{11} \leftarrow \hat{L}_{11} = L_{11}^{-1}$
$L_{10} \leftarrow \hat{L}_{10} = L_{11}^{-1}L_{10}$
$L_{20} \leftarrow L_{20}^{\star} = L_{20} - L_{21}\hat{L}_{10}$
$(= L_{20} - (A_{21}U_{11}^{-1})(L_{11}^{-1}L_{10}))$
$L_{21} \leftarrow L_{21}^{\star} = -L_{21}L_{11}^{-1}$
$(= -(A_{21}U_{11}^{-1})L_{11}^{-1})$
Continue with
$\left(\begin{array}{c||c} L_{TL} & 0 \\ \hline\hline L_{BL} & L_{BR} \end{array}\right) = \left(\begin{array}{c||c|c} \hat{L}_{00} & 0 & 0 \\ \hline\hline \mathbf{\hat{L}_{10}} & \mathbf{\hat{L}_{11}} & 0 \\ \hline \mathbf{L_{20}^{\star}} & \mathbf{L_{21}^{\star}} & L_{22} \end{array}\right)$
enddo

Compute $\hat{A} \leftarrow \hat{U}\hat{L} = U^{-1}L^{-1} = A^{-1}$**:**

Partition $\hat{L}\backslash\hat{U} = \left(\begin{array}{c||c} A_{TL} & A_{TR} \\ \hline\hline A_{BL} & A_{BR} \end{array}\right)$
where A_{TL} is 0×0
do until A_{BR} is 0×0
Determine block size b
View
$\left(\begin{array}{c||c} A_{TL} & A_{TR} \\ \hline\hline A_{BL} & A_{BR} \end{array}\right) = \left(\begin{array}{c||c|c} \hat{A}_{00} & \hat{U}_{01} & \hat{U}_{02} \\ \hline\hline \hat{L}_{10} & \hat{L}\backslash\hat{U}_{11} & \hat{U}_{12} \\ \hline \hat{L}_{20} & \hat{L}_{21} & \hat{L}\backslash\hat{U}_{22} \end{array}\right)$
where $\hat{L}\backslash\hat{U}_{11}$ is $b \times b$

$\hat{A}_{00} \leftarrow \hat{A}_{00}^{\star} = \hat{A}_{00} + \hat{U}_{01}\hat{L}_{10}$
$(= \hat{A}_{00} - (\hat{U}_{01}U_{11}^{-1})(L_{11}^{-1}L_{10}))$
$\hat{L}_{10} \leftarrow \hat{A}_{10}^{\star} = \hat{U}_{11}\hat{L}_{10}$
$(= U_{11}^{-1}(L_{11}^{-1}L_{10}))$
$\hat{U}_{01} \leftarrow \hat{A}_{01}^{\star} = \hat{U}_{01}\hat{L}_{11}$
$(= -(U_{01}U_{11}^{-1})L_{11}^{-1})$
$\hat{L}\backslash\hat{U}_{11} \leftarrow \hat{A}_{11}^{\star} = \hat{U}_{11}\hat{L}_{11}$
$(= U_{11}^{-1}L_{11}^{-1})$
Continue with
$\left(\begin{array}{c||c} A_{TL} & A_{TR} \\ \hline\hline A_{BL} & A_{BR} \end{array}\right) = \left(\begin{array}{c||c|c} \mathbf{\hat{A}_{00}^{\star}} & \mathbf{\hat{A}_{01}^{\star}} & \hat{U}_{02} \\ \hline\hline \mathbf{\hat{A}_{10}^{\star}} & \mathbf{\hat{A}_{11}^{\star}} & \hat{U}_{12} \\ \hline \hat{L}_{20} & \hat{L}_{21} & \hat{L}\backslash\hat{U}_{22} \end{array}\right)$
enddo

FIG. 2. *Blocked algorithms for a four-stage algorithm to compute* A^{-1}.

Partition $A = \left(\begin{array}{c\|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right)$,
where A_{TL} is 0×0
do until A_{BR} is 0×0
Determine block size b
View
$$\left(\begin{array}{c\|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right) = \left(\begin{array}{c\|c|c} A_{00} & A_{01} & A_{02} \\ \hline A_{10} & A_{11} & A_{12} \\ \hline A_{20} & A_{21} & A_{22} \end{array}\right),$$
where A_{11} is $b \times b$

$A_{11} \rightarrow L_{11}U_{11}$

Continue with
$$\left(\begin{array}{c\|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right) =$$
$$\left(\begin{array}{c|c\|c} A_{00} - (A_{01}U_{11}^{-1})(L_{11}^{-1}A_{10}) & -(A_{01}U_{11}^{-1})L_{11}^{-1} & A_{02} + (U_{01}U_{11}^{-1})(L_{11}^{-1}A_{12}) \\ \hline U_{11}^{-1}(L_{11}^{-1}A_{10}) & A_{11}^{-1} & U_{11}^{-1}(L_{11}^{-1}A_{12}) \\ \hline A_{20} - (A_{21}U_{11}^{-1})(L_{11}^{-1}A_{10}) & -(A_{21}U_{11}^{-1})L_{11}^{-1} & A_{22} - (A_{21}U_{11}^{-1})(L_{11}^{-1}A_{12}) \end{array}\right)$$
= Eqn. (1)
enddo

FIG. 3. *Blocked algorithm attained by putting all four stages together.*

update maintains the state

$$A^{(k)} = \left(\begin{array}{c\|c} A_{TL}^{(k)} & \star \\ \hline A_{BL}^{(k)} & A_{BR}^{(k)} \end{array}\right) = \left(\begin{array}{c\|c} L_{TL}^{-T}L_{TL}^{-1} & \star \\ \hline L_{BL}L_{TL}^{-1} & A_{BR} - L_{BL}L_{BL}^{T} \end{array}\right).$$

Note that once $k = n$, $A^{(k)} = A^{-1}$, which is the desired result.

5. Parallel implementation. Sadly, the above described algorithms yield no real performance benefits on a sequential computer. The primary reason is that on sequential architectures the individual stages of the traditional algorithm lend themselves well to optimization. On distributed memory parallel architectures, the story is different.

All three stages of the traditional approach suffer from load-imbalance when executed on distributed memory parallel architectures: During the LU factorization, the active part of the matrix shrinks from an $n \times n$ matrix to a 1×1 matrix as the computation unfolds. For the inversion of U, initially the active matrix is 1×1, eventually expanding to $n \times n$. In addition, the computation only acts on the upper triangular portion of the matrix. During the computation of the solution X to $XL = U^{-1}$, the active matrix expands from $1 \times n$ to $n \times n$. While cyclic wrapping of the matrices helps load-balance, there is still a considerable performance hit. Similar issues affect the three stages for inverting an SPD matrix.

The blocked algorithm presented in the previous section does not suffer the same fate: The bulk of the computation is in the rank-k update given in section 2.2 which parallelizes almost perfectly. Indeed, the computation required for the GJE-based inversion algorithm is almost identical to the sequence of rank-k updates used to implement highly efficient parallel matrix-matrix multiplication algorithms [1, 11, 19]. Cyclic wrapping of the matrix is not even necessary for load-balance! Even when the inversion is part of a larger computation and wrapping is desirable, excellent performance can be achieved, as we will show in the next section. For inverting an

SPD matrix, wrapping is still needed since only the lower triangular part of the matrix is updated.

6. Experimental results. We present results for three parallel implementations of algorithms for inversion of a general matrix and one for inversion of an SPD matrix. `SL_IGEP` is part of ScaLAPACK [6] and implements the traditional inversion algorithm via LU factorization; `SL_IGJEP` was coded using ScaLAPACK parallel BLAS kernels (PBLAS). It implements a blocked GJE algorithm with the algorithmic block size limited by the distribution block size. All operations required to update the current column panel are performed within a single column of processors. A second implementation of the blocked GJE algorithm, `PLA_IGJEP`, was coded using PLAPACK [20]. In this implementation the current column panel is redistributed so that all processors participate when the unblocked algorithm is used to update that panel. This results in better load-balance during this stage of the algorithm and simplifies the coding when the algorithmic block size is to be larger than the distribution block size. While `SL_IGJEP` represents a basic implementation of the algorithm, `PLA_IGJEP` includes a number of optimizations, some of which cannot be (easily) implemented using ScaLAPACK. `PLA_ISPD` implements a parallel blocked one-sweep algorithm for the inversion of SPD matrices.

Performance results are given for two different distributed memory architectures using 64-bit real arithmetic, the Cray T3E-600 (300 MHz), and a Beowulf cluster. The Cray T3E-600 consists of DEC Alpha EV5 with 128 Mbytes of RAM each, interconnected via a three-dimensional toroidal interconnect. The Beowulf cluster consists of Intel Pentium-II (300 MHz) processors running on a 66 MHz Intel Atlanta motherboard with 128 Mbytes of RAM each, and interconnected via a 1Gbps Myrinet switch.

We used ScaLAPACK available from Netlib (version 1.6) with the MPIBLACS precompiled for Cray T3E[1] and Linux. The PLAPACK version, implemented using an alpha release of PLAPACK R2.0, is also MPI-based. Optimized sequential BLASs were used to attain high performance locally on each processor. The sequential BLAS routine `gemm` delivers 400–450 Mflop/s (millions of flops per second) on a single processor of the Cray T3E and 175–200 Mflop/s on one Pentium-II (300 MHz) processor. We report performance of the inverse routines measuring Mflop/s per processor using the established operation count of $2n^3$ flops for inversion of a general $n \times n$ matrix and n^3 flops for inversion of an SPD matrix.

In Figure 4, we show that very respectable performance is attained on both architectures using 32 processors. Algorithmic and distribution block sizes of 48 and 32 were used for `SL_IGEP` and `SL_IGJEP` on the T3E and cluster, respectively. As expected, `SL_IGJEP` outperforms `SL_IGEP` since it incurs less communication and exhibits better load-balance. For `PLA_IGJEP` we report performance when algorithmic and distribution block sizes are both equal to those used for ScaLAPACK (`PLA_IGJEP_48_48` and `PLA_IGJEP_32_32`). We also show that better performance is attained when a smaller distribution block size and larger algorithmic block size is used (`PLA_IGJEP_24_96` and `PLA_IGJEP_16_64`). The larger algorithmic block size allows local matrix-matrix multiplication to attain higher performance, which explains the asymptotically better performance attained by `PLA_IGJEP_24_96`. The smaller distribution block size yields

[1]The Cray Scientific library provides a version of ScaLAPACK that uses the *shmem* library for communication. For a fair comparison between `PLA_IGJEP` and `SL_IGJEP`, we chose the MPI-based version of ScaLAPACK. Only for small matrices is the shmem-based version of ScaLAPACK noticeably faster than the MPI-based version.

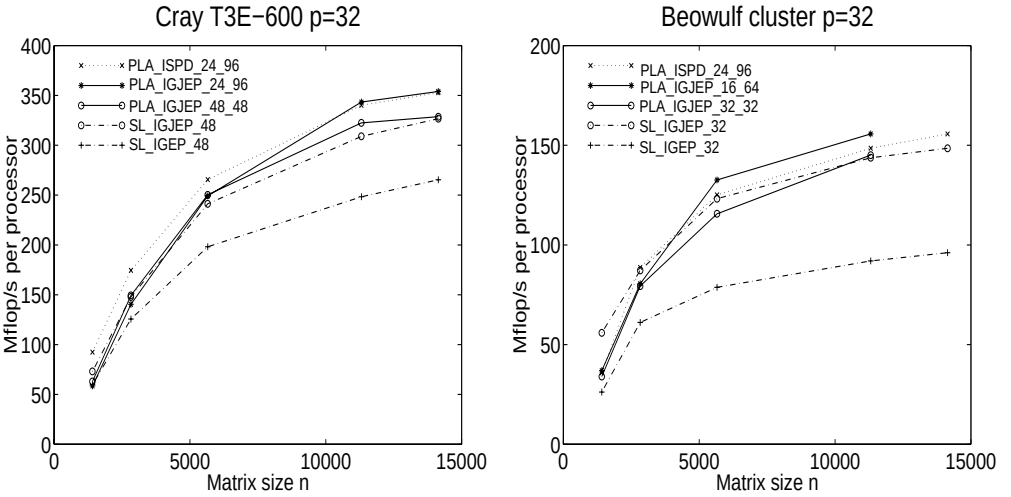


FIG. 4. *Mflop/s per processor on the Cray T3E-600 (left) and the Beowulf cluster (right).*

better load-balance. For smaller problems, the additional cost of redistributing the current column panel results in a reduction in performance relative to the ScaLAPACK implementation. Finally, in Figure 4 (left) we report performance of the SPD inversion routine using an algorithmic block size of 96/64 and distribution block size of 24/16 (`PLA_ISPD_24_96` and `PLA_ISPD_16_64`). In Figure 4 (right) we present similar performance results for the Beowulf cluster.

We do not present scalability results for these algorithms since scalability is clearly at least as good at that of parallel implementations for the individual stages of the traditional algorithm, which themselves are known to have very good scalability properties. For details, see [18].

7. Concluding remarks. We have described matrix inversion via Gauss–Jordan elimination for general matrices and a related algorithm for SPD matrices. The approaches present the same arithmetic cost as the conventional inversion algorithms while maintaining their numerical properties. Our matrix inversion algorithms render simple programming and performance optimization, which are especially appropriate for parallel computers with distributed memory. The experimental results on a Cray T3E-600 and a Beowulf cluster show that very high performance is attained by the parallel Gauss–Jordan inversion algorithm for general matrices and the parallel one-sweep inversion algorithm for SPD matrices.

Implementations are available at http://www.cs.utexas.edu/users/plapack/inverse/.

Acknowledgments. We express our gratitude to Y. J. Wu and N. Higham for their valuable suggestions.

Access to equipment was provided by the National Partnership for Advanced Computational Infrastructure (NPACI), The University of Texas Advanced Computing Center (TACC), and the Earth and Space Sciences (ESS) Project of the NASA HPCC Program.

REFERENCES

[1] R. C. Agarwal, F. Gustavson, and M. Zubair, *A high-performance matrix multiplication algorithm on a distributed memory parallel computer using overlapped communication*, IBM J. Research and Development, 38 (1994), pp. 673–682.

[2] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, and D. Sorensen, *LAPACK Users' Guide*, SIAM, Philadelphia, PA, 1995.

[3] G. Baker, *Implementation of parallel processing to selected problems in satellite geodesy*, Ph.D. thesis, Department of Aerospace Engineering, The University of Texas, Austin, TX, 1997.

[4] Y. Bard, *Nonlinear parameter estimation*, Academic Press, New York, NY, 1974.

[5] F. L. Bauer and C. Reinsch, *Inversion of positive definite matrices by the Gauss-Jordan method*, in Linear Algebra, Handbook for Automatic Computation, Vol. II, J. H. Wilkinson and C. Reinsch, eds., Springer-Verlag, Berlin, 1971, Contribution I/3, pp. 45–49.

[6] L. S. Blackford, J. Choi, A. Cleary, E. D'Azevedo, J. Demmel, I. Dhillon, J. Dongarra, S. Hammarling, G. Henry, A. Petitet, K. Stanley, D. Walker, and R. C. Whaley, *ScaLAPACK Users' Guide*, SIAM, Philadelphia, PA, 1997.

[7] T. Cwik, R. van de Geijn, and J. Patterson, *The application of parallel computation to integral equation models of electromagnetic scattering*, J. Opt. Soc. Amer. A, 11 (1994), pp. 1538–1545.

[8] J. J. Dongarra, J. R. Bunch, C. B. Moler, and G. W. Stewart, *LINPACK Users' Guide*, SIAM, Philadelphia, PA, 1979.

[9] J. D. Gardiner and A. J. Laub, *A generalization of the matrix-sign-function solution for algebraic Riccati equations*, Int. J. Control, 44 (1986), pp. 823–832.

[10] A. V. Gerbessiotis, *Algorithmic and Practical Considerations for Dense Matrix Computations on the BSP Model*, Technical Report PRG-TR-32-97, Oxford University Computing Laboratory, Oxford, UK, 1997.

[11] J. Gunnels, C. Lin, G. Morrow, and R. van de Geijn, *A flexible class of parallel matrix multiplication algorithms*, in Proceedings of the First Merged International Parallel Processing Symposium and Symposium on Parallel and Distributed Processing (1998 IPPS/SPDP '98), Orlando, FL, 1998, IEEE Computer Society, Los Alamitos, CA, 1998, pp. 110–116.

[12] M. T. Heath, G. A. Geist, and J. B. Drake, *Early Experience with the Intel iPSC/860 at ORNL*, Technical Report ORNL/TM-11655, Oak Ridge National Laboratory, Oak Ridge, TN, 1990.

[13] N. J. Higham, *Accuracy and Stability of Numerical Algorithms*, SIAM, Philadelphia, PA, 1996.

[14] A. S. Householder, *The Theory of Matrices in Numerical Analysis*, Dover Publications, Blaisdell, NY, 1974.

[15] J. H. Maindonald, *Statistical Computation*, Wiley, New York, 1984.

[16] P. McCullagh and J. A. Nelder, *Generalized Linear Models*, Chapman and Hall, London, 1989.

[17] J. Roberts, *Linear model reduction and solution of algebraic Riccati equations by the use of the sign function*, Int. J. Control, 32 (1980), pp. 677–687.

[18] X. Sun, E. S. Quintana, G. Quintana, and R. van de Geijn, *Efficient Matrix Inversion via Gauss-Jordan Elimination and Its Parallelization*, Technical Report CS-98-19, Department of Computer Sciences, The University of Texas, Austin, TX, 1998. Available online at http://www.cs.utexas.edu/users/plapack/papers.

[19] R. van de Geijn and J. Watts, *SUMMA: Scalable universal matrix multiplication algorithm*, Concurrency: Practice and Experience, 9 (1997), pp. 255–274.

[20] R. A. van de Geijn, *Using PLAPACK: Parallel Linear Algebra Package*, The MIT Press, Cambridge, MA, 1997.

SIAM J. SCI. COMPUT.
Vol. 22, No. 5, pp. 1772–1790
© 2001 Society for Industrial and Applied Mathematics

ON THE COMPUTATION OF THE RANK-ONE CONVEX HULL OF A FUNCTION*

ERNESTO ARANDA† AND PABLO PEDREGAL†

Abstract. We report on several numerical experiments where the rank-one convexification of an energy density is computed. The explicit examples cover a whole spectrum of typical situations one may encounter. One of those is especially relevant for the computation of microstructures in crystalline solids.

Key words. laminates, maximum number of levels, rank-one convex hull

AMS subject classifications. 49J45, 49M, 73C50, 73V20

PII. S1064827599362028

1. Introduction.

The relevance of different types of convex hulls for vector variational problems is by now well established [23], [50]. One of the main driving forces that has led the effort in examining nonconvex vector variational problems has been the analysis of phase transitions in crystalline solids where symmetry is the main cause for this lack of convexity [3], [4]. In such situations, highly oscillatory minimizing sequences represent optimal behavior from an energetic point of view, and this in turn reflects rather accurately the configurations observed in experiments (the so-called microstructures), so that this energetic explanation seems to be the clue to understanding the behavior of such nonlinear materials. As can be easily inferred, the numerical analysis and simulation of such nonconvex vector variational problems has also received a lot of attention: [6], [7], [8], [9], [10], [11], [12], [13], [14], [15], [16], [17], [18], [19], [20], [24], [25], [26], [28], [29], [30], [31], [34], [35], [36], [37], [38], [39], [40], [41], [42], [44], [45], [46], [51]. In particular, the computation and approximation of the above-mentioned convex hulls can be considered as a first step toward the simulation of more complex examples with nonaffine boundary conditions [2], [34], [52]. In the present paper, we would like to explain our efforts in computing, from a practical point of view, the rank-one convex hull of a function and its meaning with respect to oscillatory behavior for vector, nonconvex variational problems.

Stable equilibrium configurations of hyperelastic materials are minimizers of the energy functional

$$I(u) = \int_\Omega \varphi(\nabla u(x))\, dx,$$

where $\Omega \subset \mathbf{R}^2$ is a regular domain and $u : \Omega \to \mathbf{R}^2$ represents a deformation of the body. As such, u must verify several additional restrictions which are not relevant to our discussion. Under a global condition of place, we require $u = u_0$ all over $\partial\Omega$. The direct method [23], [50] to show optimal solutions of the previous variational principle relies on the property of weak lower semicontinuity of I,

*Received by the editors October 4, 1999; accepted for publication (in revised form) July 26, 2000; published electronically January 16, 2001. This work is part of the European Community TMR network Phase Transitions in Crystalline Solids, contract FMRX-CT98-0229. It has also been supported by DGES (Spain) through grant PB96-0534.

http://www.siam.org/journals/sisc/22-5/36202.html

†ETSI Industriales, Universidad de Castilla-La Mancha, 13071 Ciudad Real, Spain (earanda@ind-cr.uclm.es, ppedrega@ind-cr.uclm.es).

$$u_j \rightharpoonup u \quad \text{implies} \quad I(u) \leq \liminf_{j\to\infty} I(u_j),$$

where weak convergence takes place on a suitable Sobolev space. This convenient property is in turn equivalent to the quasi-convexity of the energy density $\varphi : \mathbf{M} \to \mathbf{R}$, where $\mathbf{M}$ designates the space of 2×2 matrices [43]. Such φ is said to be quasi-convex if

$$\varphi(F) \leq \frac{1}{|T|}\int_T \varphi(F + \nabla v(x))\, dx$$

for all matrices F, all test fields $v \in W^{1,\infty}(T)$, and some domain T. This condition requires, under affine boundary conditions determined by the matrix F, that this affine deformation all over Q has less (or equal) energy than any other admissible deformation. When an energy density φ does not enjoy the quasi-convexity property, as is the case of crystalline materials, highly oscillatory behavior is expected in many circumstances. In such cases the analysis of the corresponding variational problem proceeds, examining a relaxed formulation where the integrand φ is replaced by its quasi-convexification $Q\varphi$:

$$Q\varphi(F) = \inf\left\{\frac{1}{|T|}\int_T \varphi(F + \nabla v(x))\, dx : v = 0 \text{ on } \partial\Omega\right\}.$$

T can be any domain since this infimum does not depend on this choice. A typical relaxation theorem says that, under technical assumptions, the infimum of both variational problems, the one with integrand φ and the one with density $Q\varphi$, are the same [21]. Moreover, the computation of this quasi-convex hull, as indicated above, can be considered as the first step towards the numerical analysis of the original nonconvex variational problem [48], [49].

The difficulties attached to understanding quasi-convexity and the quasi-convex hull are so great that some other intimately connected convex hulls have been introduced. In particular, rank-one convexity for a function φ requires the usual convexity inequality

$$\varphi(tA + (1-t)B) \leq t\varphi(A) + (1-t)\varphi(B), \quad t \in [0,1],$$

only when the difference matrix $A-B$ is of rank one. It turns out that quasi-convexity implies rank-one convexity, and this is not hard to see. But the converse is not true in general [53]. Nonetheless, in most of the explicit examples and experiments known, replacing the quasi-convex hull by its rank-one convex counterpart produces good results. In fact, minimizing sequences that arise from the rank-one convex hull of an energy density (the so-called laminates) provides configurations with a remarkable degree of accuracy both qualitative and quantitative. For these reasons we will restrict our attention to the rank-one convex hull in this work.

In a previous paper [1], we proposed to approximate the rank-one convex hull of a function by exploiting the structure of laminates [22], [47] and obtained an estimate of the type

$$0 \leq \varphi^{(L)}(F) - R\varphi(F) \leq C\lambda^L, \quad C > 0, \quad 0 < \lambda < 1,$$

where L indicates a precise level of approximation to the rank-one convexification of φ, $R\varphi$, and the constants C and λ are independent of L (see section 2). This time we would like to complete that analysis by trying to compute, from a practical

perspective, the approximations to $R\varphi$ indicated by $\varphi^{(L)}$. We have adapted to our situation an optimization algorithm based on the "simulated annealing" process, and we test such a procedure with several typical examples. The first one is the minimum of two quadratic functions where the location of the minima for both quadratics makes the approximation scheme different. This example was explicitly computed in closed form in [32] by using Fourier techniques. Since the structure of oscillations hidden in the nonconvexity of this example is rather simple, our analysis focuses on comparing the degree of precision for different situations. Our second example corresponds to the Ericksen–James energy [27]

$$\varphi(F) = \phi(F^T F),$$

where for $C = (c_{ij})$ the Cauchy–Green tensor

$$\phi(C) = \kappa_1(\operatorname{tr} C - 2 - \delta^2)^2 + \kappa_2(c_{12}^2 - \delta^2)^2 + \kappa_3(c_{11} - 1)^2.$$

κ_i and δ are given, positive constants. This test energy exhibits most of the relevant features that real energy densities for crystalline materials have, namely, frame indifference (this is elementary to check)

$$\varphi(F) = \varphi(RF) \quad \text{for all rotations } R,$$

and two potential wells: if

$$F_0^{\pm} = \mathbf{1} \pm \delta e_1 \otimes e_2,$$

then $\varphi(F_0^{\pm}) = 0$ and $\varphi > 0$ otherwise. By putting together these two pieces of information we conclude that the zero set of φ, K is the disjoint union of two wells

$$K = SO(2)F_0 \cup SO(2)F_0^{-1}.$$

Our simulations make an attempt to compute and approximate $R\varphi(F)$ for a selection of F's and to determine the underlying laminates. In this case, because of the relevance of this example with respect to real materials, we depict the information recorded in such laminates in the form of oscillations, as shown in Figure 1.

Finally, our third example is the remarkable four-matrix configuration where an infinite order laminate is involved [5], [54]. In this situation, our main concern was to detect computationally the structure of the laminates and how computations evolved as we let the number of levels increase. We discover some interesting numerical facts that speak about the complexity of computing the rank-one convexification of a function because of the global nature of the underlying optimization problem. Our pictures for this example show the best computed laminates constrained in the maximum number of levels allowed. In the sequence of pictures in Figures 2–4 the last level has been colored differently.

The paper has three more sections. We first describe the algorithm and make some observations of a technical nature. Next, we precisely describe the three above situations analyzed by using the algorithm. We make some final remarks in section 4.

2. Description of the algorithm. Our algorithm tries to compute the approximated rank-one convex hull of a function as described in [1]. Specifically, if $\varphi : \mathbf{M}^{m\times N} \to \mathbf{R}$ is a continuous function satisfying the technical assumptions

$$c(|A|^p - 1) \le \varphi(A) \le C(1 + |A|^p), \quad p > N, \quad C > c > 0,$$

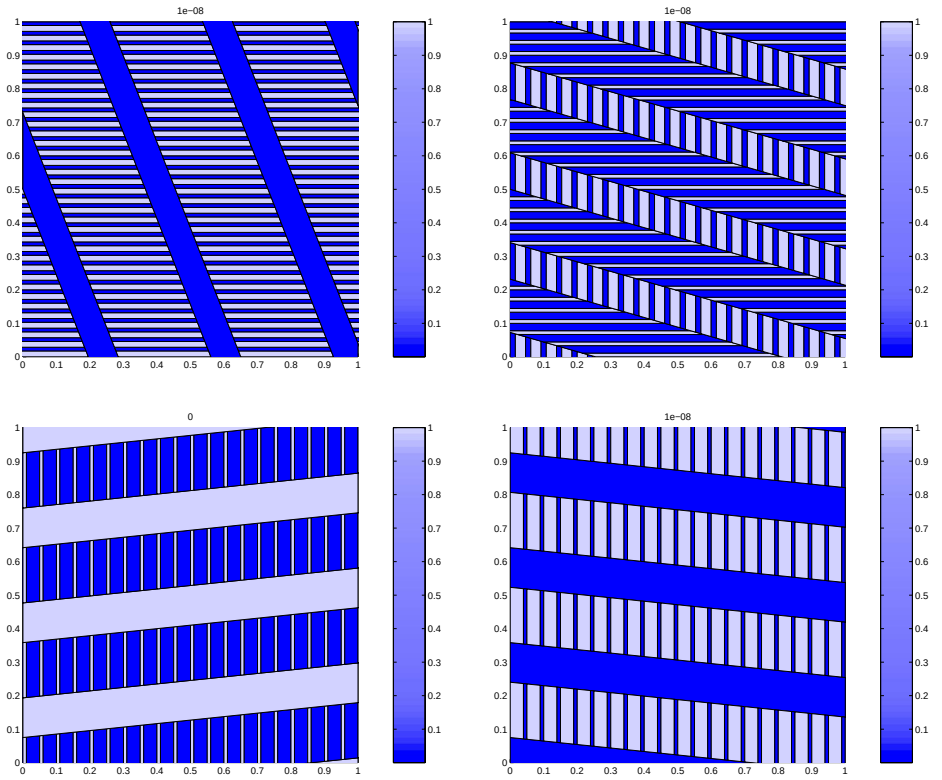


FIG. 1.

its approximated rank-one convex hull of order L is given by

$$\varphi_\epsilon^{(L)}(F) = \inf\left\{\sum_i t_i\varphi(F_i) : \{(t_i, F_i)\} \text{ verifies the } (H_\epsilon^L) \text{ condition and has barycenter } F\right\}. \tag{2.1}$$

Here, $\epsilon > 0$ is meant to designate a realistic measure of accuracy in our actual computations, and the (H_ϵ^L) conditions are defined as follows.

DEFINITION 2.1. *A set of pairs $\{(t_i, F_i)\}_{1\le i\le l}$, where $t_i > 0$, $\sum_i t_i = 1$, $F_i \in \mathbf{M}$, is said to satisfy the (H_ϵ^L) condition, $\epsilon \ge 0$, if*

(i) *the only set of pairs satisfying the (H_ϵ^0) is $\{(1, F)\}$;*

(ii) *for $L > 0$, there exist*

  (a) *$\{(s_j, Z_j)\}_{1\le j\le k}$ satisfying the (H_ϵ^{L-1}) condition, $k \le L-1$;*

  (b) *matrices A_j, $|A_j| = 1$ with $\operatorname{rank}(A_j) = 1$;*

  (c) *nonnegative numbers α_j, β_j such that either $\alpha_j = 0$ or else $\alpha_j > \epsilon$ and the same for β_j*

such that all the matrices $Z_j + \alpha_j A_j$ and $Z_j - \beta_j A_j$ belong to the set $\{F_i\}_{1\le i\le l}$ and for every fixed i, $1 \le i \le l$,

$$t_i = \sum_{Z_j+\alpha_j A_j = F_i} s_j \frac{\alpha_j}{\alpha_j+\beta_j} + \sum_{Z_j-\beta_j A_j = F_i} s_j \frac{\beta_j}{\alpha_j+\beta_j}.$$

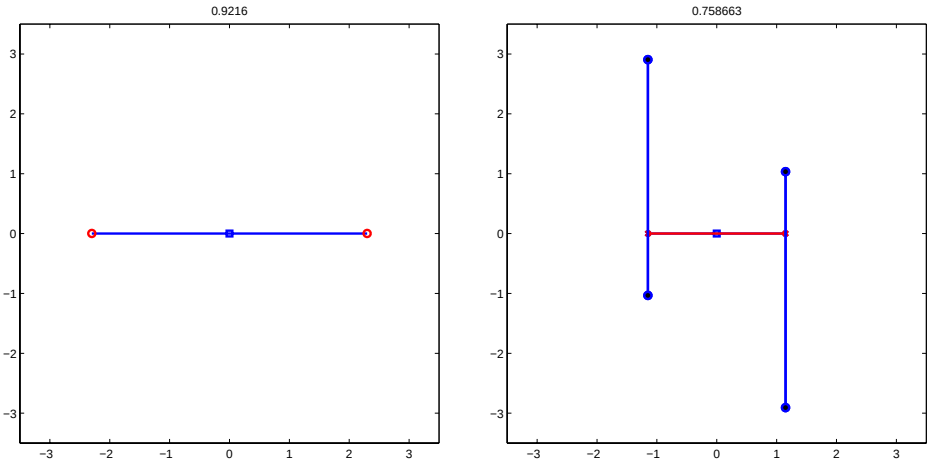


FIG. 2.

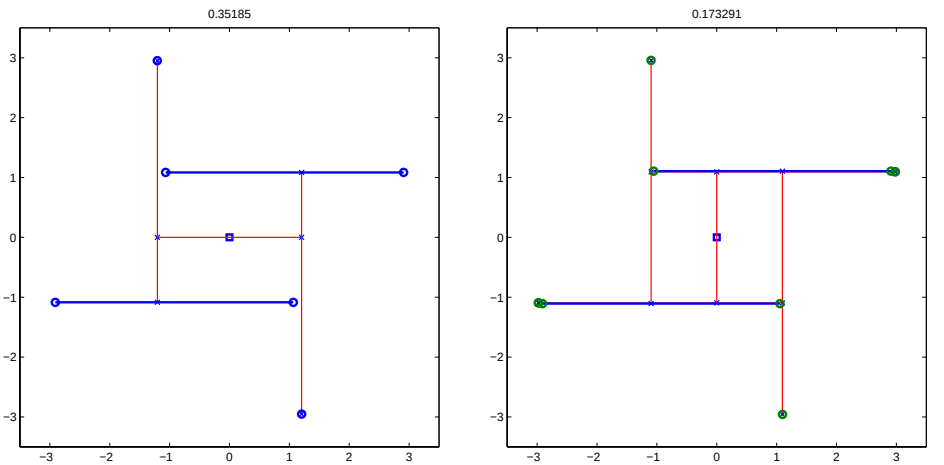


FIG. 3.

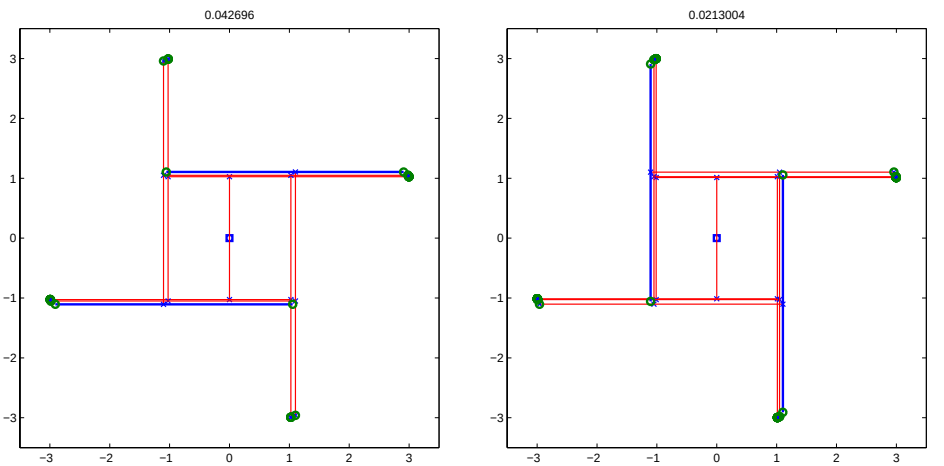


FIG. 4.

This definition differs from the typical (H_L) condition [22], [47] in a formal way. We found this format more convenient for our purposes.

The algorithm we present approximates $\varphi_\epsilon^{(L)}(F)$ by trying to find one optimal (H_ϵ^L) condition with barycenter F according to (2.1). In practice, we have to deal with and construct (H_ϵ^L) conditions. In order to generate one of these, we rely on the basic building block of these conditions. Starting from one arbitrary matrix Y, we choose a rank-one direction (eventually a null-direction) A, normalized in some reasonable way, along which we decompose Y by fixing two distances, $x_1 \geq \epsilon$ and $x_2 \geq \epsilon$, from Y along A:

$$Y_1 = Y + x_1 A, \quad Y_2 = Y - x_2 A.$$

If

$$\lambda_i = \frac{x_j}{x_1 + x_2}, \quad j \neq i,$$

then it is trivial to check

$$Y = \lambda_1 Y_1 + \lambda_2 Y_2, \quad Y_1 - Y_2, \text{ rank-one.}$$

(H_ϵ^L) conditions are built by applying recursively this decomposition to all of the matrices obtained. Starting from the first moment F and choosing A, x_1, x_2 as above, we find (F_1, λ_1), (F_2, λ_2). Applying this same decomposition to each F_i, we would find

$$(F_{11}, \lambda_1\lambda_{11}), (F_{12}, \lambda_1\lambda_{12}), (F_{21}, \lambda_2\lambda_{21}), (F_{22}, \lambda_2\lambda_{22}).$$

This set of pairs would make up an (H_ϵ^2) condition. If we keep doing these decompositions with all of the F_{ij}, we would end up with an (H_ϵ^3) condition, and so on. The parameters that determine one of such (H_ϵ^L) conditions are the rank-one matrices A and the associated distances x_1, x_2.

In the case of 2×2 matrices, rank-one directions can be determined by two angles α, β in the interval $[0, \pi)$ in the form

$$M = (\cos\alpha_1, \sin\alpha_1) \otimes (\cos\alpha_2, \sin\alpha_2).$$

Rank-one matrices are recorded in a vector α. On the other hand, the distances associated to such rank-one matrices are recorded in a vector x. If we are interested in approximating $\varphi_\epsilon^L(F)$, (H_ϵ^L) conditions can be specified by a vector, α, that generates rank-one matrices, and a corresponding vector x of associated distances. Namely, matrices and weights in the (H_ϵ^L) condition are obtained recursively by putting

$$\begin{aligned} F_{2i-1} &= F_{i-1} + x_{2i-1} M_i, \quad F_{2i} = F_{i-1} - x_{2i} M_i, \\ \lambda_{2i-1} &= \lambda_{i-1} \frac{x_{2i}}{x_{2i-1} + x_{2i}}, \quad \lambda_{2i} = \lambda_{i-1} \frac{x_{2i-1}}{x_{2i-1} + x_{2i}} \end{aligned}$$

for $1 \leq i \leq 2^L - 1$, $\lambda_0 = 1$, $F_0 = F$, where $\epsilon \leq x \leq D$ for some constant D. The energy associated to such a laminate would be

$$E(\alpha, x) = \sum_{i=2^L-1}^{2^{L+1}-2} \lambda_i \varphi(F_i).$$

It is this function E whose global minimum, and vectors where it is taken on, we are interested in.

In order to tackle the global optimization problem we are faced with, we have adopted a scheme where the roles of α and x are different. Indeed, we have used for α a discretization of equidistant nodes in the interval $[0, \pi)$, generating a set of discrete rank-one directions Γ_h, where h is the distance between nodes in $[0, \pi)$. The optimization procedure goes alternatively from optimizing on x for given α, and optimizing on α for given x. For the first part, we have used an algorithm of limited memory [7], [8] which is a quasi-Newton algorithm that approximates the hessian of the objective function by limited-memory matrices. For the second part, we have employed the simulated annealing technique in order to find the best configuration of rank-one matrices from the discrete set of possibilities Γ_h, in some instances improving the optimal solution thus obtained by an easy corrector procedure. Because of the nature of our problem, the initialization of the algorithm is a delicate point. In some instances one can simply try to decompose matrices in the best way using only one level at a time. However, this scheme will not produce nontrivial decompositions starting from the barycenter in the case of the four well. It is when we allow two or more levels that we are able to move away from a delta centered at the first moment. In some cases, several thousands of initial guesses were necessary in order to initialize the algorithm in a convenient way. This feature actually shows how complex and tricky these computations may become.

3. Numerical examples. As announced in the introduction, we will be concerned with three quite different situations, and we use our algorithm to make computations.

For the first one we take

$$\varphi : \mathbf{M} \to \mathbf{R}, \quad \varphi(A) = \min\{|A - M|^2, |A|^2\},$$

where M is a given matrix. The quasi-convexification (which in this case coincides with the rank-one convexification) of this type of examples was explicitly computed in [32]. Namely, one has to distinguish two cases depending on the rank of M. If it is one, the quasi-convexification equals the usual convexification. If it is two, the quasi-convexification is given explicitly by

$$Q\varphi(A) = \begin{cases} |A|^2, & |A - M|^2 - |A|^2 \geq g/2, \\ |A - M|^2, & |A - M|^2 - |A|^2 \leq -g/2, \\ |A|^2 - \frac{1}{2g}\left(|A|^2 - |A - M|^2 + \frac{g}{2}\right)^2, & ||A - M|^2 - |A|^2| \leq g/2. \end{cases}$$

When the rank of M is one, $g = 2|M|^2$, while in the case of rank two, $g = 2\sigma_0$, where σ_0 represents the greatest of the eigenvalues of the matrix $M^T M$. In both cases, the rank-one convexification is achieved by $\varphi^{(1)}$ one-level decompositions.

The experiments performed on this family of examples correspond to four different choices for M, two in the rank-one case and two in the rank-two case. Namely,

$$M_1 = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}, \quad M_2 = \left(\cos\frac{\pi}{5}, \sin\frac{\pi}{5}\right) \otimes \left(\cos\frac{\pi}{3}, \sin\frac{\pi}{3}\right),$$
$$M_3 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad M_4 = \begin{pmatrix} 2 & 1 \\ 1/2 & 1/2 \end{pmatrix}.$$

For each one of the corresponding densities, we have computed $\varphi_h^{(1)}$, the approximation to $\varphi_\epsilon^{(1)}$ provided by our algorithm, over a set of matrices Δ_k uniformly distributed over the cube $[-D, D]^4$, and then we have examined the error

$$\left\|\varphi_h^{(1)} - R\varphi\right\|_{\infty,\Delta_k} = \max\left\{\left|\varphi_h^{(1)}(A) - R\varphi(A)\right| : A \in \Delta_k\right\}.$$

These errors are presented in Tables 1 and 2. The parameter $N = \pi/h$ is a measure of the fineness of the discretization of rank-one directions, as explained in the preceding section.

TABLE 1

	M_1	M_3
$N = 2$	$0.44408921 \times 10^{-15}$	$0.88817842 \times 10^{-15}$
$N = 6$	$0.44408921 \times 10^{-15}$	$0.17763568 \times 10^{-14}$
$N = 12$	$0.44408921 \times 10^{-15}$	$0.17763568 \times 10^{-14}$

TABLE 2

	M_2	M_4
$N = 2$	$0.86372827 \times 10^{-1}$	0.36354086
$N = 6$	$0.19029316 \times 10^{-8}$	$0.33946668 \times 10^{-9}$
$N = 12$	$0.19029316 \times 10^{-8}$	$0.58557923 \times 10^{-8}$

We notice that the approximation is better for the rank-one case, and that the error depends on how close M, or its projection over the set of rank-one matrices, is to the discrete set of rank-one directions Γ_h.

We have also observed, even in this simple setting, the difficulties associated to local minima. For example, for the density corresponding to M_2, the matrix

$$A = \begin{pmatrix} 1 & -1/2 \\ -1 & 1/2 \end{pmatrix}$$

is such that $\varphi(A) = 2.5$ while $R\varphi(A) = 2.49978037599552$. Even if we let $N = 12$, the algorithm does not lower the energy, and only when $N = 20$ does the best energy have an error of 0.22×10^{-10} over the rank-one convexification.

The second situation we have examined concerns the Ericksen–James energy density

$$\varphi(F) = \phi(F^T F), \quad \phi(C) = \kappa_1(\operatorname{tr} C - 2 - \delta^2)^2 + \kappa_2(c_{12}^2 - \delta^2)^2 + \kappa_3(c_{11} - 1)^2,$$

where $\kappa_i, \delta > 0$ are constants. This function enjoys the frame indifference property, and its zero set, $K = \{\varphi = 0\}$, has the structure of two potential wells

$$K = SO_2F_0 \cup SO_2F_0^{-1}, \quad F_0^{\pm 1} = \mathbf{1} \pm \delta e_1 \otimes e_2.$$

We are especially interested in computing the rank-one convexification of φ for matrices F that may support stress free microstructures $R\varphi(F) = 0$. These matrices F, as well as the compatibility of wells, reciprocal twins, and laminates supported in K, are very well known [4], [50].

The visualization of our experiments has been implemented by showing the structure of layers within layers for different levels of approximation. In order to represent the states for the gradients of a minimizing sequence, we have used a color function

$$\psi(F) = \frac{\left|F^T F - F_0^T F_0\right|}{\left|F^T F - F_0^T F_0\right| + \left|F^T F - (F_0^{-1})^T F_0^{-1}\right|},$$

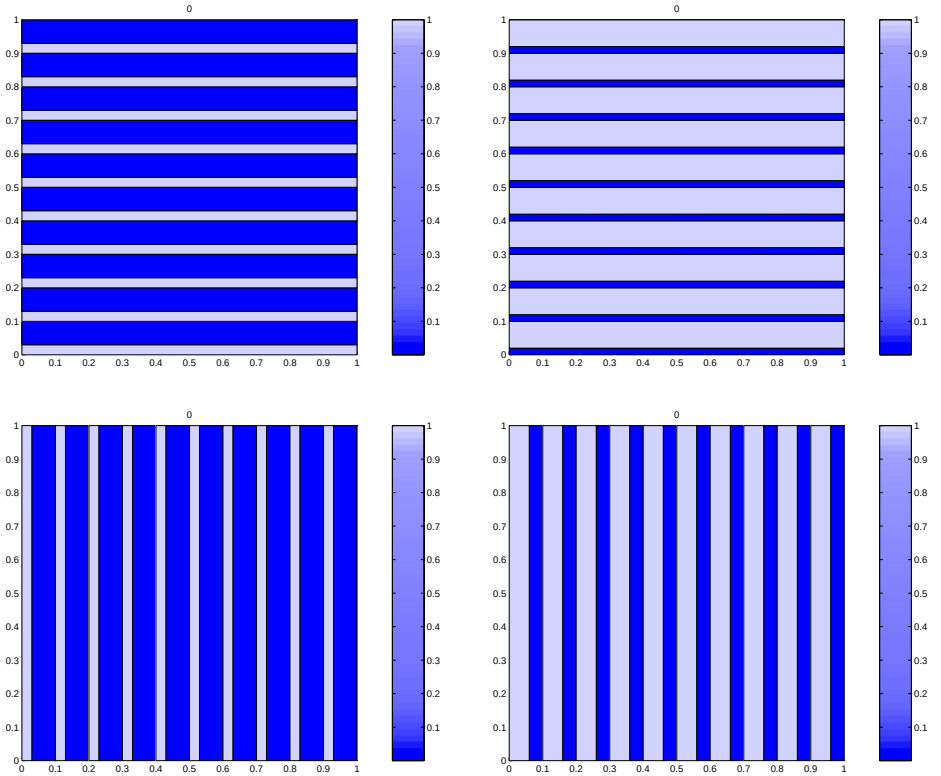


FIG. 5.

which takes on the value 1 if F belongs to $SO_2F_0^{-1}$, and its value is 0 if F lies on the other well SO_2F_0. We have also utilized the computed normals for the laminates in order to depict the oscillatory behavior which would be associated to such minimizing sequences. The width of layers is indicated by the weights on the associated laminate. We stick to the notation

$$U^i_{\lambda,t} = (1-\lambda)Q_tR_iF_0 + \lambda Q_tF_0^{-1}, \quad t \in [0, 2\pi], \quad i = 1, 2, \quad \lambda \in [0, 1], \tag{3.1}$$

where

$$Q_t = \begin{pmatrix} \cos t & -\sin t \\ \sin t & \cos t \end{pmatrix}, \quad R_1 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad R_2 = \frac{1}{1+\delta^2}\begin{pmatrix} 1-\delta^2 & -2\delta \\ 2\delta & 1-\delta^2 \end{pmatrix}.$$

R_2 represents the reciprocal twin. Our experiments take place on the domain $\Omega = (0,1)^2$ with $\kappa_1 = 10$, $\kappa_2 = 3$, $\kappa_3 = 1$, and $\delta = 1$.

The first experiments deal with several situations where the uniqueness of the underlying laminate is known [50]. According to the above notation, we consider affine boundary values given by the matrices

$$U^1_{0.3,0},\ U^1_{0.8,\pi/5},\ U^2_{0.3,0},\ U^1_{0.6,\pi/7}.$$

The computed laminates are shown in Figure 5, while computed values of the energy and volume fractions are contained in Table 3.

TABLE 3

	$U^1_{0.3,0}$	$U^1_{0.8,\pi/5}$	$U^2_{0.3,0}$	$U^1_{0.6,\pi/7}$
φ	9.17280	5.32480	4.05720	5.29920
E	0.43×10^{-19}	0.13×10^{-10}	0.19×10^{-17}	0.45×10^{-10}
λ	0.3	0.19999970	0.3	0.60000094

We are aware that if the matrix chosen as the underlying barycenter does not admit a decomposition as in (3.1), then a surprising nonuniqueness for the underlying microstructure occurs. Indeed, many laminates of the form

$$(1-\mu)(1-\lambda)\delta_{\hat{Q}F_0} + (1-\mu)\lambda\delta_{\hat{Q}F_0^{-1}} + \mu(1-\lambda)\delta_{F_0^{-1}} + \mu\lambda\delta_{F_0}$$

for

$$\hat{Q} = \frac{1}{1+\delta^2(1-2\lambda)^2}\begin{pmatrix} 1-\delta^2(1-2\lambda)^2 & -2\delta(1-2\lambda) \\ 2\delta(1-2\lambda) & 1-\delta^2(1-2\lambda)^2 \end{pmatrix},$$

for different choices of μ and λ, can be found for the same barycenter [50]. If we take

$$U = \begin{pmatrix} 4/5 & 1/2 \\ -2/5 & 1 \end{pmatrix}, \tag{3.2}$$

which corresponds to taking $\mu = 1/2$, $\lambda = 3/4$, and

$$\hat{Q} = \begin{pmatrix} 3/5 & 4/5 \\ -4/5 & 3/5 \end{pmatrix},$$

the associated laminate is the one in Figure 6(a). However, this particular microstructure has not been obtained in our numerous experiments, but rather different laminates (with zero energy) for the same underlying affine deformation given by U in (3.2) have been computed. The one in Figure 6(b) is associated to the convex combination

$$(1-\lambda)(1-\mu)\delta_{F_0} + (1-\lambda)\mu\delta_{R_2^T F_0^{-1}} + \lambda(1-\sigma)\delta_{\overline{Q}F_0} + \lambda\sigma\delta_{\overline{Q}F_0^{-1}},$$

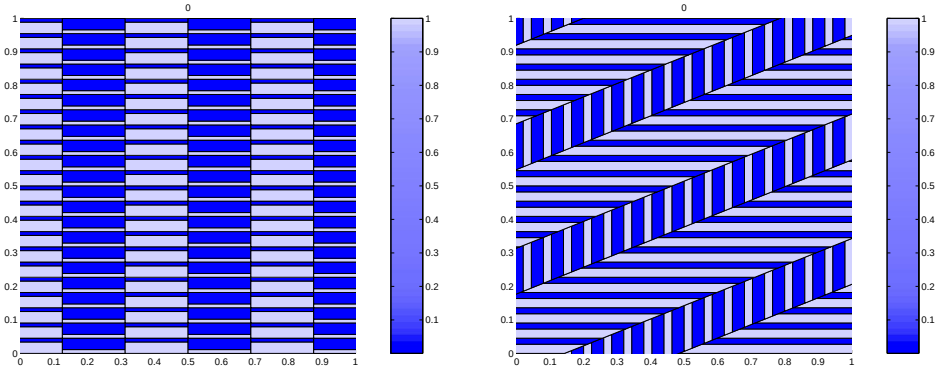
where

$$\lambda = \frac{1}{\sqrt{21}-3}, \quad \mu = \frac{2}{5}, \quad \sigma = \frac{7-\sqrt{21}}{4},$$
$$\overline{Q} = \begin{pmatrix} \sqrt{21}/5 & 2/5 \\ -2/5 & \sqrt{21}/5 \end{pmatrix}, \quad m_1 = \frac{1}{\sqrt{29}}\begin{pmatrix} -2 \\ 5 \end{pmatrix}.$$

m_1 is the normal to the interface between $(1-\mu)F_0 + \mu R_2^T F_0^{-1}$ and $(1-\sigma)\overline{Q}F_0 + \sigma\overline{Q}F_0^{-1}$, while $m_2 = (1,0)$ is the normal between F_0 and $R_2^T F_0^{-1}$, and $m_3 = (1,0)$ is the one corresponding to $\overline{Q}F_0$ and $\overline{Q}F_0^{-1}$. Table 4 shows the computed values for all these parameters.

We have also found laminates with the same barycenter U in (3.2) supported in three matrices [4]

$$(1-\lambda_i)Q_{s_i}F_0^{-1} + \lambda_i(1-\mu_i)Q_{t_i}F_0 + \lambda_i\mu_i Q_{t_i}F_0^{-1}, \quad i = 1,2,$$



(a) (b)

FIG. 6.

where

$$\lambda_1 = \frac{17}{42} + \frac{4}{105}\sqrt{85}, \quad \lambda_2 = \frac{17}{49} + \frac{5}{196}\sqrt{85},$$
$$\mu_1 = 5 - \frac{\sqrt{85}}{2}, \quad \mu_2 = \frac{8}{3} - \frac{4}{15}\sqrt{85},$$
$$\cos s_1 = \frac{56}{97} - \frac{17}{485}\sqrt{85}, \quad \cos s_2 = \frac{56}{97} + \frac{17}{485}\sqrt{85},$$
$$\sin s_1 = \frac{68}{97} + \frac{14}{485}\sqrt{85}, \quad \sin s_2 = \frac{68}{97} - \frac{14}{485}\sqrt{85},$$
$$\cos t_1 = \frac{9}{85}\sqrt{85}, \quad \cos t_2 = -\frac{2}{85}\sqrt{85},$$
$$\sin t_1 = \frac{2}{85}\sqrt{85}, \quad \sin t_1 = \frac{9}{85}\sqrt{85}.$$

Figure 7 and Table 5 show the approximated minimizing gradients and computed numerical values. Finally, Figure 1 corresponds to other different microstructures associated to the same affine underlying deformation.

TABLE 4

λ	0.63191385	m_1	$(-0.37141829, 0.92846565)$
μ	0.40001470	m_2	$(-1, 0)$
σ	0.60432472	m_3	$(0.00002523, 1)$

Our last situation refers to the example with four matrices, none of which is rank-one related to the other three. Despite this fact, nontrivial infinite order laminates are supported in this set of four matrices. This is one of those examples that must be tested for any new way of computing laminates and rank-one convex hulls of functions. We will consider the particular configuration with

$$A_1 = \begin{pmatrix} -3 & 0 \\ 0 & -1 \end{pmatrix}, \quad A_2 = \begin{pmatrix} 1 & 0 \\ 0 & -3 \end{pmatrix},$$
$$A_3 = \begin{pmatrix} 3 & 0 \\ 0 & 1 \end{pmatrix}, \quad A_4 = \begin{pmatrix} -1 & 0 \\ 0 & 3 \end{pmatrix},$$

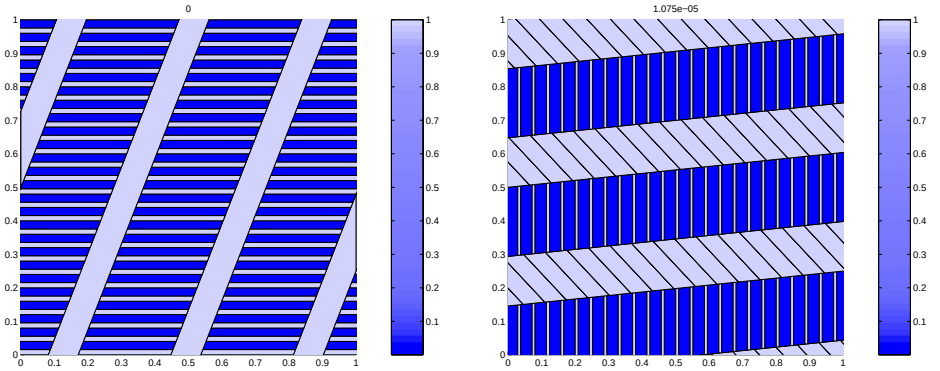


FIG. 7.

TABLE 5

λ_1	0.24402617	λ_2	0.41786234
μ_1	0.39022623	μ_2	0.20813357
E_1	0.14×10^{-8}	E_2	1.07×10^{-5}

and the integrand $\varphi : \mathbf{M}^{2\times 2} \to \mathbf{R}$ given by

$$\varphi(A) = 10^{-4} \prod_{i=1}^{4} |A - A_i|^2 .$$

If

$$B_1 = \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix}, \quad B_2 = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix},$$
$$B_3 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad B_4 = \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix},$$

then it is well known (see the above references) that the rank-one convexification of φ vanishes within the square $B_1B_2B_3B_4$ and that this value can only be achieved by an infinite order laminate. From a numerical point of view, we would like to see if our algorithm is capable of capturing the successive approximated (H^L) conditions. Since in this situation we are interested not so much in the structure of the underlying minimizing gradients as in the structure of the approximated (H^L) conditions, we will draw these in the two-dimensional subspace of diagonal matrices rather than representing deformation gradients in the reference configuration.

One of the main difficulties in computing optimal (H^L) conditions is the fact that knowing the optimal such configuration for L does not help in finding the one for $L+1$. Indeed, in many cases these are surprisingly unrelated. In what follows we will identify points in the plane (x, y) with diagonal matrices

$$\begin{pmatrix} x & 0 \\ 0 & y \end{pmatrix}.$$

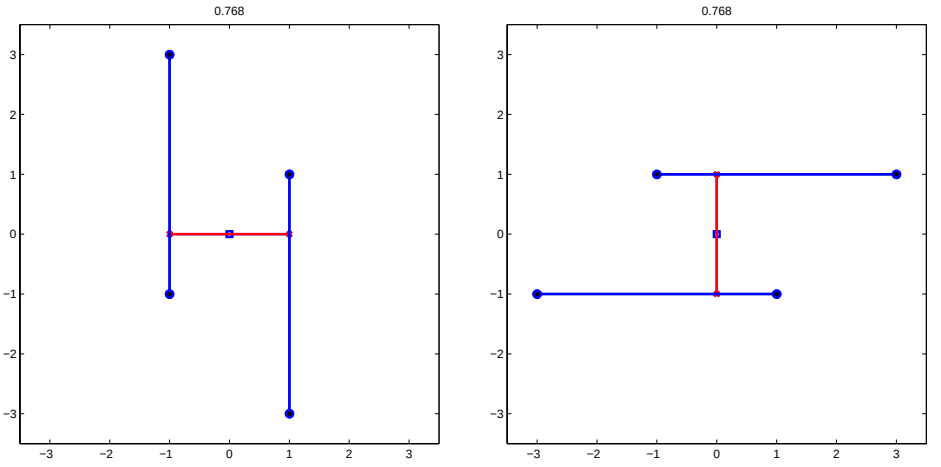


FIG. 8.

We will start by examining optimal (H^L) conditions with first moment the zero matrix, but before we do that we believe it is instructive to describe the basic idea on the construction of the infinite order laminate which yields a vanishing value for the rank-one convexification of φ. We will describe the situation taking as barycenter B_1, but then it can immediately be adapted to other matrices. Indeed, it is a matter of checking the decompositions that follow, making sure that they are made along rank-one matrices, and using them in a recursive manner:

$$B_1 = \frac{1}{2}A_1 + \frac{1}{2}B_2, \quad B_2 = \frac{1}{2}A_2 + \frac{1}{2}B_3,$$
$$B_3 = \frac{1}{2}A_3 + \frac{1}{2}B_4, \quad B_4 = \frac{1}{2}A_4 + \frac{1}{2}B_1.$$

Then

$$B_1 = \frac{1}{2}A_1 + \frac{1}{2}\left(\frac{1}{2}A_2 + \frac{1}{2}B_3\right),$$
$$B_1 = \frac{1}{2}A_1 + \frac{1}{2}\left(\frac{1}{2}A_2 + \frac{1}{2}\left(\frac{1}{2}A_3 + \frac{1}{2}B_4\right)\right)$$
$$\cdots$$

If we keep doing these decompositions, we build an infinite order laminate supported on the four matrices A_i, and hence the rank-one convexification of φ on B_1 vanishes. The same is true for any matrix in the square $B_1B_2B_3B_4$, as pointed out above, and also for the segments $[A_i, B_i]$, $i = 1, 2, 3, 4$.

Figure 8 shows the expected (in the sense of the recursive decompositions given above) (H^2) conditions

$$(0,0) = \frac{1}{2}(1,0) + \frac{1}{2}(-1,0)$$
$$= \frac{1}{2}\left[\frac{3}{4}(1,1) + \frac{1}{4}(1,-3)\right] + \frac{1}{2}\left[\frac{3}{4}(-1,-1) + \frac{1}{4}(-1,3)\right],$$

or

$$(0,0) = \frac{1}{2}(0,1) + \frac{1}{2}(0,-1)$$
$$= \frac{1}{2}\left[\frac{3}{4}(-1,1) + \frac{1}{4}(3,1)\right] + \frac{1}{2}\left[\frac{3}{4}(1,-1) + \frac{1}{4}(-3,-1)\right],$$

both with energy $E = 0.768$. Our experiments yield a slight variation on these decompositions. We see in Figure 2 that the optimal (H^1) has not been taken as a starting point for the best (H^2) condition. This in turn is slightly different from the one shown in Figure 8. The computed matrices are

$$(1.1493, 1.03359), (1.1493, -2.9066), (-1.1493, 2.9066), (-1.1493, -1.0335)$$

with an energy value slightly better $E = 0.75866$. In Figure 3, optimal structures of level three and four ($L = 3$ and $L = 4$, respectively) obtained by our algorithm have been drawn. The values for the energy are $E_3 = 0.35185$ and $E_4 = 0.17329$, respectively. They are again better than the ones corresponding to the limit laminate (see Table 6). It is also interesting to notice that the support of optimal (H^L) conditions tends to the zero set, $K = \{A_i\}$, of φ as L increases. This can be seen in the supports for the computed optimal (H^3) and (H^4) conditions

$$K_3 = \{(1.2042, -2.9511), (-1.2042, 2.9511), (-2.9094, -1.0851),$$
$$(2.9094, 1.0851), (1.0646, -1.0851), (-1.0646, 1.0851)\},$$
$$K_4 = \{(-2.9793, -1.0951), (1.0960, -2.9589), (2.9088, 1.1063), (-2.9088, -1.1063),$$
$$(-1.0960, 2.9589), (2.9793, 1.0951), (-1.0545, 1.1063), (1.0545, -1.1063)\}.$$

For a greater number of levels, this trend is maintained. Optimal values of the energy are a bit better than the expected values, while the matrices in the support tend to the wells of φ. In Figure 4, we show the optimal (H^6) and (H^7) conditions.

TABLE 6

Expected	Computed
$E_2 = 0.768$	$E_2 = 0.75866$
$E_3 = 0.384$	$E_3 = 0.35185$
$E_4 = 0.192$	$E_4 = 0.17329$
$E_5 = 0.096$	$E_5 = 0.08578$
$E_6 = 0.048$	$E_6 = 0.04269$
$E_7 = 0.024$	$E_7 = 0.0213$

Similar results have been obtained for matrices located over the boundary of the square $B_1B_2B_3B_4$. Figure 9 shows computed optimal (H^5) conditions for some of those matrices.

So far, the experiments have shown a clear similarity with the expected result. For matrices in the interior of the square mentioned in the last paragraph this is not so. For instance, let us focus on $(1/2, 1/2)$. Up to level four ($L = 4$), one would expect a structure similar to (see Figure 10(a))

$$\left(\frac{1}{2}, \frac{1}{2}\right) = \frac{3}{4}\left(\frac{1}{2}, 1\right) + \frac{1}{4}\left(\frac{1}{2}, -1\right)$$
$$= \frac{3}{4}\left[\frac{5}{8}(-1,1) + \frac{3}{8}(3,1)\right] + \frac{1}{4}\left[\frac{7}{8}(1,-1) + \frac{1}{8}(-3,-1)\right]$$

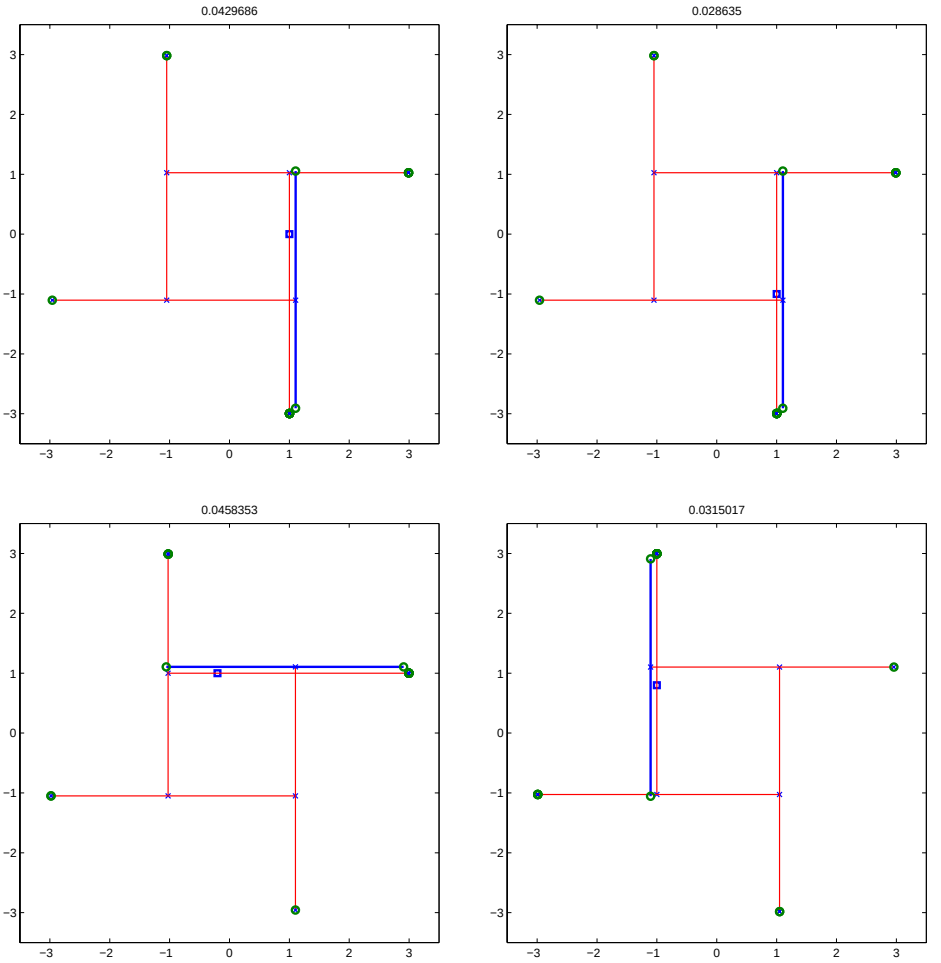
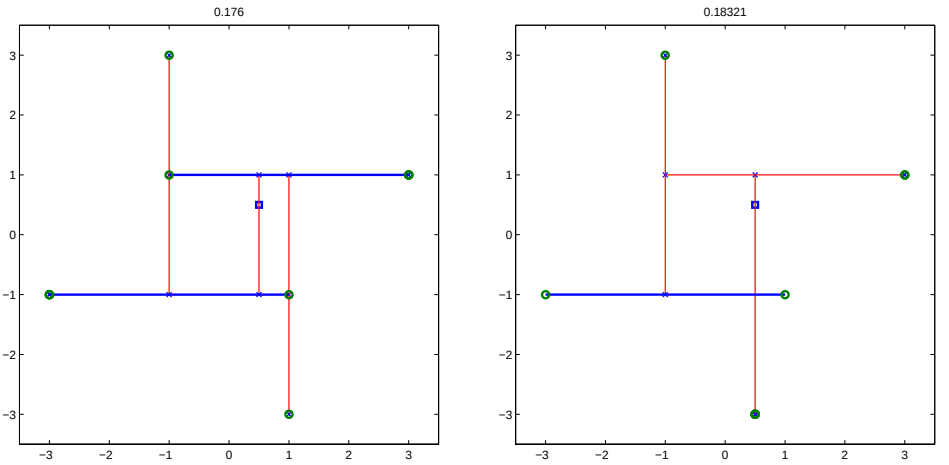


FIG. 9.

$$
\begin{aligned}
&= \frac{15}{32}\left[\frac{1}{2}(-1,-1)+\frac{1}{2}(-1,3)\right]+\frac{7}{32}\left[\frac{1}{2}(1,1)+\frac{1}{2}(1,-3)\right]\\
&\quad+\frac{9}{32}(3,1)+\frac{7}{32}(-3,-1)\\
&= \frac{15}{64}\frac{1}{2}(1,-1)+\frac{7}{64}\frac{1}{2}(-1,1)+\frac{15}{64}(-1,3)+\frac{7}{64}(1,-3)\\
&\quad+\left(\frac{7}{32}+\frac{15}{128}\right)(-3,-1)+\left(\frac{9}{32}+\frac{7}{128}\right)(3,1).
\end{aligned}
$$

However, we found that the decomposition (Figure 10(b))

$$
\begin{aligned}
\left(\frac{1}{2},\frac{1}{2}\right) &= \frac{7}{8}\left(\frac{1}{2},1\right)+\frac{1}{8}\left(\frac{1}{2},-3\right)\\
&= \frac{7}{8}\left[\frac{5}{8}(-1,1)+\frac{3}{8}(3,1)\right]+\frac{1}{8}\left(\frac{1}{2},-3\right)
\end{aligned}
$$



(a) (b)

FIG. 10.

$$= \frac{35}{64}\left[\frac{1}{2}(-1,-1)+\frac{1}{2}(-1,3)\right]+\frac{21}{64}(3,1)+\frac{1}{8}\left(\frac{1}{2},-3\right)$$
$$= \frac{35}{128}\frac{1}{2}(1,-1)+\frac{35}{128}\frac{1}{2}(-1,3)+\frac{35}{128}(-3,-1)+\frac{21}{64}(3,1)+\frac{1}{8}\left(\frac{1}{2},-3\right)$$

is the best one up to three levels, and only from four levels on the expected decomposition beats the last one. The numerical values are in Table 7, and the optimal structures are in Figure 11. This deviation from expected structures is more clear as we move toward the boundary of the square. In some cases one needs to let up to five levels of decomposition in order to see the expected structure for the limit optimal laminate. For instance, for the matrix $(8/10, 8/10)$ one must go up to seven levels.

TABLE 7

Expected	Computed
$E_2 = 0.704$	$E_2 = 0.509254$
$E_3 = 0.352$	$E_3 = 0.30375$
$E_4 = 0.176$	$E_4 = 0.15801$
$E_5 = 0.08$	$E_5 = 0.07837$

4. Final remarks. We would like to make some comments on the algorithm used to implement the numerical experiments in this paper.

It is important to notice that the number of variables involved exponentially increases with the number of levels L, thus making the computations unfeasible for large values of L. On the other hand, the main estimate in [1] ensures that the error reduces exponentially with L as well, so that there is no need, in principle, to use many different levels in real computations. Nonetheless, it would be interesting to have error estimates in approximating φ^L by our algorithm. Since the computation of $\varphi^L(A)$ is essentially a global optimization problem, there is no standard way of measuring how far we are from the real global optimum. An attempt in this direction

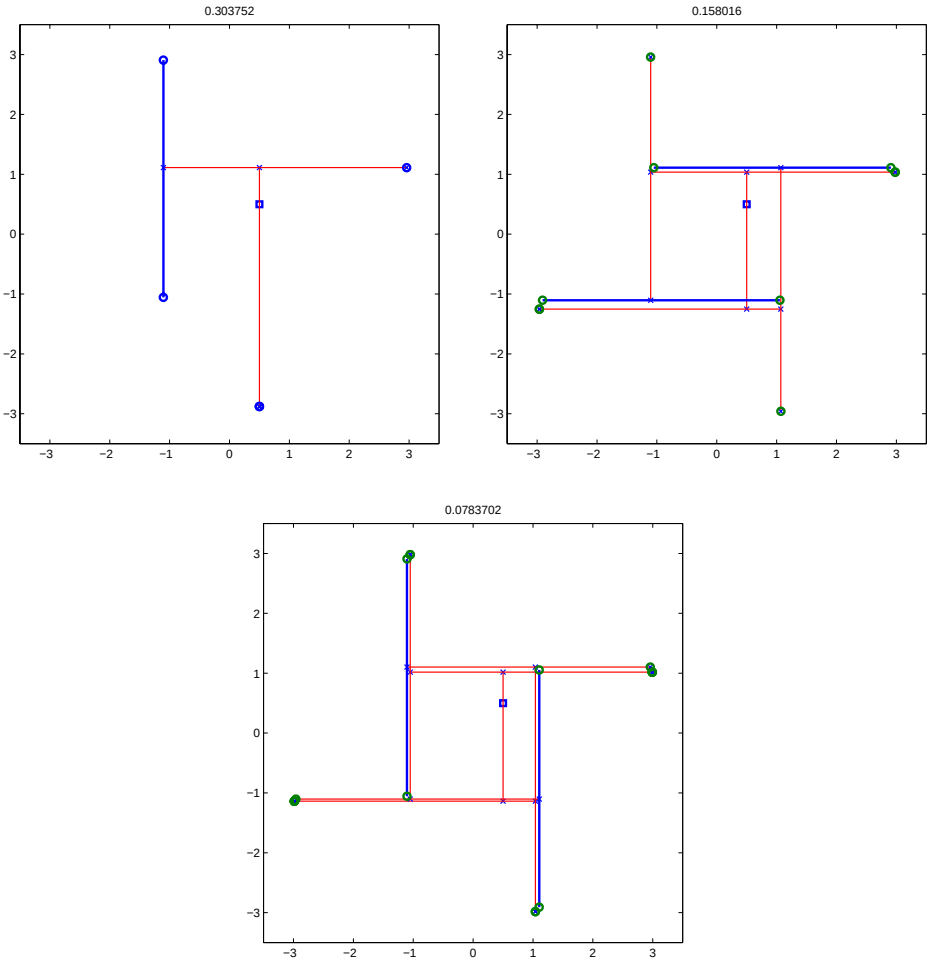


FIG. 11.

is made in [24], [25], [26]. In many cases we cannot be sure if we are even close to the optimum sought, although in our examples that seems to be the case. In most of the practical algorithms utilized for global optimization problems (like simulated annealing) several parameters must be tuned for good results. In our case, only the situation with the four matrices required fine adjustments when the number of levels increased beyond a certain point. The most delicate point was the choice of the initial guess to start out the algorithm, in order to avoid local minima. This is again a typical difficulty of these methods. Finally, it is important to point out that the continuity of the density φ turns out to be vital for good results, which, on the other hand, must not be surprising. We tried the algorithm with the example [33]

$$\varphi(A) = \begin{cases} 1 + |A|^2, & A \neq 0, \\ 0, & A = 0, \end{cases}$$

and most of the time the algorithm does not go out of a trivial configuration as could be easily anticipated.

REFERENCES

[1] E. Aranda and P. Pedregal, *On the Approximated Rank-One Convex Hull of a Function*, manuscript.

[2] E. Aranda and P. Pedregal, *Numerical approximation of non-homogeneous, non-convex vector variational problems*, Numer. Math., to appear.

[3] J. M. Ball and R. D. James, *Fine phase mixtures as minimizers of energy*, Arch. Rational Mech. Anal., 100 (1987), pp. 13–52.

[4] J. M. Ball and R. D. James, *Proposed experimental tests of a theory of fine microstructure and the two well problem*, Phil. Trans. R. Soc., 338 (1992), pp. 389–450.

[5] K. Battacharya, N. Firoozye, R. D. James, and R. V. Kohn, *Restrictions on microstructure*, Proc. Roy. Soc. Edinburgh Sect. A, 124 (1994), pp. 843–878.

[6] B. Brighi and M. Chipot, *Approximated convex envelope of a function*, SIAM J. Numer. Anal., 31 (1994), pp. 128–148.

[7] R. H. Byrd, P. Lu, J. Nocedal, and C. Zhu, *A Limited Memory Algorithm for Bound Constrained Optimization*, Technical Report NAM-08, Department of Electrical Engineering and Computer Science, Northwestern University, Evanston, IL, 1994.

[8] R. H. Byrd, J. Nocedal, and R. Schnabel, *Representations of Quasi-Newton Matrices and Their Use in Limited Memory Methods*, Technical Report NAM-03, Department of Electrical Engineering and Computer Science, Northwestern University, Evanston, IL, 1996.

[9] M. Chipot, *Numerical analysis of oscillations in nonconvex problems*, Numer. Math., 59 (1991), pp. 747–767.

[10] M. Chipot and C. Collins, *Numerical approximation in variational problems with potential wells*, SIAM J. Numer. Anal., 29 (1992), pp. 1002–1019.

[11] M. Chipot, C. Collins, and D. Kinderlehrer, *Numerical analysis of oscillations in multiple well problems*, Numer. Anal., 70 (1995), pp. 259–282.

[12] M. Chipot and V. Lécuyer, *Analysis and computations in the four-well problem*, in Nonlinear Analysis and Applications (Warsaw, 1994), GAKUTO Internat. Ser. Math. Sci. Appl. 7, Gakkōtosho, Tokyo, 1995, pp. 67–78.

[13] C. Collins, *Computation of twinning*, in Microstructure and Phase Transitions, IMA Vol. Math. Appl. 54, D. Kinderlehrer, R. James, M. Luskin, and J. Ericksen, eds., Springer-Verlag, New York, 1993, pp. 39–50.

[14] C. Collins and M. Luskin, *The computation of the austenitic-martensitic phase transition*, in PDE's and Continuum Models of Phase Transitions, Lecture Notes in Phys. 344, M. Rascle, D. Serre, and M. Slemrod, eds., Springer-Verlag, Berlin, 1989, pp. 34–50.

[15] C. Collins and M. Luskin, *Computational results for phase transitions in shape memory materials*, in Smart Materials, Structures, and Mathematical Issues, C. Rogers, ed., Technomic Publishing, Lancaster, PA, 1989, pp. 198–215.

[16] C. Collins and M. Luskin, *Numerical modeling of the microstructure of crystals with symmetry-related variants*, in Proceedings of the ARO US–Japan Workshop on Smart/Intelligent Materials and Systems, Technomic Publishing, Lancaster, PA, 1990, pp. 309–318.

[17] C. Collins and M. Luskin, *Optimal order error estimates for the finite element approximation of the solution of a nonconvex variational problem*, Math. Comp., 57 (1991), pp. 621–637.

[18] C. Collins, M. Luskin, and J. Riordan, *Computational images of crystalline microstructure*, in Computing Optimal Geometries, AMS Special Lectures in Mathematics and AMS Videotape Library, J. Taylor, ed., AMS, Providence, RI, 1991, pp. 16–18.

[19] C. Collins, M. Luskin, and J. Riordan, *Computational results for a two-dimensional model of crystalline microstructure*, in Microstructure and Phase Transitions, IMA Vol. Math. Appl. 54, D. Kinderlehrer, R. James, M. Luskin, and J. Ericksen, eds., Springer-Verlag, New York, 1993, pp. 51–56.

[20] C. Collins, D. Kinderlehrer, and M. Luskin, *Numerical approximation of the solution of a variational problem with a double well potential*, SIAM J. Numer. Anal., 28 (1991), pp. 321–332.

[21] B. Dacorogna, *Quasiconvexity and relaxation of non convex problems in the calculus of variations*, J. Funct. Anal., 46 (1982), pp. 102–118.

[22] B. Dacorogna, *Remarques sur les notions de polyconvexité, quasi-convexité et convexité de rang* 1, J. Math. Pures Appl. (9), 64 (1985), pp. 403–438.

[23] B. Dacorogna, *Direct Methods in the Calculus of Variations*, Springer-Verlag, New York, 1989.

[24] G. Dolzmann, *Numerical computation of rank-one convex envelopes*, SIAM J. Numer. Anal., 36 (1999), pp. 1621–1635.

[25] G. Dolzmann and N. Walkington, *Bounds on the effective energy by rank-one convexification*, in Enumath 97, H. G. Bock, F. Brezzi, R. Glowinski, G. Kanschat, Y. A. Kuznetsov, J. Périaux, and R. Rannacher, eds., World Sci., River Edge, NJ, 1998, pp. 270–277.

[26] G. Dolzmann and N. Walkington, *Estimates for numerical approximations of rank-one convex envelopes*, Numer. Math., 85 (2000), pp. 647–663.

[27] J. L. Ericksen, *Constitutive theory for some constrained elastic crystals*, Internat. J. Solids Structures, 22 (1986), pp. 951–964.

[28] D. A. French, *On the convergence of finite-element approximations of a relaxed variational problem*, SIAM J. Numer. Anal., 27 (1990), pp. 419–436.

[29] P.-A. Gremaud, *Numerical analysis of a nonconvex variational problem related to solid-solid phase transitions*, SIAM J. Numer. Anal., 31 (1994), pp. 111–127.

[30] D. Kinderlehrer, R. A. Nicolaides, and H. Wang, *Spurios Oscillations in Computing Microstructures*, preprint, Carnegie Mellon University, Pittsburgh, PA, 1993.

[31] P. Kloucek and M. Luskin, *Computational modelling of the martensitic transformation with surface energy*, Math. Comput. Modelling, to appear.

[32] R. V. Kohn, *The relaxation of a double-well energy*, Contin. Mech. Thermodyn., 3 (1991), pp. 193–236.

[33] R. V. Kohn and G. Strang, *Optimal design and relaxation of variational problems,* I, II, *and* III,, Comm. Pure Appl. Math., 34 (1987), pp. 113–137, 139–182, and 353–377.

[34] M. Kruzik, *Numerical approach to double well problems*, SIAM J. Numer. Anal., 35 (1998), pp. 1833–1849.

[35] Z. P. Li, *Numerical methods for minimizers and microstructures in nonlinear elasticity*, Math. Models Methods Appl. Sci., 6 (1996), pp. 957–975.

[36] Z. P. Li, *Simultaneous numerical approximation of microstructures and relaxed minimizers*, Numer. Math., 78 (1997), pp. 21–38.

[37] Z. P. Li, *Laminated microstructure in a variational problem with a non-rank-one connected double well potential*, J. Math. Anal. Appl., 217 (1998), pp. 490–500.

[38] B. Li and M. Luskin, *Finite element analysis of microstructure for the cubic to tetragonal transformation*, SIAM J. Numer. Anal., 35 (1998), pp. 376–392.

[39] M. Luskin, *Approximation of a laminated microstructure for a rotationally invariant, double well energy density*, Numer. Math., 75 (1997), pp. 205–221.

[40] M. Luskin, *On the computation of crystalline microstructure*, in Acta Numer. 5, Cambridge University Press, Cambridge, UK, 1996, pp. 191–257.

[41] L. Ma and N. J. Walkington, *On algorithms for nonconvex optimization in the calculus of variations*, SIAM J. Numer. Anal., 32 (1995), pp. 900–923.

[42] J. Matousek and P. Plechac, *On functional separately convex hulls*, Discrete Comput. Geom., 19 (1998), pp. 105–130.

[43] Ch. B. Morrey, *Quasiconvexity and the lower semicontinuity of multiple integrals*, Pacific J. Math., 2 (1952), pp. 25–53.

[44] R. A. Nicolaides and N. Walkington, *Computation of microstructure utilizing Young measure representations*, J. Intelligent Material Systems and Structures, 41 (1993), pp. 457–462.

[45] R. A. Nicolaides and N. Walkington, *Strong convergence of numerical solutions to degenerate variational problems*, Math. Comp., 64 (1995), pp. 117–127.

[46] R. A. Nicolaides, N. Walkington, and H. Wang, *Numerical methods for a nonconvex optimization problem modeling martensitic phase transitions*, SIAM J. Sci. Comput., 18 (1997), pp. 1122–1141.

[47] P. Pedregal, *Laminates and microstructure*, European J. Appl. Math., 4 (1993), pp. 121–149.

[48] P. Pedregal, *Numerical approximation of parametrized measures*, Numer. Funct. Anal. Optim., 16 (1995), pp. 1049–1066.

[49] P. Pedregal, *On the numerical analysis of nonconvex variational problems*, Numer. Math., 74 (1996), pp. 325–336.

[50] P. Pedregal, *Parametrized Measures and Variational Principles*, Birkhäuser, Basel, 1997.

[51] T. Roubícek, *Numerical approximation of relaxed variational problems*, J. Convex Anal., 3 (1996), pp. 329–347.

[52] T. Roubícek, *Relaxation in Optimization Theory and Variational Calculus*, W. De Gruyter, Berlin, 1997.

[53] V. Sverak, *Rank-one convexity does not imply quasiconvexity*, Proc. Roy. Soc. Edinburgh Sect. A, 120 (1992), pp. 185–189.

[54] L. Tartar, *Some remarks on separately convex functions*, in Microstructure and Phase Transitions, IMA Vol. Math. Appl. 54, D. Kinderlehrer, R. James, M. Luskin, and J. Ericksen, eds., Springer-Verlag, New York, 1993, pp. 191–204.

SIAM J. SCI. COMPUT.
Vol. 22, No. 5, pp. 1791–1813
© 2001 Society for Industrial and Applied Mathematics

A MOVING MESH METHOD FOR ONE-DIMENSIONAL HYPERBOLIC CONSERVATION LAWS*

JOHN M. STOCKIE†, JOHN A. MACKENZIE‡, AND ROBERT D. RUSSELL§

Abstract. We develop an adaptive method for solving one-dimensional systems of hyperbolic conservation laws that employs a high resolution Godunov-type scheme for the physical equations, in conjunction with a moving mesh PDE governing the motion of the spatial grid points. Many other moving mesh methods developed to solve hyperbolic problems use a fully implicit discretization for the coupled solution-mesh equations, and so suffer from a significant degree of numerical stiffness. We employ a semi-implicit approach that couples the moving mesh equation to an efficient, explicit solver for the physical PDE, with the resulting scheme behaving in practice as a two-step predictor-corrector method. In comparison with computations on a fixed, uniform mesh, our method exhibits more accurate resolution of discontinuities for a similar level of computational work.

Key words. moving mesh, adaptivity, equidistribution, shock capturing, hyperbolic conservation laws, finite volume methods

AMS subject classifications. 65M06, 65M50, 76L05, 35L65, 35L67

PII. S1064827599364428

1. Introduction.

In this paper, we present an adaptive algorithm for computing solutions to one-dimensional systems of hyperbolic conservation laws. We consider problems of the form

$$\boldsymbol{q}_t + \boldsymbol{f}(\boldsymbol{q})_x = 0, \tag{1.1a}$$

$$\boldsymbol{q}(x,0) = \boldsymbol{q}_0(x), \tag{1.1b}$$

where $\boldsymbol{q}(x,t) \in \mathbb{R}^m$ is an m-vector, $a \le x \le b$, and $t \ge 0$. The system is "hyperbolic" in the sense that the Jacobian matrix $\partial \boldsymbol{f}/\partial \boldsymbol{q}$ has real eigenvalues and is diagonalizable with m linearly independent eigenvectors. Such problems are characterized by moving discontinuities (i.e., fronts or shocks) that separate regions of flow where the solution is smooth. The major challenge in solving hyperbolic systems numerically is to capture the discontinuous solutions with sufficient accuracy while also keeping the computational cost within acceptable limits.

The vast majority of numerical methods for solving hyperbolic problems have been developed for fixed, uniform grids. An important class of such methods is based on the Godunov scheme [16], which is a first order, finite volume method that employs the exact solution to local Riemann problems at cell interfaces to enhance the resolution of discontinuities. Accuracy can be further improved by using higher order variants, such as the flux- and slope-limiter methods reviewed in [28, 31], and computational cost is

*Received by the editors November 22, 1999; accepted for publication (in revised form) June 8, 2000; published electronically January 19, 2001. This research was supported by fellowships from the Pacific Institute for the Mathematical Sciences and the MITACS National Centre of Excellence and by a research grant from NSERC.

http://www.siam.org/journals/sisc/22-5/36442.html

†Department of Mathematics and Statistics, University of New Brunswick, Fredericton, New Brunswick, Canada, E3B 5A3 (stockie@math.unb.ca). This work was performed while this author was a postdoctoral fellow at Simon Fraser University.

‡Department of Mathematics, University of Strathclyde, Glasgow, Scotland, G1 1XH (j.a.mackenzie@strath.ac.uk).

§Department of Mathematics and Statistics, Simon Fraser University, Burnaby, British Columbia, Canada, V5A 1S6 (rdr@cs.sfu.ca).

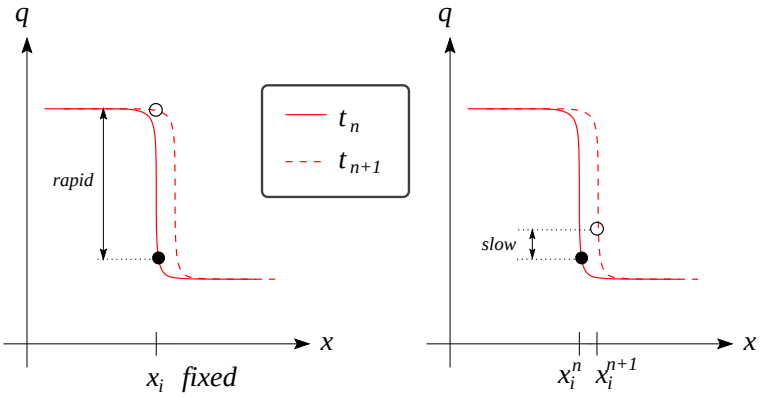


FIG. 1.1. *A comparison of the variation in solution from one time step to the next on fixed and moving meshes.*

reduced by linearizing the equations and solving approximate Riemann problems in each cell instead.

The discontinuities that are characteristic of hyperbolic problems typically move in time, and so the solution at a particular point in space can change very rapidly (see Figure 1.1). As a result, computations on a fixed, uniform spatial mesh can require that the time step be extremely small in order to reduce the error arising from the time variation in the solution at points near the front. On the other hand, it is possible to take a much larger time step when the solution is smooth and does not exhibit such large variation from one mesh point to the next. In principle, therefore, it is desirable to employ a nonuniform mesh that is sparse in regions where the solution is smooth and more concentrated near discontinuities. Since the steep fronts are not stationary, it is attractive to allow the mesh points to move in time so that fine grid resolution can be maintained near discontinuities, thereby attaining a balance between accuracy and efficiency.

There has been some success in solving hyperbolic problems on adaptive spatial meshes, starting with Harten and Hyman [17] who demonstrated how to extend a Godunov scheme to handle moving grids in one dimension. They employed a static regridding technique, in which the solution and mesh are evolved separately, with the mesh points updated in each time step to explicitly track discontinuities. Another static refinement strategy that has proven very successful, especially for higher dimensional problems, is the adaptive mesh refinement method [3] in which the mesh is refined locally based on some measure of the solution error using Cartesian subgrids. Biswas, Flaherty, and Arney [4] combined both mesh movement and local mesh refinement in a finite element framework.

In contrast with static refinement techniques is the second major class of adaptive methods based on dynamic refinement, in which one explicitly derives an equation governing the spatial mesh, which moves mesh points naturally to where they are most needed. The mesh equation is often derived from an *equidistribution principle*, which attempts to equally distribute some measure of solution error over the spatial domain. This approach was used to solve one-dimensional hyperbolic problems in [30] and [11], employing a flux splitting method for discretizing the physical PDE. The major disadvantage to this technique is that the coupling between the solution and mesh equations is nonlinear, often requiring a Newton iteration in each time

step, which can be very costly. This problem is further exacerbated by the dense clustering of mesh points near discontinuities, which degrades the convergence of the iteration.

In many moving mesh methods it has been found necessary to introduce an additional artificial viscosity term of the form $\mu \boldsymbol{q}_{xx}$ into (1.1), and solve the resulting parabolic system instead. This approach has been taken in connection with spatial discretizations based on finite differences [23, 29, 12], finite elements [15, 22], and collocation [19]. However, even with the viscous regularization, there are still convergence problems associated with taking the hyperbolic limit $\mu \to 0$ that can seriously degrade performance [15].

In this work, we propose an adaptive method that combines the flexibility and accuracy afforded by a dynamically moving mesh with the increased shock resolution capability of a Godunov-type scheme. The resulting method has the potential to reduce the cost of the moving mesh component of the algorithm significantly by eliminating the need for such a high concentration of grid points near discontinuities. We employ the moving mesh PDE (MMPDE) approach of Huang, Ren, and Russell [18] in which the number of grid points is constant and the points move throughout the domain, subject to a time-dependent PDE. The main difference from many of the other moving grid methods mentioned earlier is that the MMPDE incorporates a temporal smoothing term which improves the performance of the solution-mesh iteration.

For the physical PDE, we use the wave propagation method introduced by LeVeque [21] and implemented in the software package CLAWPACK [20]. The resulting method is similar to that of Chen [9], wherein a Godunov scheme was coupled with an MMPDE in a fully implicit time discretization based on the method of lines. However, we construct a more efficient semi-implicit scheme (similar to that developed in [29] for parabolic problems) that behaves as an explicit, predictor-corrector step for the test problems considered here. Furthermore, our method satisfies a discrete conservation principle which may not be the case for implicit discretizations based on a straightforward method-of-lines approach (see [14, 9, 12]). Our strategy is based on the principle that there is no reason to solve the physical PDE and mesh equation with the same spatial discretization or to the same level of accuracy, since they are equations of different type (one hyperbolic, one parabolic) and have fundamentally different interpretations (one physical, and the other an artificial construct).

We begin in section 2 with a description of the numerical method, including details of the wave propagation scheme for the hyperbolic conservation law, the discretization of the moving mesh equation, and the iteration that couples the two together. Section 3 presents several possible monitor functions that are designed specifically to resolve discontinuous solutions and shows how a monitor function can be generalized for systems of conservation laws where multiple fronts arise. In section 4, numerical tests are performed on several problems encompassing both scalar conservation laws and systems, to demonstrate the accuracy, efficiency, and robustness of the moving mesh method. Throughout this paper two major issues of concern are choosing a monitor function that is tailored to capturing discontinuous flow features, and temporal smoothing for controlling mesh motion.

2. The numerical method.

2.1. The physical PDE: Godunov's method on a moving mesh. We first describe the finite volume discretization of the hyperbolic conservation law (1.1a) on a

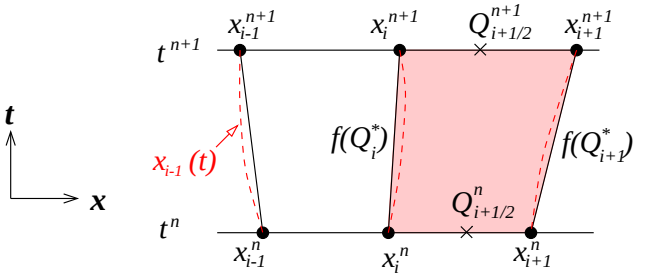


FIG. 2.1. *A computational cell on an adaptive spatial mesh. The dashed lines show the actual, curved, grid trajectories, which are approximated by straight lines in the xt-plane.*

nonuniform, moving mesh derived by Fazio and LeVeque in [13].[1] Consider a sequence of times t^n for $n = 0, 1, \ldots$, where the time steps $\Delta t^n = t^{n+1} - t^n$ need not be equal. Suppose that the spatial domain $[a, b]$ is subdivided using a set of points which move in time and are parameterized by $x_i(t) = x(\xi_i, t)$, where $\xi_i = \frac{i}{N}$ are the equally spaced computational coordinates, and

$$a \equiv x_0 < x_1(t) < x_2(t) < \cdots < x_{N-1}(t) < x_N \equiv b.$$

Figure 2.1 depicts a typical computational cell with grid points evolving from time level t^n to $t^{n+1} = t^n + \Delta t^n$, and with $x_i^n = x_i(t^n)$. In the following discussion, we develop the method in terms of the scalar quantity q to simplify notation, although the results extend easily to vectors.

In the *Godunov scheme* [16], the solution is assumed to be piecewise constant on each subinterval $[x_i, x_{i+1}]$, and the discrete solution is taken to represent the average value of the actual solution along the lower cell boundary,

$$Q_{i+1/2}^n = \frac{1}{\Delta x_{i+1/2}^n} \int_{x_i^n}^{x_{i+1}^n} q(s, t^n)\, ds, \tag{2.1}$$

where $\Delta x_{i+1/2}^n = x_{i+1}^n - x_i^n$ is the local mesh spacing. We then integrate the conservation law (1.1a) across the computational cell to obtain the formula

$$\kappa_{i+1/2}^{n+1} Q_{i+1/2}^{n+1} = \kappa_{i+1/2}^n Q_{i+1/2}^n - \frac{\Delta t^n}{\Delta \xi}\Big[(f(Q_{i+1}^*) - \dot{x}_{i+1}^n Q_{i+1}^*) - (f(Q_i^*) - \dot{x}_i^n Q_i^*)\Big], \tag{2.2}$$

where $\Delta \xi = \frac{1}{N}$ and $\kappa_{i+1/2}^n \doteq \Delta x_{i+1/2}^n / \Delta \xi$. The numerical flux $f(Q_i^*)$ is an approximation of the actual flux along the left slanted boundary of the cell and is computed using the solution to the local Riemann problem arising at the interface between the constant states $Q_{i-1/2}^n$ and $Q_{i+1/2}^n$. The intermediate state, Q_i^*, is actually the solution to the Riemann problem that lies along the straight-line characteristic $(x - x_i^n)/(t - t^n) = \dot{x}_i^n$. Here, we have assumed that the time derivative $\dot{x}_i = \partial x(\xi_i, t)/\partial t$ is constant over the time interval $[t^n, t^{n+1}]$, so that the edges of the cell are straight lines. The main

[1] Fazio and LeVeque used a simple moving mesh strategy for the Euler equations in which the contact line is tracked explicitly and a piecewise uniform mesh is fit to either side of the discontinuity. Our moving mesh approach is designed to capture discontinuities naturally as part of the solution process and is therefore much more flexible.

difference between this and Godunov's scheme on a fixed mesh is the appearance of additional $\dot{x}$ terms in (2.2) that arise due to the movement of the mesh points.

The method in the form (2.2) is nonconservative but can be modified to get a conservative discretization. Again assuming $\dot{x}_i$ is constant in a cell, we can use $x_i^{n+1} = x_i^n + \Delta t^n \dot{x}_i^n$ to obtain

$$\kappa_{i+1/2}^n Q_{i+1/2}^n = \kappa_{i+1/2}^{n+1} Q_{i+1/2}^n - \frac{\Delta t^n}{\Delta \xi}(\dot{x}_{i+1}^n - \dot{x}_i^n) Q_{i+1/2}^n.$$

Substituting this expression into (2.2) yields the discrete system

$$\begin{aligned}\kappa_{i+1/2}^{n+1} Q_{i+1/2}^{n+1} = \kappa_{i+1/2}^{n+1} Q_{i+1/2}^n - \frac{\Delta t^n}{\Delta \xi}\Big[&(f(Q_{i+1}^*) - \dot{x}_{i+1}^n (Q_{i+1}^* - Q_{i+1/2}^n)) \\ &- (f(Q_i^*) - \dot{x}_i^n (Q_i^* - Q_{i+1/2}^n))\Big].\end{aligned} \tag{2.3}$$

The advantage to this form is that the quantity $\sum_i \kappa_{i+1/2}^n Q_{i+1/2}^n$ is conserved with n and constant states Q are preserved.

For reasons which will become clear shortly, we write the Godunov scheme in *wave propagation form*:

$$Q_{i+1/2}^{n+1} = Q_{i+1/2}^n - \frac{\Delta t^n}{\kappa_{i+1/2}^{n+1} \Delta \xi}\left(\mathcal{A}^+ \Delta Q_i^n + \mathcal{A}^- \Delta Q_{i+1}^n\right) \tag{2.4}$$

with

$$\begin{aligned}\mathcal{A}^+ \Delta Q_i^n &= f(Q_{i+1/2}^n) - f(Q_i^*) - \dot{x}_i^n (Q_{i+1/2}^n - Q_i^*), \qquad \text{and} \\ \mathcal{A}^- \Delta Q_i^n &= f(Q_i^*) - f(Q_{i-1/2}^n) - \dot{x}_i^n (Q_i^* - Q_{i-1/2}^n).\end{aligned}$$

Here, $\mathcal{A}^+ \Delta Q_i^n$ is the right-going flux difference from solving the Riemann problem between $Q_{i-1/2}^n$ and $Q_{i+1/2}^n$ and models the combined effect on the cell average $Q_{i+1/2}^n$ of waves entering from the left edge. Similarly, $\mathcal{A}^- \Delta Q_{i+1}^n$ is the left-going flux difference from the Riemann problem between $Q_{i+1/2}^n$ and $Q_{i+3/2}^n$ and models the combined effect of all waves entering the cell from the right.

The Godunov scheme can be very expensive, particularly for systems of conservation laws, where the Riemann problem at each cell interface requires the solution of a nonlinear system of equations. In practice, it is possible to solve the Riemann problem approximately based on the linearized system

$$\boldsymbol{q}_t + A \cdot \boldsymbol{q}_x = 0 \tag{2.5}$$

with A an $m \times m$ matrix. The well-known approximate Riemann solver of Roe [25] is the solution to such a linearization and yields a set of m wave speeds λ_i^p and jumps $\mathcal{W}_i^p$ across each wave for $p = 1, \ldots, m$. Using the wave decomposition from the Roe solver, we can write

$$\mathcal{A}^+ \Delta Q_i = \sum_{p=1}^m (\widetilde{\lambda}_i^p)^+ \mathcal{W}_i^p, \tag{2.6a}$$

$$\mathcal{A}^- \Delta Q_i = \sum_{p=1}^m (\widetilde{\lambda}_i^p)^- \mathcal{W}_i^p, \tag{2.6b}$$

where $(\widetilde{\lambda}_i^p)^+ = \max(0, \widetilde{\lambda}_i^p)$, $(\widetilde{\lambda}_i^p)^- = \min(0, \widetilde{\lambda}_i^p)$, and $\widetilde{\lambda}_i^p = \lambda_i^p - \dot{x}_i^n$ are "shifted" wave speeds, incorporating the influence of the moving mesh. One of the major advantages of using this type of finite volume formulation is the natural way in which mesh motion is incorporated into the discrete equations, appearing only in the wave speeds. This should be contrasted with other methods based on finite differences that explicitly introduce terms of the form $q_x\dot{x}$, which can lead to problems with stability unless discretized carefully [23].

Godunov's method is only first order accurate in space, and so it suffers from a high degree of artificial dissipation which leads to significant smearing of the sharp fronts as the solution is evolved. To increase the accuracy of the discretization, one can use a high resolution flux correction (see [13, 21] for details).

Because the scheme is explicit in time, stability requires the time step to satisfy the *Courant–Friedrichs–Levy (CFL) condition* $\nu \leq 1$, where the CFL number is

$$\nu = \Delta t \max_{i,p} \left\{ \left| \frac{(\widetilde{\lambda}_i^p)^+}{\Delta x_{i+1/2}} \right|, \left| \frac{(\widetilde{\lambda}_{i+1}^p)^-}{\Delta x_{i+1/2}} \right| \right\}. \tag{2.7}$$

This inequality requires that the time step be small enough that waves from neighboring cells do not intersect.

It is important to distinguish the difference between the CFL condition for a moving mesh method and that arising from a *fixed* nonuniform grid, which is the same as (2.7) except that $\widetilde{\lambda}_i^p$ is replaced by the unshifted wave speeds λ_i^p. On a fixed mesh, $\Delta x_{i+1/2}$ can be very small, leading to a very strict global requirement on the time step. In contrast, the moving mesh method allows for a much less restrictive CFL condition when the shifted wave speeds are close to zero; i.e., provided that mesh points move at approximately the same speed as solution discontinuities, $\dot{x}_i \approx \lambda_i^p$. The possibility of relaxing the global CFL restriction for explicit calculations on nonuniform meshes is one of the major advantages of using a moving mesh for hyperbolic problems.

2.2. The MMPDE. Now we consider the mesh computation and for this formulate a moving mesh PDE that is based on an equidistribution principle. Following [18], we require that the transformation $x(\xi, t)$ from physical to computational coordinates satisfy

$$\int_a^{x(\xi,t)} M(s,t)\, ds = \xi \cdot \int_a^b M(s,t)\, ds. \tag{2.8}$$

The function $M(x,t) > 0$ is called the *monitor function* and can be the most important component of the adaptive mesh algorithm. In principle, M can be any appropriately chosen measure of the numerical error in the solution of the physical PDE. However, an advantage of the moving mesh approach is that M can be chosen to capitalize on some underlying property of the solution (for example, self-similarity or scaling invariance [5]). This is a feature that can be exploited in order to place mesh points precisely where they are needed in order to achieve optimal accuracy.

It is possible to discretize the integral form (2.8) directly, leading to a set of algebraic equations for the mesh locations which can be extremely difficult to solve efficiently when coupled with the physical PDE. Instead we take the MMPDE approach used by Huang, Ren, and Russell [18], wherein a PDE describing the moving mesh points is obtained by taking the time derivative of (2.8) and introducing temporal smoothing. There are a large number of possible MMPDEs, one being MMPDE4

of [18]:[2]

$$\frac{\partial}{\partial \xi}\left(M\frac{\partial \dot{x}}{\partial \xi}\right) = -\frac{1}{\tau}\frac{\partial}{\partial \xi}\left(M\frac{\partial x}{\partial \xi}\right). \tag{2.9}$$

The parameter τ can be thought of as the time scale over which the mesh relaxes towards equidistribution.

A commonly used form of M is the arclength monitor function

$$M = \sqrt{1+|q_x|^2}. \tag{2.10}$$

The corresponding centered finite difference approximation at cell midpoints is

$$M_{i+1/2} = \sqrt{1+\left|\frac{\overline{Q}_{i+1}-\overline{Q}_i}{x_{i+1}-x_i}\right|^2}, \tag{2.11}$$

where $\overline{Q}_i = (Q_{i+1/2}\Delta x_{i-1/2} + Q_{i-1/2}\Delta x_{i+1/2})/(\Delta x_{i+1/2} + \Delta x_{i-1/2})$ is a weighted average of cell-centered solution values, located at cell edges. Notice that M is largest where the solution changes most rapidly, and as a result the mesh equation (2.9) serves to concentrate grid points in regions with large solution gradients.

It is well known that some sort of smoothing of the mesh is required in order to maintain reasonable accuracy in the computation of a solution on an adaptive mesh. Rather than smoothing the mesh itself, a commonly applied technique in the moving mesh framework is to replace the monitor function in the equations above by a regularized version $\widetilde{M}$ given by

$$\widetilde{M}_{i+1/2} = \sqrt{\sum_{k=i-i_p}^{i+i_p} M_{k+1/2}^2 \left(\frac{\gamma_s}{1+\gamma_s}\right)^{|k-i|} \Bigg/ \sum_{k=i-i_p}^{i+i_p} \left(\frac{\gamma_s}{1+\gamma_s}\right)^{|k-i|}}. \tag{2.12}$$

$\widetilde{M}_{i+1/2}$ can be thought of as a weighted average of the neighboring $2i_p+1$ values of M (with weighting factors determined by the parameter $\gamma_s > 0$) that serves to eliminate local oscillatory or nonsmooth behavior that may arise from solving the MMPDE.

We discretize the MMPDE using centered finite differences in space, yielding

$$M_{i+1/2}\left(\dot{x}_{i+1}-\dot{x}_i\right) - M_{i-1/2}\left(\dot{x}_i-\dot{x}_{i-1}\right) = -\frac{E_i}{\tau}, \tag{2.13a}$$

where E_i is a centered approximation to the term on the right-hand side of (2.9) given by

$$E_i = M_{i+1/2}\left(x_{i+1}-x_i\right) - M_{i-1/2}\left(x_i-x_{i-1}\right). \tag{2.13b}$$

Since the monitor function depends on the solution values $Q_{i+1/2}$, the discrete physical equation (2.4) and (2.13a) form a coupled, nonlinear system of equations to be solved in each time step. In one-dimensional problems, it is common to employ a fully implicit time discretization and solve the resulting system of stiff ODEs using a package such as DASSL [24]. In two dimensions this procedure becomes very costly, and other approaches are needed (see [6], for example). We wish to construct a time-stepping procedure that preserves the conservation properties of the physical PDE, an issue addressed in the following section.

[2]Another MMPDE, MMPDE6, performs better for problems in which there are viscous shocks [22] or blow-up [5]. Its success is attributable to a certain scaling invariance property. However, we have found in numerical experiments that MMPDE6 is unsuitable for solving hyperbolic problems, which may be related to some alternate form of scaling invariance.

2.3. Time integration. Here we describe an algorithm for solving the discrete equations (2.4) and (2.13). By itself, the wave propagation scheme is explicit, but when coupled with the mesh equation, the system becomes implicit due to the dependence of the mesh equation on Q through the monitor function.

In our experience, a straightforward explicit discretization in time will lead to instabilities, and therefore some form of implicit time differencing is required (see also [23, 15]). Intuitively, it is easy to see the need for some form of iteration: if we were to employ a fully explicit algorithm, basing the mesh locations on the solution for the previous time step, then grid points can lag behind the solution, leading to serious violations of the CFL condition. However, a fully implicit scheme, wherein the coupled nonlinear system is solved in each time step, is far too expensive for practical calculations. Our method is similar to the fully explicit, predictor-corrector step constructed in [29], except that the step is iterated. Our aim in constructing a scheme is to leave the solution step for the physical PDE unaltered, so that we can employ the conservative, high resolution algorithms in CLAWPACK, and build around it a fixed point iteration on the mesh.

We propose the following algorithm.

Moving Mesh Algorithm

1. *Let* $n = 0$.
2. *Given an initial solution* Q^0 *at time* $t = t^0$*, equidistribute the mesh exactly using a discretization of the exact equidistribution principle* $(Mx_\xi)_\xi = 0$ *(corresponding to* $\tau = 0$*).*
3. *Step the solution to time level* $n+1$ *using the following iteration:*
   (a) *Let* $m = 0$*. Take a guess at the new mesh positions using* $x_i^{n+1,0} = x_i^n + \Delta t^n \dot{x}_i^n$*, unless this leads to mesh crossings, in which case we take the conservative guess* $x_i^{n+1,0} = x_i^n$*.*
   (b) *Use* CLAWPACK *to obtain an initial guess* $Q^{n+1,0}$ *for the solution at time level* $n+1$ *using the mesh* $x_i^{n+1,0}$*.* CLAWPACK *returns an adaptively-chosen time step* Δt^n *which is used for the remainder of the iteration.*
   (c) *Iterate on* m*:*
      i. *Compute the raw and smoothed monitor function values,* $M_{i+1/2}$ *and* $\widetilde{M}_{i+1/2}$*, and the quantity* E_i *from* (2.13b)*, all based on the current solution iterate* $Q_{i+1/2}^{n+1,m}$*.*
      ii. *Employ a Crank–Nicolson discretization of* (2.13a) *for updating the mesh:*

$$\widetilde{M}_{i+1/2}^{n+1,m}(x_{i+1}^{n+1,m+1} - x_i^{n+1,m+1}) - \widetilde{M}_{i-1/2}^{n+1,m}(x_i^{n+1,m+1} - x_{i-1}^{n+1,m+1})$$

$$= \widetilde{M}_{i+1/2}^{n+1,m}(x_{i+1}^n - x_i^n) - \widetilde{M}_{i-1/2}^{n+1,m}(x_i^n - x_{i-1}^n) - \frac{\Delta t^n}{2\tau}\left[E_i^{n+1,m} + E_i^n\right],$$

      for which a tridiagonal system is solved for the unknowns $x_i^{n+1,m+1}$*.*
      iii. *Use* CLAWPACK *to obtain the new solution approximation* $Q_i^{n+1,m+1}$*, using mesh positions* $x_i^{n+1,m+1}$*, mesh velocities* $\dot{x}_i^{n+1} = (x_i^{n+1,m+1} - x_i^n)/\Delta t^n$*, and a CFL restriction of* $\nu \le 0.9$*.*
      iv. *If* $\|x^{n+1,m+1} - x^{n+1,m}\|_1 > TOL$*, then increment* m*, and go to step* 3(c)(i)*.*

4. *Once the iteration has converged, increment n and begin a new time step in step* 3.

Some of the key aspects of the algorithm worth noting are that

- the physical PDE is integrated explicitly in time using a conservative method;
- the convergence tolerance is applied on the mesh only;
- there is only a single parameter (the temporal smoothing, τ) needed to control the mesh motion;
- because the physical PDE is solved using the general purpose solver CLAWPACK, the method is easy to generalize to other problems.

These features combine together to yield a method that to our knowledge is distinct from any other adaptive approach that has appeared in the literature.

3. Monitor functions for resolving discontinuities. The monitor function should in principle be chosen so that it represents some measure of the error in the computed solution in an appropriate norm. At points where the error is large, M should also be large so that mesh points will tend to concentrate in those areas where higher resolution is needed. While solution arclength (2.10) concentrates mesh points where the solution has large gradients and gives excellent results for many parabolic problems, we have found that it is inappropriate for hyperbolic equations. Particularly when high resolution Godunov schemes are used, computed discontinuous flow features can be very steep, and so the MMPDE may drive more points than necessary into the neighborhood of these discontinuities. Several investigations have already been made into how the arclength monitor function can be modified to avoid excessive clustering of points and so control the conditioning of the mesh equation [10, 1, 22]. Our objective in this section is to obtain regularized variants of the arclength monitor, which are still large where discontinuities arise, but not so large that they over-resolve steep layers.

One way to modify the arclength monitor function so as to balance the number of points inside and outside a steep internal layer is to introduce a "regularizing factor" α in the following manner:[3]

$$M = \sqrt{1 + \frac{1}{\alpha}\,|q_x|^2}. \tag{3.1}$$

The factor α allows one to reduce the magnitude of the monitor function in situations where $|q_x|$ is very large, thereby avoiding over-resolution of steep layers, while also ensuring that M still retains a significant peak near these discontinuities. Beckett and Mackenzie [1] applied this type of regularization in the context of singularly perturbed boundary value problems, and using a monitor function based on curvature, $M = 1 + \frac{1}{\alpha}|q_{xx}|^{1/m}$. Taking α to be any constant greater than one in (3.1) will tend to move points out of a steep front into regions where the solution is smooth. However, if we are to construct a monitor function that will give reasonably consistent results for shocks of arbitrary steepness, then α will have to depend on the solution.

A scaling of the arclength function using the maximum solution value (with $\alpha \propto \max_x |q|^2$) was applied in [11] to eliminate large variations in the solution components

[3]Note that the monitor functions $M = \sqrt{\alpha + |q_x|^2}$ and $M = \sqrt{1 + |q_x|^2/\alpha}$ give identical results because of the invariance of MMPDE4 in (2.9) under the scaling $M \mapsto cM$, where c is a constant. MMPDE6, incidentally, is not invariant under this scaling.

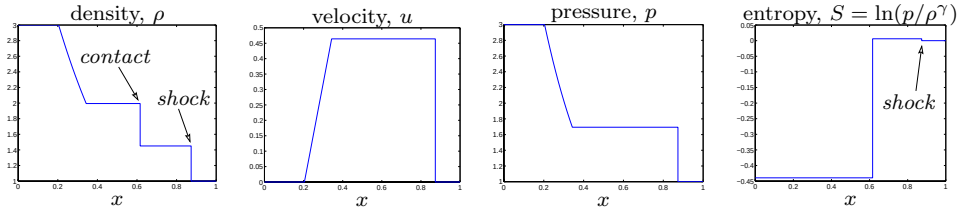


FIG. 3.1. *Plots of density, velocity, pressure, and entropy for a solution to the Euler equations. In each plot, the rarefaction wave is on the left, the shock on the right, and the contact line lies in between.*

for systems of conservation laws. This monitor will still over-resolve discontinuous fronts, and so we consider instead a scaling based on the maximum derivative value,

$$\alpha = \alpha^{max} \doteq \max_x |q_x|^2/\beta, \tag{3.2}$$

where the parameter $\beta > 1$ controls the concentration of mesh points. A related regularization, used in [1], scaled the derivative term in (3.1) by its average value over the domain using

$$\alpha = \alpha^{avg} \doteq \frac{1}{|b-a|} \int_a^b |q_x|^2 \, dx. \tag{3.3}$$

The advantage to this type of monitor function is that there is no free parameter needed for mesh control.

3.1. Generalization to systems of equations. The discussion of monitor functions thus far has been restricted to the case where the unknown function is a scalar. The definition extends naturally to systems of equations in which $\boldsymbol{q}$ is an m-vector. Perhaps the most obvious approach is to replace the absolute value $|q_x|$ with the vector 2-norm $\|\boldsymbol{q}\|$ in (3.1). However, in problems such as the Euler equations of gas dynamics, the solution components can take on values that differ widely in magnitude, in which case it makes no sense to weight the individual components equally. Furthermore, shocks are characterized by jumps in all three components, whereas contact lines appear as discontinuities in the density only, as pictured in Figure 3.1. As a result shocks are weighted more heavily in the calculation of the monitor function and contact lines are essentially ignored. This is undoubtedly what limits the accuracy of contact line resolution in other moving mesh computations such as those reported in [8, 22].

In order to obtain a monitor function that is better able to capture all discontinuous flow features, some form of rescaling must be performed on the solution components. Dorfi and Drury [11] used a weighted 2-norm of the solution in which each component $q^{(j)}$ is scaled by a factor $\alpha^{(j)} \approx \max_x |q^{(j)}|^2$, and as a result the contribution of each component to the monitor function is the same.

Rather than using a simple scaled vector norm, we construct two distinct monitor functions, one tailored to recognizing shocks and one to contact lines, and scale each separately. Since the velocity, u, is discontinuous only across shocks, a natural choice for constructing a "shock monitor" is the following:

$$M^s = \sqrt{1 + \beta \left(\frac{|u_x|}{\max_x |u_x|} \right)^2}, \tag{3.4}$$

although pressure could equally well be used in place of velocity. We choose to regularize the monitor function using α^{max}, since then the maximum magnitude of M is independent of the shock steepness or the magnitude of the velocity component. It is then possible to normalize the monitor functions so that different discontinuities are given equal weight, something which is not possible with the averaged regularization parameter α^{avg}.

In a similar manner, we construct a "contact monitor" function defined in terms of the entropy $S = \ln(p/\rho^\gamma)$ as

$$M^c = \sqrt{1 + \beta \left(\frac{|S_x|}{\max_x |S_x|} \right)^2}. \tag{3.5}$$

While the entropy is discontinuous across both shocks and contact lines (see Figure 3.1) and so is not an ideal choice for a contact monitor, entropy typically has a much smaller jump across a shock than a contact line. Therefore, (3.5) should still be a good measure of the location of contact discontinuities.

The moving mesh algorithm requires a single monitor function, which we define using the simple convex combination

$$M^{cs} = \theta M^c + (1 - \theta) M^s. \tag{3.6}$$

In all numerical simulations performed herein, we weight the two classes of discontinuity equally and take $\theta = \frac{1}{2}$. The resolution of the shock and contact line is fairly insensitive to the choice β in (3.4) and (3.5) provided it is taken large enough, and in practice a value of $\beta = 100$ has proven to give satisfactory results.

4. Numerical results. In this section, we will focus on three test problems: the inviscid Burgers equation; the Buckley–Leverett equation (which is distinguished by a nonconvex flux function); and the Euler equations of gas dynamics (a system of three hyperbolic conservation laws). We evaluate the various monitor functions introduced in section 3 by comparing accuracy and computational cost. We also investigate the importance of the temporal smoothing parameter τ appearing in the moving mesh equation and whether there are any particular issues associated with solving problems that have discontinuous solutions. The solution quality and algorithm performance are much less sensitive to the level of spatial smoothing, and so we fix the parameters $i_p = 4$ and $\gamma_s = 2$ in (2.12) for the remainder.

Computational costs are reported as CPU times in seconds measured on a Sun SPARCstation-20, and in all of our simulations the overhead cost associated directly with computing the mesh amounted to at most 25% of the total CPU time. As a result, any differences in total cost between the fixed and moving mesh results can be attributed primarily to differences in the time step in conjunction with any additional moving mesh iterations required in each time step.

Errors are reported relative to an "exact" solution which is actually a finely resolved numerical approximation obtained from a fixed mesh computation with $N = 2500$. An appropriate norm for measuring errors in discontinuous solutions is the L^1-norm, which we approximate using the formula

$$\|Q - q\|_1 = \sum_{i=1}^{N} \left| Q_{i+1/2} - q(x_{i+1/2}) \right| \cdot \Delta x_{i+1/2},$$

where Q is the computed solution and q the exact solution.

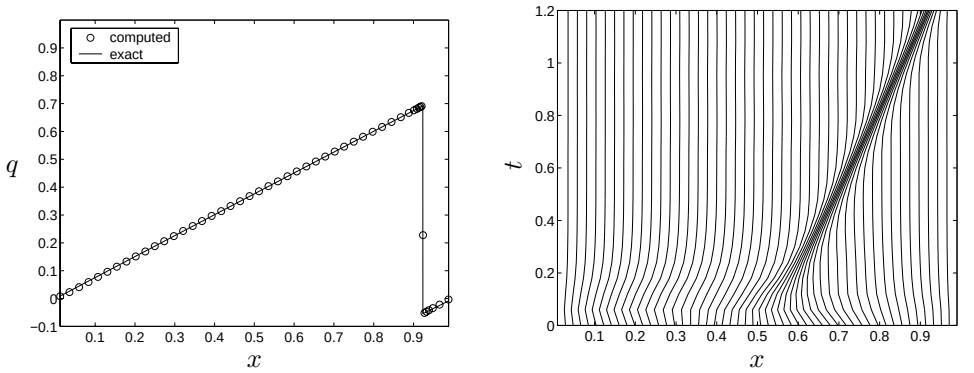


FIG. 4.1. *Moving mesh solution to Burgers's equation at time $t = 1.2$ sec. On the left is the solution and on the right are the mesh contours (α^{avg}, $\tau = 0.05$, $N = 50$).*

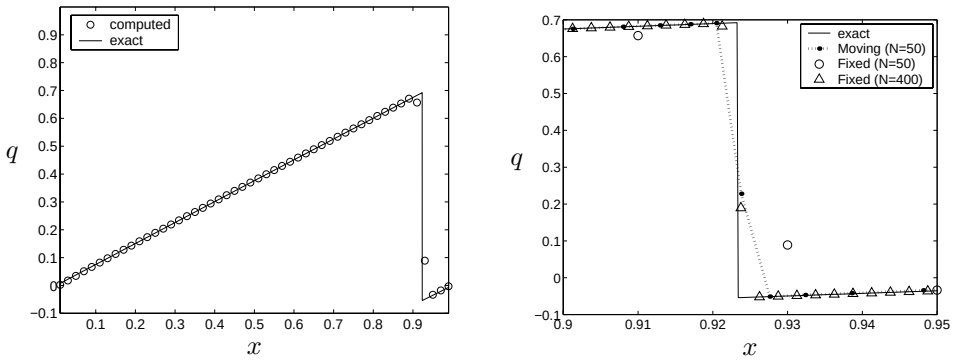


FIG. 4.2. *Fixed mesh solution with $N = 50$ points (left) and a blow-up of various fixed and moving mesh calculations in the neighborhood of the shock (right).*

4.1. Inviscid Burgers equation. Our first test of the moving mesh method is the inviscid Burgers equation

$$q_t + \left(\frac{1}{2}q^2\right)_x = 0 \tag{4.1}$$

for $0 \le x \le 1$, and $t \ge 0$, with periodic boundary conditions and an initial solution profile given by $q_0(x) = \sin(2\pi x) + \frac{1}{2}\sin(\pi x)$. The solution propagates to the right, steepening until the *singularity time* $t_s = 64/(129\pi) \approx 0.15792$, at which point a shock forms.

The solution up to time $t = 1.2$ is displayed in Figure 4.1 for a moving mesh calculation with $N = 50$, $\tau = 0.05$, and regularization parameter α^{avg}, while the fixed grid solution with the same number of mesh points is given in Figure 4.2 for comparison. It is evident that the moving mesh computation yields considerably better shock resolution than the fixed mesh method because of the clustering of mesh points near the shock. The ability of the mesh to capture and follow the moving shock is demonstrated by the mesh contour plot in Figure 4.1.

A more quantitative comparison of solution errors is afforded by Table 4.1, which lists the L^1 error in the solution for various fixed and moving mesh calculations. The

TABLE 4.1

Comparison of various fixed and moving grid solutions for Burgers's equation. The NT column reports the number of time steps and $\mathcal{R}$ the mesh racing factor. The arclength computation flagged with a "∗" experienced difficulties with stability of the mesh equation which accounts for the exceptionally large CPU time.

Description	*CPU time*	*NT*	L^1 *error*	$\mathcal{R}$
Fixed mesh:				
$N = 50$	0.44	78	0.0042	
$N = 100$	0.92	149	0.0028	
$N = 200$	2.43	289	0.0015	
$N = 400$	7.36	562	0.0007	
Moving mesh ($\tau = 0.1$)*:*				
α^{max}, $N = 50$	1.66	185	0.0017	0.104
α^{avg}, $N = 50$	1.58	172	0.0013	0.101
α^{avg}, $N = 100$	4.84	331	0.0005	0.056
Arclength monitor ($N = 50$)*:*				
$\tau = 0.2$	2.67	323	0.0015	0.022
∗ $\tau = 0.1$	31.20	4256	0.0016	0.001

moving mesh calculations with 50 grid points are as accurate as for a fixed mesh with 200 points, and they require less CPU time.

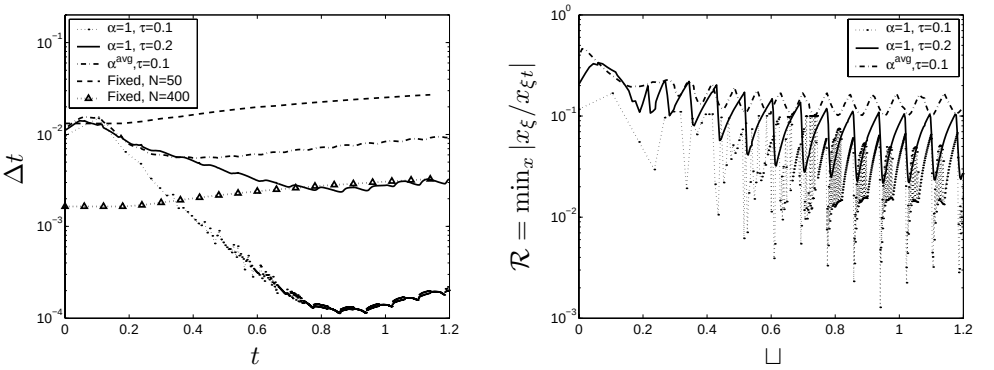
For the computations using regularization parameters α^{avg} and α^{max}, the number of iterations in each time step is at most 2. Therefore, the major factor determining the efficiency of the method is the CFL condition (2.7) arising from the physical PDE. The time steps in the moving mesh calculations are considerably larger than that which would be allowed on a fixed mesh with a similar level of refinement near the shock. This is illustrated by the time step plots in Figure 4.3(a), which correspond to a 400-point fixed mesh and a moving grid with α^{avg}. However, the improvement is not as dramatic as we might have expected from (2.7) because the shifted wave speeds are never exactly zero. It is not possible to constrain mesh points to lie always along the front since they must periodically enter the shock layer from the right and leave from the left, as seen in Figure 4.1. This phenomenon, known as *"mesh racing,"*[4] is unavoidable in r-adaptive methods for which mesh points are not explicitly added or removed throughout the adaptation procedure.

The connection between mesh racing and the CFL number (2.7) can be made more apparent by considering two mesh trajectories, x_{i-1} and x_i, that lie along a shock as pictured in Figure 4.4. The mesh point x_i continues moving along the shock, and so the wave speed is $\lambda \approx \dot{x}_i$. The local CFL number corresponding to the mesh point x_{i-1} is then given by

$$\nu_{i-1/2} = \frac{|\lambda - \dot{x}_{i-1}|\Delta t}{x_i - x_{i-1}} \approx \frac{|\dot{x}_i - \dot{x}_{i-1}|}{\Delta\xi} \cdot \frac{\Delta\xi}{x_i - x_{i-1}} \cdot \Delta t \approx \Delta t \Big/ \left|\frac{x_\xi}{x_{\xi t}}\right|. \tag{4.2}$$

Consequently, when mesh racing occurs, there is an accompanying stability constraint on the time step that is determined by the *"mesh racing factor,"* $\mathcal{R} \doteq \min_x |x_\xi / x_{\xi t}|$. Table 4.1 and Figure 4.3(b) both show that smaller values of $\mathcal{R}$ (associated with clustering of grid points and large mesh curvature) correspond to reductions in the time step. More careful investigation shows that this correlation between time step and mesh racing factor persists for all moving mesh calculations.

[4]While not standardized in the literature, the term "mesh racing" has been used in the past by K. Miller.



(a) Time step. *(b) Mesh racing factor.*

FIG. 4.3. *Plots of the time step (left) and mesh racing factor* $\mathcal{R} = \min_x |x_\xi / x_{\xi t}|$ *(right) for several fixed and moving mesh calculations. The moving mesh computations with arclength and* α^{avg} *monitor functions (with* $N = 50$*) demonstrate that smaller time steps correspond to smaller values of* $\mathcal{R}$*.*

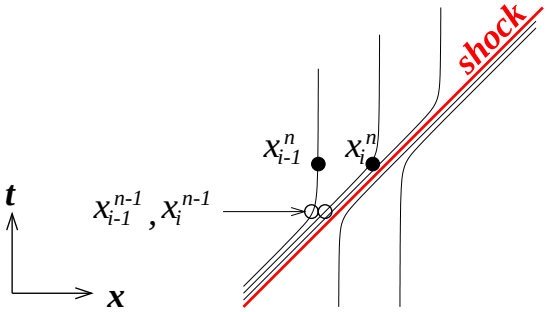


FIG. 4.4. *A plot of moving mesh trajectories in the* $x - t$ *plane, depicted in the neighborhood of a discontinuity, denoted by the thick gray line. Mesh racing is illustrated, in which two mesh trajectories lying along the shock,* x_{i-1} *and* x_i*, diverge from each other between one time level and the next.*

Temporal smoothing plays a central role in the performance of the algorithm and the resolution of discontinuous solution features. The primary influence of the parameter τ is to smooth the mesh trajectories, or in other words, to lessen the mesh "curvature," $x_{\xi t}$. Hence, temporal smoothing is integrally connected to mesh racing. Too much smoothing prevents the moving mesh algorithm from recognizing a discontinuity at all, resulting in a mesh that is nearly uniform. On the other hand, too little smoothing can lead to mesh racing, excessively small time steps, and difficulties with convergence of the solution-mesh iteration. Table 4.2 displays the CPU time and errors for several values of τ, from which it is evident that the algorithm can be quite sensitive to the level of smoothing.

The temporal smoothing parameter can also be interpreted as a time scale over which the grid relaxes towards equidistribution. Consequently, a secondary effect of temporal smoothing is a slight time delay in the mesh motion. This is manifested in Figure 4.1 as a higher concentration of mesh points at the trailing edge of the shock.

TABLE 4.2
Comparison of various fixed and moving grid solutions for Burgers's equation (α^{avg}, $N = 50$). The NT column reports the number of time steps and $\mathcal{R}$ the mesh racing factor. The arclength computation marked with a "∗" experienced difficulty with stability of the mesh equation.

τ	*CPU time*	*NT*	L^1 *error*	$\mathcal{R}$
∗ 0.01	63.38	324	0.0072	0.005
0.025	3.68	261	0.0016	0.050
0.05	1.96	236	0.0022	0.068
0.10	1.58	171	0.0013	0.101
0.20	1.25	117	0.0016	0.200
0.50	0.98	89	0.0035	0.769

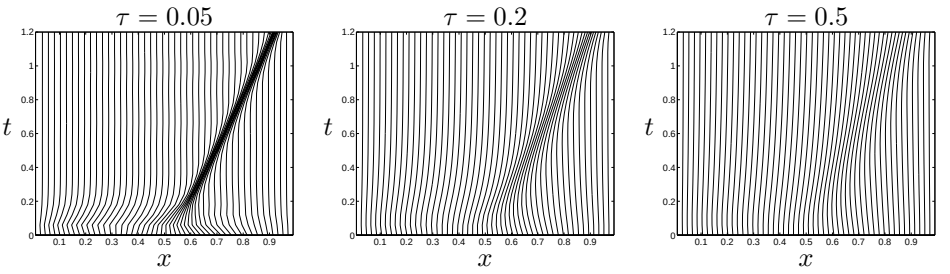


FIG. 4.5. *Mesh trajectories for various temporal smoothing parameters (α^{avg}, $N = 50$).*

A tangible connection can be made between mesh racing and the level of temporal smoothing, based on the analysis of moving mesh methods performed by Smith and Stuart [26] for a general class of time-dependent scalar PDEs. In this work, the authors derive the following bounds on the mesh derivatives in terms of τ and $\bar{M} \doteq \max_x(M)$:

$$|x_{\xi t}| \leq \frac{\bar{M}^2}{\tau} \qquad \text{and} \qquad x_\xi \geq \frac{1}{\bar{M}},$$

which can be combined to obtain a bound on the mesh racing factor,

$$\mathcal{R} \geq \frac{\tau}{\bar{M}^3}. \tag{4.3}$$

While this is only a lower bound on $\mathcal{R}$, it does suggest that our strategy of combining temporal smoothing with a regularized monitor function serves to control mesh racing. This is particularly evident in the case of the α^{max} regularization, for which M is bounded by a constant and, in fact, $\mathcal{R} \geq \tau/\beta^{3/2}$. Referring to the contour plots in Figure 4.5 and the entries in Table 4.2, τ clearly does have a mollifying influence on the mesh curvature and hence also on mesh racing.

We now move on to a discussion of the choice of monitor function. Using solution arclength for problems with steep fronts, the mesh equation is known to become ill-conditioned as the viscosity, and hence also the front thickness, go to zero [15], causing mesh points to cluster in the layer. The results in Table 4.1 demonstrate that the arclength monitor requires a higher value of τ to control the mesh behavior, and the iteration diverges even when τ is as large as 0.1. As long as the temporal smoothing is taken large enough the arclength results are comparable to that for other monitor functions, although the accuracy suffers somewhat because of the time lag in the mesh motion. However, as we will see later in computations with the Euler equations, arclength can introduce severe ill-conditioning in the mesh equation, so we do not advocate its use for hyperbolic problems.

TABLE 4.3

Comparison of moving grid calculations for Burgers's equation based on the method of lines ($N = 50$, $\tau = 0.05$). The columns NT and NJAC give the number of time steps and number of Jacobian evaluations, respectively.

Description	*CPU time*	*NT*	*NJAC*	L^1 *error*
Inviscid Burgers, α^{max}	15.70	1264	221	0.0063
Viscous Burgers, $\alpha \equiv 1$, $\mu = 10^{-3}$	9.08	580	44	0.0041
$\mu = 10^{-4}$	14.70	830	79	0.0039
$\mu = 10^{-5}$	32.90	1523	214	0.0036

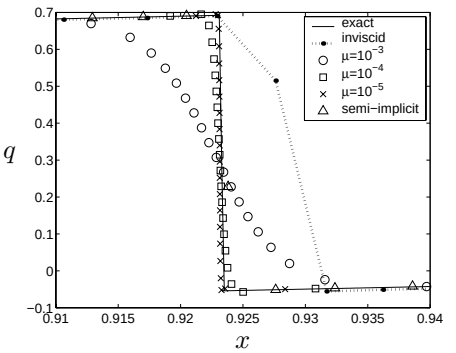


FIG. 4.6. *Close-up of the solution to the viscous and inviscid Burgers equation at time $t = 1.2$ using a fully implicit method of lines approach. Our semi-implicit moving mesh solution is shown for comparison purposes.*

We close with a brief comparison to other more common moving mesh approaches, and report on simulations for which the physical and mesh PDEs are discretized implicitly in time using a method of lines approach. The resulting nonlinear system is solved using the backward difference formula (BDF) solver DASSL [24] with error tolerances $atol = rtol = 10^{-6}$. We employ two different spatial discretizations: the first being a centered finite difference approximation of the viscous Burgers equation with the viscosity $\mu \to 0$, and the second a Godunov scheme for the inviscid problem. The CPU times and errors are listed in Table 4.3, from which it is easy to see the significant increase in cost required for a fully implicit method. The close-up plots of the solution near the shock in Figure 4.6 indicate the clustering together of points as the layer steepens, with CPU times demonstrating the corresponding increase in difficulty of solving the mesh equation.

These difficulties can only be expected to worsen for hyperbolic problems which have no natural dissipative mechanism to control the mesh. In fact, the inviscid Burgers computation fails to converge at all using the arclength monitor function, and so an alternate monitor had to be employed. Using the α^{max} regularization we were able to compute the solution pictured in Figure 4.6. While the L^1 error is only slightly larger in comparison to the other computations, there is clearly a significant error in the shock speed. This points to a serious drawback in using a straightforward method of lines approach for hyperbolic problems: namely, that the resulting discretization may not be conservative. This is one of our main motivations for constructing a semi-implicit algorithm using an explicit, conservative step in the physical solution.

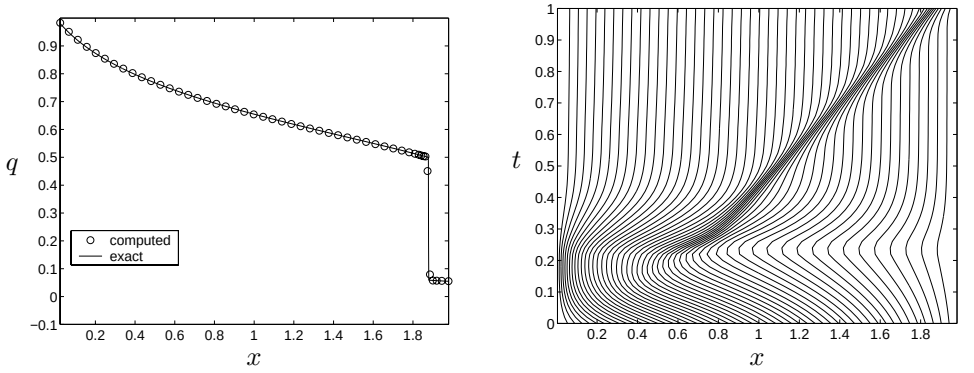


FIG. 4.7. *Moving mesh solution to the Buckley–Leverett equation (α^{max}, $\tau = 0.05$, $N = 50$).*

4.2. Buckley–Leverett equation. We next consider the scalar Buckley–Leverett problem which has been used to describe the one-dimensional flow of two immiscible fluids in a porous medium, neglecting capillary pressure and gravity. This may be used to model gas and oil in a reservoir, for which the equations take the form of a scalar conservation law (1.1a), where q is the fluid saturation (water volume/pore volume) and the flux function is

$$f(q) = \frac{q^2}{q^2 + 0.5(1-q)^2}. \tag{4.4}$$

The solution is obtained on the interval $x \in [0, 1]$, with initial and boundary conditions

$$q(x, 0) = \frac{1}{1 + 10x}, \qquad q(0, t) = 1, \qquad q(1, t) = \frac{1}{11},$$

corresponding to an "oil recovery" scenario where water is pumped into the reservoir at $x = 0$ and oil is forced out at $x = 1$. In contrast with Burgers's equation, the Buckley–Leverett flux function $f(q)$ is nonconvex, which leads to difficulties with some numerical schemes and so serves as an excellent test of a numerical method.

The solution to time $t = 1.0$ sec is shown for a moving mesh in Figure 4.7 and the corresponding fixed mesh simulation in Figure 4.8. Again, the adaptive mesh does an excellent job of capturing and following the discontinuity, and it resolves the shock layer much more accurately than the fixed mesh method with the same number of points.

4.3. Euler equations. The Euler equations describing the evolution of an inviscid, compressible, polytropic gas in one dimension are

$$\frac{\partial}{\partial t}\begin{bmatrix} \rho \\ \rho u \\ e \end{bmatrix} + \frac{\partial}{\partial x}\begin{bmatrix} \rho u \\ \rho u^2 + p \\ u(e+p) \end{bmatrix} = 0, \tag{4.5}$$

where ρ is the density, u the velocity, p the pressure, $e = \frac{p}{\gamma - 1} + \frac{1}{2}\rho u^2$ the internal energy, and γ the ratio of specific heats ($\gamma = 1.4$ for air). The system (4.5) is in the form of a conservation law (1.1a) with $q = (\rho, \rho u, e)^T$ and $f(q) = qu + (0, p, pu)^T$. We

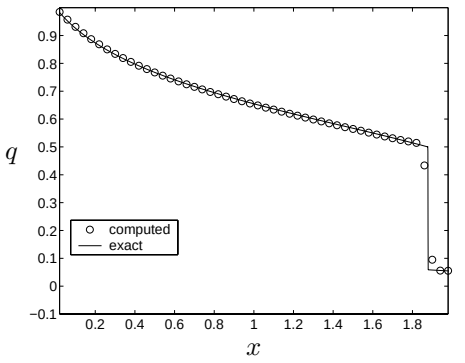


FIG. 4.8. *Fixed mesh solution to the Buckley–Leverett equation ($N = 50$).*

consider Sod's classical shock tube problem [27] which has the initial conditions

$$q(x,0) = \begin{cases} (1.0, 0.0, 2.5) & \text{if } 0 \le x < 0.5, \\ (0.125, 0.0, 0.25) & \text{if } 0.5 \le x \le 1, \end{cases}$$

and reflecting boundary conditions at $x = 0$ and $x = 1$.

In contrast with the Burgers or Buckley–Leverett problems of the previous sections (in which the initial data are smooth), the selection of an appropriate initial mesh is of particular importance here because the initial conditions are discontinuous. In order to allow mesh points to concentrate on or near initial discontinuities, the data must be smoothed over some finite width. We therefore replace jumps in each component of the initial data with smoothed, hyperbolic tangent profiles of the form

$$\widetilde{q}(x) = q_L + \frac{1}{2}(q_R - q_L)\left(1 + \tanh\left(\frac{x - x_c}{\epsilon}\right)\right), \tag{4.6}$$

where q_L and q_R are the left and right jump states, $x = x_c$ is the location of the discontinuity, and ϵ is the smoothing width. Reasonable mesh distributions are obtained when points are exactly equidistributed (i.e., $\tau = 0$) with a smoothing width of $\epsilon \approx 0.005$. The resulting nonlinear system of equations is solved using a fixed point iteration starting from an equally spaced mesh, and in order to improve the convergence of the iteration we use a continuation procedure in which ϵ is gradually decreased from 0.1 to 0.005. The iteration proceeds until the difference between mesh locations for successive iterates goes to zero, and then the resulting mesh and solution data are used as the initial conditions in the numerical scheme.

The performance of the moving mesh algorithm with the arclength monitor function was much worse than for Burgers's equation, requiring an excessively large value of τ for stability. We therefore consider only regularized monitor functions, and begin with $M = \sqrt{1 + \|\boldsymbol{q}_x\|_2^2/\alpha}$, based on a straightforward 2-norm of the solution components, with $\alpha = \int \|\boldsymbol{q}_x\|_2^2 \, dx$. The computed density and mesh contour plot are displayed in Figure 4.9, from which we see that the moving mesh solution provides much better resolution of the shock and rarefaction wave than the fixed mesh algorithm (shown in Figure 4.10). However, the contact discontinuity is not detected at all, and consequently, the resolution of the contact line is no better than for a fixed

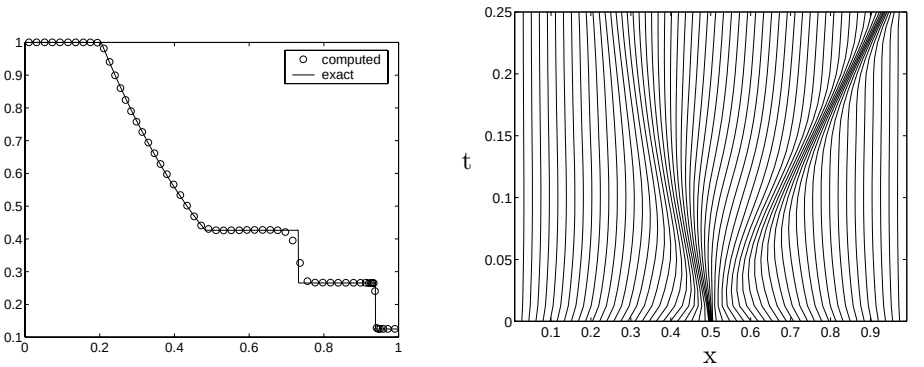


FIG. 4.9. *Moving mesh solution (left: density component; right: mesh contours) for the Euler equations, with monitor function based on the* 2*-norm* (α^{avg}, $\tau = 0.005$, $N = 60$).

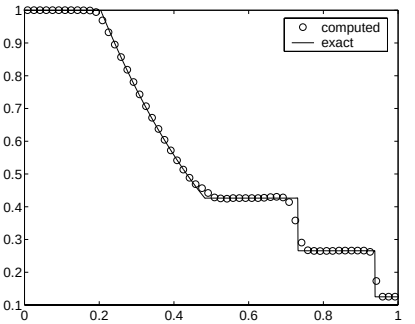


FIG. 4.10. *Fixed mesh solution for the Euler equations with* $N = 60$.

mesh. These qualitative observations are supported by the L^1 error values reported in Table 4.4.

We next demonstrate the effectiveness of the monitor functions tailored specifically to resolving individual elementary waves. The mesh computed with the shock and contact monitor functions, M^s and M^c from (3.4) and (3.5), respectively, are pictured in Figure 4.11. Clearly, both do an excellent job of capturing their corresponding discontinuity, although the errors in Table 4.4 demonstrate the loss in accuracy suffered when only one of the two discontinuous flow features is resolved. Furthermore, the computational time is increased, due primarily to reductions in time step from local CFL violations near the discontinuity that is not being adequately resolved.

The results computed with the monitor function M^{cs} are displayed in Figure 4.12. While the solution is not as sharp at the individual fronts as the solution obtained with each monitor function individually, the averaged monitor function does capture both discontinuities and improves the overall accuracy in the solution relative to the fixed mesh method. The resolution of the contact line is especially good considering the difficulties in resolving this class of wave that are reported for other moving mesh methods (e.g., [12, 22]). It is also important to mention here that we have made no effort to optimize the efficiency of the moving mesh component of the code, and it is certain that considerable gains in CPU time can still be made.

While it is difficult to perform an indepth comparison of accuracy and CPU time

TABLE 4.4
Comparison of various fixed and moving grid solutions for the Euler equations.

Description	*CPU time*	*NT*	L^1 *error*	$\mathcal{R}$
Fixed grid:				
$N = 60$	0.82	56	0.0047	
$N = 120$	2.03	98	0.0026	
$N = 240$	6.60	179	0.0013	
$N = 480$	21.19	339	0.0006	
Moving grid ($N = 60$, $\tau = 0.005$)*:*				
α^{avg} (M based on $\|q_x\|_2^2$)	5.46	152	0.0032	0.0123
Moving grid ($N = 60$, $\tau = 0.005$, α^{max})*:*				
$M = M^c$	5.03	103	0.0041	0.0145
$M = M^s$	4.50	117	0.0043	0.0136
$M = M^{cs}$	3.32	80	0.0026	0.0150

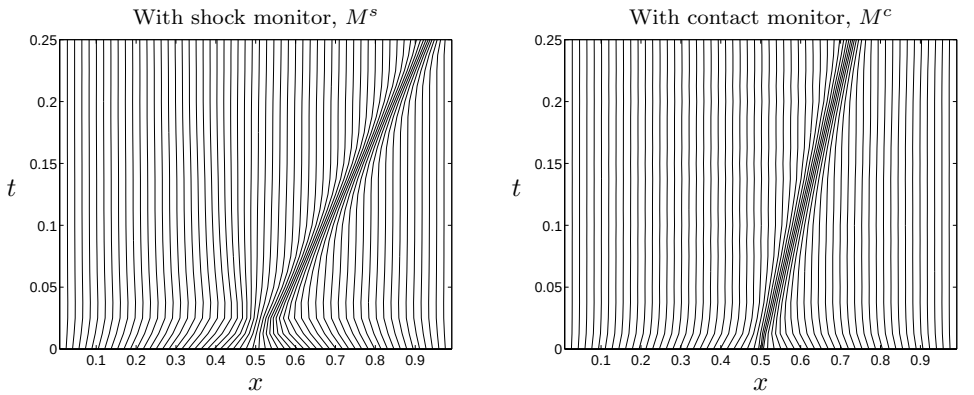


FIG. 4.11. *Mesh contours obtained using the shock and contact monitor functions* ($\tau = 0.005$, $N = 60$).

with other moving mesh results reported in the literature, we can still make some general comments. When compared to other methods (e.g., [14, 2, 22]), even those that employ fully implicit time-stepping, our moving mesh scheme requires at most the same number of time steps for a comparable level of accuracy. If we then take into account the added cost of evaluating and inverting the Jacobian matrix in implicit methods, our scheme is much cheaper. Furthermore, in the case of the Euler equations, we observe a significant improvement in the resolution of contact lines.

5. Conclusions. We have developed an adaptive method for solving one-dimensional hyperbolic conservation laws that is accurate and efficient. This is despite the fact that difficulties identified with solving parabolic problems on moving meshes in the literature (namely, mesh racing and ill-conditioning of the solution-mesh iteration) are exacerbated for hyperbolic problems due in large part to the absence of a natural dissipative mechanism in the physical equations.

The method is based on the existing fast, wave propagation algorithms implemented in the software package CLAWPACK. The accuracy of our results derives from our combining the high resolution, Godunov-type scheme for the physical equations with a moving mesh PDE that uses a monitor function specifically designed to capture solution discontinuities. Numerical results demonstrate the efficacy of the method in comparison with fixed mesh computations and other moving mesh results reported in

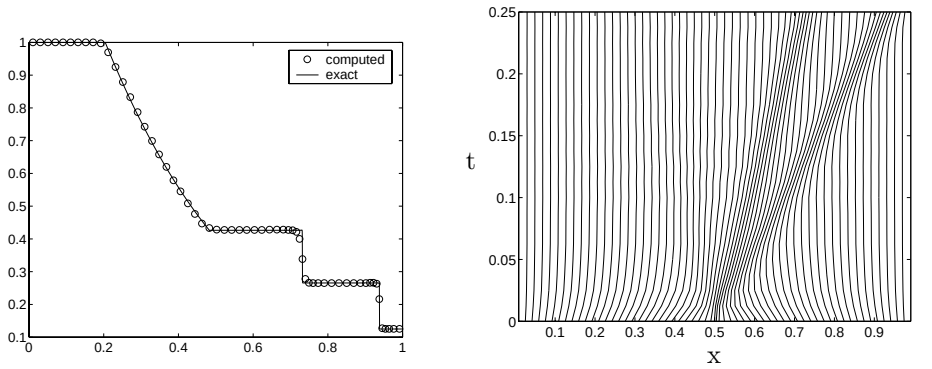


FIG. 4.12. *Moving mesh solution computed with the discontinuity-adaptive monitor function* M^{cs} *(*$\tau = 0.005$*,* $N = 60$*).*

the literature, particularly as regards resolution of contact discontinuities.

The efficiency of our solution algorithm stems primarily from our construction of a semi-implicit iteration for the solution and mesh, which behaves as a two- or three-step predictor-corrector method for the test problems considered here. This should be compared with other moving mesh algorithms, which typically employ a fully implicit time discretization for the coupled solution-mesh system that iterates both solution and mesh to convergence. The rapid convergence of the iteration is aided by a combination of a discontinuity-adaptive monitor function that avoids over-resolving sharp fronts with temporal smoothing that mollifies the effects of mesh racing. The temporal smoothing parameter τ is the only free parameter required in the moving mesh algorithm. However, the level of smoothing required varies between problems, and so this is one area where further work is required. It is likely that performance can be enhanced by allowing τ to depend on the solution throughout a given computation.

We have come to two fundamental conclusions regarding adaptive solution of hyperbolic problems. First, the arclength monitor function, which is the one most commonly used in moving mesh computations, is not appropriate for hyperbolic problems, where high concentrations of mesh points near discontinuous solution features can cause the mesh equations to become ill-conditioned. Second, temporal smoothing is essential for controlling mesh racing, which can otherwise cause serious CFL violations and further degrade the solution-mesh iteration.

We are not advocating r-adaptive methods as a panacea for all hyperbolic problems, particularly because of the proven success of static refinement methods such as adaptive mesh refinement (AMR) [3]. However, we have shown that there is considerable advantage to be gained by some degree of dynamic mesh movement, and so an optimal "overall" solution strategy is likely to derive from a combination of grid movement (r-refinement) and grid subdivision (h-refinement).

The most obvious extension of this work is to higher dimensional problems where multiple shock interactions pose particular challenges for mesh adaptation. The CLAWPACK code has already proven very effective for problems on nonuniform stationary grids in two dimensions, and so our next step will be to generalize the wave propagation scheme to a moving grid. When used in combination with the variational approach for two-dimensional moving meshes developed in Cao, Huang, and Russell [7], our discontinuity-adaptive monitor function should exhibit similar advan-

tages in higher dimensions.

There is also a need for further theoretical work on the behavior of the nonlinear system of equations for the solution and mesh. It is our hope that an analysis for hyperbolic problems (perhaps in the context of the scalar advection equation) will provide insight into the importance of both the monitor function and temporal smoothing, and their relationship to the phenomenon of mesh racing.

REFERENCES

[1] G. Beckett and J. A. Mackenzie, *On a uniformly accurate finite difference approximation of a singularly perturbed reaction-diffusion problem using grid equidistribution*, J. Comput. Appl. Math, to appear.

[2] J. B. Bell and G. R. Shubin, *An adaptive grid finite difference method for conservation laws*, J. Comput. Phys., 52 (1983), pp. 569–591.

[3] M. J. Berger and R. J. LeVeque, *Adaptive mesh refinement using wave-propagation algorithms for hyperbolic systems*, SIAM J. Numer. Anal., 35 (1998), pp. 2298–2316.

[4] R. Biswas, J. E. Flaherty, and D. C. Arney, *An adaptive mesh-moving and refinement procedure for one-dimensional conservation laws*, Appl. Numer. Math., 11 (1993), pp. 259–282.

[5] C. J. Budd, W. Huang, and R. D. Russell, *Moving mesh methods for problems with blow-up*, SIAM J. Sci. Comput., 17 (1996), pp. 305–327.

[6] W. Cao, W. Huang, and R. D. Russell, *An r-adaptive finite element method based upon moving mesh PDEs*, J. Comput. Phys., 149 (1999), pp. 221–244.

[7] W. Cao, W. Huang, and R. D. Russell, *A study of monitor functions for two-dimensional adaptive mesh generation*, SIAM J. Sci. Comput., 20 (1999), pp. 1978–1994.

[8] N. N. Carlson and K. Miller, *Design and application of a gradient-weighted moving finite element code* I*: In one dimension*, SIAM J. Sci. Comput., 19 (1998), pp. 728–765.

[9] J. Chen, *Numerical Study of Blowup Problems and Conservation Laws with Moving Mesh Methods*, Master's thesis, Simon Fraser University, Burnaby, British Columbia, Canada, 1996.

[10] K. Chen, *Error equidistribution and mesh adaptation*, SIAM J. Sci. Comput., 15 (1994), pp. 798–818.

[11] E. A. Dorfi and L. O. Drury, *Simple adaptive grids for* 1*-D initial value problems*, J. Comput. Phys., 69 (1987), pp. 175–195.

[12] K. Farrell and L. O. Drury, *An explicit adaptive grid algorithm for one-dimensional initial value problems*, Appl. Numer. Math., 26 (1998), pp. 3–12.

[13] R. Fazio and R. J. LeVeque, *Moving-mesh methods for one-dimensional hyperbolic problems using Clawpack*, ftp://amath.washington.edu/pub/rjl/papers/rf-rjl:movingmesh.ps.gz, 1998.

[14] R. M. Furzeland, J. G. Verwer, and P. A. Zegeling, *A numerical study of three moving grid methods for one-dimensional partial differential equations which are based on the method of lines*, J. Comput. Phys., 89 (1990), pp. 349–388.

[15] R. J. Gelinas, S. K. Doss, and K. Miller, *The moving finite element method: Applications to general partial differential equations with multiple large gradients*, J. Comput. Phys., 40 (1981), pp. 202–249.

[16] S. K. Godunov, *Difference method of numerical computation of discontinuous solutions of hydrodynamic equations*, Mat. Sb., 47 (1959), pp. 271–306 (in Russian).

[17] A. Harten and J. M. Hyman, *Self-adjusting grid methods for one-dimensional hyperbolic conservation laws*, J. Comput. Phys., 50 (1983), pp. 235–269.

[18] W. Huang, Y. Ren, and R. D. Russell, *Moving mesh methods based on moving mesh partial differential equations*, J. Comput. Phys., 113 (1994), pp. 279–290.

[19] W. Huang and R. D. Russell, *A moving collocation method for solving time dependent partial differential equations*, Appl. Numer. Math., 20 (1996), pp. 101–116.

[20] R. J. LeVeque, *Clawpack–A software package for solving multidimensional conservation laws*, in Hyperbolic Problems: Theory, Numerics, Applications, World Scientific Publishing, River Edge, NJ, 1996, pp. 188–197.

[21] R. J. LeVeque, *Wave propagation algorithms for multi-dimensional hyperbolic systems*, J. Comput. Phys., 131 (1997), pp. 327–353.

[22] S. Li, *Adaptive Methods and Software for Time-Dependent Partial Differential Equations*, Ph.D. thesis, University of Minnesota, 1998.

[23] S. Li, L. Petzold, and Y. Ren, *Stability of moving mesh systems of partial differential equations*, SIAM J. Sci. Comput., 20 (1999), pp. 719–738.

[24] L. R. Petzold, *A description of DASSL: A differential/algebraic system solver*, in Transactions on Scientific Computation, Vol. 1, R. S. Stepleman, ed., IMACS, New Brunswick, NJ, 1982, pp. 65–68.

[25] P. L. Roe, *Approximate Riemann solvers, parameter vectors, and difference schemes*, J. Comput. Phys., 43 (1981), pp. 357–372.

[26] J. H. Smith and A. M. Stuart, *Analysis of Continuous Moving Mesh Equations*, Technical Report SCCM-96-12, Scientific Computing and Computational Mathematics Program, Stanford University, Stanford, CA, 1996.

[27] G. A. Sod, *A survey of finite difference methods for systems of nonlinear hyperbolic conservation laws*, J. Comput. Phys., 27 (1978), pp. 1–31.

[28] P. K. Sweby, *High resolution schemes using flux limiters for hyperbolic conservation laws*, SIAM J. Numer. Anal., 21 (1984), pp. 995–1011.

[29] J. G. Verwer, J. G. Blom, and J. M. Sanz-Serna, *An adaptive moving grid method for one-dimensional systems of partial differential equations*, J. Comput. Phys., 82 (1989), pp. 454–486.

[30] K.-H. A. Winkler, M. L. Norman, and M. J. Newman, *Adaptive mesh techniques for fronts in star formation*, Phys. D, 12 (1984), pp. 408–425.

[31] P. Woodward and P. Colella, *The numerical simulation of two-dimensional fluid flow with strong shocks*, J. Comput. Phys., 54 (1984), pp. 115–173.

SIAM J. SCI. COMPUT.
Vol. 22, No. 5, pp. 1814–1839
© 2001 Society for Industrial and Applied Mathematics

LOCKING AND RESTARTING QUADRATIC EIGENVALUE SOLVERS*

KARL MEERBERGEN†

Abstract. This paper studies the solution of quadratic eigenvalue problems by the quadratic residual iteration method. The focus is on applications arising from finite-element simulations in acoustics. One approach is the shift-and-invert Arnoldi method applied to the linearized problem. When more than one eigenvalue is wanted, it is advisable to use locking or deflation of converged eigenvectors (or Schur vectors). In order to avoid unlimited growth of the subspace dimension, one can restart the method by purging unwanted eigenvectors (or Schur vectors). Both locking and restarting use the partial Schur form. The disadvantage of this approach is that the dimension of the linearized problem is twice that of the quadratic problem. The quadratic residual iteration and Jacobi–Davidson methods directly solve the quadratic problem. Unfortunately, the Schur form is not defined, nor are locking and restarting. This paper shows a link between methods for solving quadratic eigenvalue problems and the linearized problem. It aims to combine the benefits of the quadratic and the linearized approaches by employing a locking and restarting scheme based on the Schur form of the linearized problem in quadratic residual iteration and Jacobi–Davidson. Numerical experiments illustrate quadratic residual iteration and Jacobi–Davidson for computing the linear Schur form. It also makes a comparison with the shift-and-invert Arnoldi method.

Key words. quadratic eigenvalue problem, linearization, Schur factorization, Davidson, shift-and-invert Arnoldi, deflation, purging

AMS subject classifications. 65F15, 65F50

PII. S106482759935174X

1. Introduction.

This paper studies the solution of the quadratic eigenvalue problem

$$Ku + i\omega Cu - \omega^2 Mu = 0, \qquad u \neq 0, \tag{1.1}$$

where K, C, and M have dimension $n \times n$ and M is symmetric positive definite. The scalar ω is called an eigenvalue, u is a corresponding eigenvector, and (ω, u) is an eigenpair. This problem arises from the finite-element simulation of damped acoustic problems, where K is the stiffness matrix and is symmetric positive (semi)definite, M is the mass matrix and is symmetric positive definite, and C is the damping matrix and is symmetric and sometimes complex. The condition number of M is usually relatively small, since M is the discretization of the continuous identity operator. Typically, K, C, and M are large (of the order of 10,000 or more unknowns) and sparse. In engineering applications, the eigenvalue ω is complex. The real part is the resonance frequency, while the imaginary part represents the exponential damping of the eigenmode. In applications, all the eigenvalues in a frequency range are wanted; this is a few tens to a few hundreds of eigenpairs.

The problem (1.1) can be "linearized" into

$$\begin{bmatrix} K & 0 \\ 0 & M \end{bmatrix} \begin{pmatrix} u \\ \omega u \end{pmatrix} = \omega \begin{bmatrix} -iC & M \\ M & 0 \end{bmatrix} \begin{pmatrix} u \\ \omega u \end{pmatrix}, \tag{1.2}$$

*Received by the editors February 11, 1999; accepted for publication (in revised form) July 31, 2000; published electronically January 31, 2001.
http://www.siam.org/journals/sisc/22-5/35174.html

†Free Field Technologies, 16 place de l'Université, 1348 Louvain-la-Neuve, Belgium (karl.meerbergen@fft.be).

which we also denote as $Ax = \omega Bx$ with

$$A = \begin{bmatrix} K & 0 \\ 0 & M \end{bmatrix} \quad \text{and} \quad B = \begin{bmatrix} -iC & M \\ M & 0 \end{bmatrix}. \tag{1.3}$$

(See [21, Chapter X] and [31] for alternatives.) Note that for acoustic finite-element applications, A is symmetric positive (semi)definite and B is (in general) complex symmetric. Since M is nonsingular, this problem has $2n$ finite eigenvalues. When C is purely imaginary, $Ax = \omega Bx$ is a Hermitian problem, so all eigenvalues are real. In general, C has a real component, and hence complex eigenvalues are present. When C is real, then the spectrum is symmetric with respect to the imaginary axis: indeed, if (ω, u) satisfies (1.1) and C is real, then $(-\bar{\omega}, \bar{u})$ is also an eigenpair. Note that when $C = 0$, the spectral transformation block Lanczos method [7] is a very robust and efficient solver.

In the literature, methods have been proposed for solving (1.1) and (1.2). The linearized problem can be solved by the shift-and-invert Arnoldi method [21, 18]. This method computes the eigenpairs of the shift-and-invert transformation $(A - \sigma B)^{-1}B$, where σ is called the shift. Alternatively, the rational Krylov method [20] or the Jacobi–Davidson method [27] may be used. The advantage of the linearized approach is that existing methods and software can be used. A disadvantage is that the dimension is doubled.

Methods have been developed for directly tackling (1.1). They solve a sequence of linear systems

$$(K + i\sigma C - \sigma^2 M)y = r, \tag{1.4}$$

where σ may change at each iteration. When a direct method is used for solving (1.4), a matrix factorization on each iteration is inevitable. Neumaier [19] and Huitfeldt and Ruhe [9] propose methods that use a fixed σ. This allows the same factorization to be used for several iterations. Another approach is the Jacobi–Davidson method [26, 28, 32] for the quadratic problem. The methods that we study build a subspace. For reasons of computational cost and memory, the subspace dimension is limited. When this limit has been reached without convergence of the sought-after eigenvalues, the method needs to be restarted. The concept of restarting eigenvalue solvers is very well understood for linear problems. See the recent work for the Arnoldi method [29, 11, 16], the Jacobi–Davidson method [5], and the rational Krylov method [20, 3]. When more than one eigenvalue is wanted, it is usually advisable to lock the converged eigenpairs. This is proposed for subspace iteration [30], Arnoldi's method [11], Jacobi–Davidson [5], and rational Krylov [20]. Both restarting and locking use the partial Schur form.

The purpose of this paper is the development of reliable deflation and restarting in methods that solve the quadratic problem (1.1). The problem is that the Schur form is not defined for quadratic problems. We give a definition and show that this Schur form does not always exist. Therefore, we propose using the Schur form of the linearized problem (1.2).

We also want to stress that all theory in this paper can be extended to the m degree polynomial case with $m > 2$. The polynomial problem

$$A_0 u + \lambda A_1 u + \cdots + \lambda^m A_m u = 0$$

can be solved by linearization into

$$\begin{bmatrix} A_0 & & & \\ & I & & \\ & & \ddots & \\ & & & I \end{bmatrix} \begin{pmatrix} u \\ \lambda u \\ \lambda^2 u \\ \vdots \\ \lambda^{m-1} u \end{pmatrix} = \lambda \begin{bmatrix} -A_1 & -A_2 & \cdots & -A_m \\ I & & & \\ & \ddots & & \\ & & I & \end{bmatrix} \begin{pmatrix} u \\ \lambda u \\ \lambda^2 u \\ \vdots \\ \lambda^{m-1} u \end{pmatrix}.$$

The paper is organized as follows. In section 2, we present the quadratic residual iteration method for solving (1.1) and prove a relationship with a modification of the generalized Davidson method for the solution of the linearized problem (1.2). In section 3, we use the theory developed in section 2 to link the Jacobi–Davidson methods for (1.1) and (1.2). In section 4, the notion of partial Schur form is extended to quadratic eigenvalue problems, and the theory of section 2 is used to efficiently compute an approximate partial Schur form of the linearized problem by orthogonal projection of the quadratic problem. In section 5, a deflation or locking technique is proposed for the linearized problem, and in section 6, we discuss restarting the linearized problem by purging Schur vectors. In section 7, we explain which vectors we use to expand the subspace when we want to compute a partial Schur form of the linearized problem. Section 8 presents a practical algorithm that is illustrated by numerical examples including one from applications. In section 9, we compare quadratic residual iteration and Jacobi–Davidson for computing a partial Schur form. In section 10, we compare the shift-and-invert Arnoldi method with quadratic residual iteration and Jacobi–Davidson. Finally, we summarize the main conclusions in section 11. Throughout the paper, $\|\cdot\|$ is used for the 2-norm of matrices and vectors and $\|\cdot\|_F$ for the Frobenius norm.

2. Quadratic residual iteration. In this section, we derive a relationship between methods for solving (1.1) and (1.2). All conclusions assume exact arithmetic. For results on the backward error and condition of the linearized problem we refer to Tisseur [31]. For the linearized problem, we consider the generalized Davidson method [15] (which formally covers the Jacobi–Davidson method [27, 17]) and for the quadratic problem we consider the residual iteration method [19] with subspace projection. We also discuss the Jacobi–Davidson variant for this problem. This section is devoted to the development of a relationship between the two approaches. Therefore, we also define a modified Davidson technique for the linearized problem (1.2) which is shown to produce the same results as the quadratic residual iteration on (1.1).

The generalized Davidson method for the linearized problem is described by the following algorithm [15]. The ω_k's form the sequence of Ritz values, and the σ_k's are chosen shifts.

ALGORITHM 2.1 (generalized Davidson method).

1. *Given* $v_1 \in \mathbf{C}^{2n}$ *with* $\|v_1\| = 1$.
2. *For* $k = 1, 2, \ldots$ *do*
   - 2.1. *Let* $\mathcal{V}_k = [v_1, \ldots, v_k]$.
   - 2.2. *Compute the projection matrices* $A_k = \mathcal{V}_k^* A \mathcal{V}_k$, *and* $B_k = \mathcal{V}_k^* B \mathcal{V}_k$.
   - 2.3. *Compute the eigenpair* (ω_k, z_k) *of interest of* $A_k z = \omega B_k z$.
   - 2.4. *Compute the corresponding Ritz vector* $x_k = \mathcal{V}_k z_k$.
   - 2.5. *Compute the corresponding residual* $r_k = Ax_k - \omega_k B x_k$.
   - 2.6. *Solve the linear system* $(A - \sigma_k B) y_k = r_k$.
   - 2.7. *Orthonormalize* y_k *against* $v_1, \ldots, v_k$ *into* v_{k+1}.

This algorithm consists of a sequence of Cayley transformations

$$y_k = (A - \sigma_k B)^{-1}(A - \omega_k B)x_k \tag{2.1}$$

and a projection step for computing the approximate eigenpair. The Cayley transform aims to improve the approximation of eigenvalues near σ_k. For projection methods, approximate eigenpairs are called Ritz pairs. The approximate eigenvalue is a Ritz value, and the approximate eigenvector a Ritz vector. The small eigenvalue problem in step 2.3 is usually solved by the QZ method [6]. We select the eigenpair of interest, e.g., corresponding to the eigenvalue nearest σ_k. When the linear system in step 2.6 is solved by an iterative method, Algorithm 2.1 is the generalized Davidson method. In Davidson's method, one usually employs $\sigma_k = \omega_k$ and step 2.6 is executed approximately [1], or one can use the Jacobi–Davidson method (see section 3). This choice of σ_k leads to quadratic convergence. The generalized Davidson method does not exploit the special structure of eigenvectors of (1.2). Clearly, it is sufficient to compute the first n components of the eigenvectors and then construct the remaining components. The following algorithm is a modification of the generalized Davidson method that uses the first n components only.

ALGORITHM 2.2 (modified Davidson method).

1. *Given* $v_1 \in \mathbf{C}^n$ *with* $\|v_1\| = 1$.
2. *For* $k = 1, 2, \ldots$ *do*
   - 2.1. *Let* $V_k = [v_1, \ldots, v_k]$ *and* $\mathcal{V}_{2k} = \begin{pmatrix} V_k & 0 \\ 0 & V_k \end{pmatrix}$.
   - 2.2. *Compute the projection matrices* $A_{2k} = \mathcal{V}_{2k}^* A \mathcal{V}_{2k}$, *and* $B_{2k} = \mathcal{V}_{2k}^* B \mathcal{V}_{2k}$.
   - 2.3. *Compute the eigenpair* (ω_k, z_k) *of interest of* $A_{2k}z = \omega B_{2k}z$.
   - 2.4. *Compute the corresponding Ritz vector* $x_k = \mathcal{V}_{2k} z_k$.
   - 2.5. *Compute the corresponding residual* $r_k = Ax_k - \omega_k B x_k$.
   - 2.6. *Solve the linear system* $(A - \sigma_k B)y_k = r_k$.
   - 2.7. *Orthonormalize the first* n *components of* y_k *against* $v_1, \ldots, v_k$ *into* v_{k+1}.

Alternatively, one could select the last n components of y_k in step 2.7. As we shall see, there is no advantage in exact arithmetic. The basis vectors satisfy the projection equation

$$A\mathcal{V}_{2k} - B\mathcal{V}_{2k}H_{2k} = \mathcal{F}_{2k}$$

with projection matrix $H_{2k} = B_{2k}^{-1} A_{2k}$ and with residual term $\mathcal{F}_{2k}$ satisfying $\mathcal{V}_{2k}^* \mathcal{F}_{2k} = 0$. The projection equation is frequently used in section 4.

The quadratic residual iteration is now discussed.

ALGORITHM 2.3 (quadratic residual iteration).

1. *Given* $v_1 \in \mathbf{C}^n$ *with* $\|v_1\| = 1$.
2. *For* $k = 1, 2, \ldots$ *do*
   - 2.1. *Let* $V_k = [v_1, \ldots, v_k]$.
   - 2.2. *Compute the projection matrices* $K_k = V_k^* K V_k$, $C_k = V_k^* C V_k$, *and* $M_k = V_k^* M V_k$.
   - 2.3. *Compute the eigenpair* (ω_k, z_k) *of interest of* $K_k z + i\omega C_k z - \omega^2 M_k z = 0$.
   - 2.4. *Compute the corresponding Ritz vector* $u_k = V_k z_k$.
   - 2.5. *Compute the corresponding residual* $r_k = Ku_k + i\omega_k C u_k - \omega_k^2 M u_k$.
   - 2.6. *Solve the linear system* $(K + i\sigma_k C - \sigma_k^2 M)y_k = r_k$.
   - 2.7. *Orthonormalize* y_k *against* $v_1, \ldots, v_k$ *into* v_{k+1}.

The algorithm is very similar to Algorithms 2.1 and 2.2, but instead of a regular Cayley transform, a quadratic Cayley transform

$$y_k = (K + i\sigma_k C - \sigma_k^2 M)^{-1}(K + i\omega_k C - \omega_k^2 M)u_k \tag{2.2}$$

is employed. The small eigenvalue problem at step 2.3 is solved by a method for solving quadratic eigenvalue problems, e.g., Newton's method, or by solving the corresponding linearized problem.

In the following, we derive a relationship between Algorithms 2.2 and 2.3. The main operations in these algorithms are the Cayley transform and the projection step.

Cayley transformation. On each iteration, the subspace is expanded by the Cayley transformation applied to a Ritz vector. Here we establish a relationship between the Cayley transforms (2.1) and (2.2) applied to a vector with a special structure.

LEMMA 2.1. *Let A and B be defined by* (1.3) *and*

$$\begin{pmatrix} x \\ y \end{pmatrix} = (A - \sigma B)^{-1}(A - \omega B) \begin{pmatrix} u \\ \omega u \end{pmatrix}. \tag{2.3}$$

Then

$$x = (K + i\sigma C - \sigma^2 M)^{-1}(K + i\omega C - \omega^2 M)u \qquad \text{and} \qquad y = \sigma x. \tag{2.4}$$

Proof. Write (2.3) as

$$\begin{aligned} \begin{bmatrix} K + i\sigma C & -\sigma M \\ -\sigma M & M \end{bmatrix} \begin{pmatrix} x \\ y \end{pmatrix} &= \begin{bmatrix} K + i\omega C & -\omega M \\ -\omega M & M \end{bmatrix} \begin{pmatrix} u \\ \omega u \end{pmatrix} \\ &= \begin{pmatrix} (K + i\omega C - \omega^2 M)u \\ 0 \end{pmatrix}. \end{aligned}$$

The last row readily produces $y = \sigma x$. Substituting this in the first row leads to the first statement in (2.4). This proves the lemma. ∎

Projection. The projection used in Algorithm 2.2 produces the same Ritz pairs as the projection in Algorithm 2.3 if the V_k's are the same.

LEMMA 2.2. *If the projection of* (1.1) *on* Range(V) *produces the Ritz pair* (ω, u), *then the projection of* (1.2) *on*

$$\mathcal{V} = \text{Range}\left(\begin{pmatrix} V & 0 \\ 0 & V \end{pmatrix}\right)$$

produces the Ritz pair

$$\left(\omega, \begin{pmatrix} u \\ \omega u \end{pmatrix}\right). \tag{2.5}$$

Proof. The projection of (1.2) on $\mathcal{V}$ is

$$\begin{aligned} &\begin{pmatrix} V & 0 \\ 0 & V \end{pmatrix}^* \begin{bmatrix} K & 0 \\ 0 & M \end{bmatrix} \begin{pmatrix} V & 0 \\ 0 & V \end{pmatrix} \begin{pmatrix} z \\ t \end{pmatrix} \\ &= \omega \begin{pmatrix} V & 0 \\ 0 & V \end{pmatrix}^* \begin{bmatrix} -iC & M \\ M & 0 \end{bmatrix} \begin{pmatrix} V & 0 \\ 0 & V \end{pmatrix} \begin{pmatrix} z \\ t \end{pmatrix}. \end{aligned}$$

This leads to $V^*KVz = \omega(-iV^*CVz + V^*MVt)$ and $t = \omega z$. This implies that $V^*(K + i\omega C - \omega^2 M)Vz = 0$, which is the projection of (1.1) on Range(V). This completes the proof. ∎

The theory can now be used for establishing the following relationship.

THEOREM 2.3. *Suppose that v_1 is the same for Algorithms* 2.2 *and* 2.3. *When $k \leq n$, both algorithms produce the same Ritz pairs.*

Proof. The proof is given by induction. Suppose that we use an initial vector v_1 for Algorithms 2.2 and 2.3. Suppose that the theorem is true at the end of iteration $k-1$, i.e., V_k is the same for both algorithms. Following Lemma 2.2, Algorithm 2.3 produces (ω, u), and Algorithm 2.2 produces $(\omega, \begin{pmatrix} u \\ \omega u \end{pmatrix})$. Finally, following Lemma 2.1, x_k is the same for both algorithms, so both produce the same v_{k+1}. This implies that V_{k+1} is equal in both algorithms, from which the proof follows. ∎

An important question is how much better is the subspace generated by Algorithm 2.2 than Algorithm 2.1. Let $\begin{pmatrix} u \\ \omega u \end{pmatrix}$ be a Ritz vector, and let $\begin{pmatrix} y \\ \sigma y \end{pmatrix}$ be the result of the Cayley transformation. In Algorithm 2.2, we add both $\begin{pmatrix} y \\ 0 \end{pmatrix}$ and $\begin{pmatrix} 0 \\ y \end{pmatrix}$ to the subspace. When $\sigma \neq 0$, this is mathematically equivalent to adding $\begin{pmatrix} y \\ \sigma y \end{pmatrix}$ and $\begin{pmatrix} y \\ -\sigma y \end{pmatrix}$, since both vector pairs have the same span. The first vector is the result of the Cayley transform

$$\begin{pmatrix} y \\ \sigma y \end{pmatrix} = (A - \sigma B)^{-1}(A - \omega B) \begin{pmatrix} u \\ \omega u \end{pmatrix}.$$

The first vector tries to improve the eigenvector components corresponding to the eigenvalues near σ. When $C = 0$, then the second vector is

$$\begin{pmatrix} y \\ -\sigma y \end{pmatrix} = (A + \sigma B)^{-1}(A + \omega B) \begin{pmatrix} u \\ -\omega u \end{pmatrix},$$

i.e., tries to improve the eigenvectors corresponding to eigenvalues nearest $-\sigma$. In general, however, $\begin{pmatrix} y \\ -\sigma y \end{pmatrix}$ does not have a particular meaning. When $\|C\|$ is small, this vector may help finding eigenvalues near $-\sigma$.

Stopping criterion. Usually, iterative eigenvalue solvers use a stopping criterion based on the residual. If $Ax - \lambda Bx = r$, then (λ, x) is an exact eigenpair of the perturbed problem $(A + E)x = \lambda Bx$ with $Ex = -r$. So $\|r\|/\|x\|$ is a measure of the backward error. A relationship between the residual and the forward error on the eigenvalues is given by the Bauer–Fike theorem [21, Theorem 3.6].

THEOREM 2.4 (Bauer–Fike). *Consider an $n \times n$ matrix G that has n independent eigenvectors. Let λ be an eigenvalue of G, and let (μ, x) be an approximate eigenpair with residual $r = Gx - \mu x$ and $\|x\| \neq 0$. Then*

$$|\lambda - \mu| \leq \kappa \|r\|/\|x\|,$$

where κ is the condition number of the matrix of eigenvectors.

The residual for the linearized problem with Ritz pair (2.5) becomes

$$r_L = A \begin{pmatrix} u \\ \omega u \end{pmatrix} - \omega B \begin{pmatrix} u \\ \omega u \end{pmatrix} = \begin{pmatrix} r_P \\ 0 \end{pmatrix}$$

with $r_P = Ku + i\omega Cu - \omega^2 Mu$. The first component of r_L is the quadratic residual, so the 2-norm of both residuals is equal. So, for quadratic problems, the residual can also be regarded as a backward error. Note, however, that the Ritz vectors, u and $\begin{pmatrix} u \\ \omega u \end{pmatrix}$, have a different norm. In order to use the Bauer–Fike theorem, consider the residual $B^{-1}Ax - \omega x$, which becomes

$$\begin{aligned} B^{-1}A \begin{pmatrix} u \\ \omega u \end{pmatrix} - \omega \begin{pmatrix} u \\ \omega u \end{pmatrix} &= \begin{bmatrix} -iC & M \\ M & 0 \end{bmatrix}^{-1} \begin{pmatrix} Ku + i\omega Cu - \omega^2 Mu \\ 0 \end{pmatrix} \\ &= \begin{pmatrix} 0 \\ M^{-1}Ku + i\omega M^{-1}Cu - \omega^2 u \end{pmatrix}. \end{aligned}$$

So the error between the exact eigenvalue λ and an approximation ω can be bounded by

$$|\omega - \lambda| \leq \kappa \|M^{-1}\| \frac{\|Ku + i\omega Cu - \omega^2 Mu\|}{\|u\|\sqrt{1+|\omega|^2}},$$

where κ is the condition number of the matrix of eigenvectors of $B^{-1}A$, and the denominator is the norm of the Ritz vector.

Accuracy of the Cayley transform. A comment is in order on the solution of the linear system

$$(K + i\sigma C - \sigma^2 M)y = (K + i\omega C - \omega^2 M)u \equiv r_P.$$

When we use a linear system solver, we have a residual s so that

$$(K + i\sigma C - \sigma^2 M)y = (K + i\omega C - \omega^2 M)x - s.$$

When a direct method is used, $\|s\|$ is usually of the order of machine precision times $\|r_P\|$ and the quadratic Cayley transform can be considered as exact. When an iterative solver is used, $\|s_k\| \leq \tau \|r_P\|$ with τ as the relative residual tolerance. The smaller τ is, the more expensive the iterative solver is. We therefore want to choose the tolerance not too small. There is a relationship with the linear problem, since

$$\begin{bmatrix} K + i\sigma C & -\sigma M \\ -\sigma M & M \end{bmatrix} \begin{pmatrix} y \\ \sigma y \end{pmatrix} = \begin{bmatrix} K + i\omega C & -\omega M \\ -\omega M & M \end{bmatrix} \begin{pmatrix} u \\ \omega u \end{pmatrix} - \begin{pmatrix} s \\ 0 \end{pmatrix}.$$

When iterative solvers are used for computing the linear Cayley transform in eigenvalue solvers, we talk about the inexact Cayley transform [14, 13, 10, 17]. Two elements play a role in the convergence of an inexact Cayley transform iteration method. The first one is the convergence for an exact Cayley transformation, and the second one is the accuracy of the linear system solver, τ. When τ is not too large, the convergence is as fast as for $\tau = 0$. We give an example in section 8.3.

Cayley transform and shift-and-invert. A problem arises when $\sigma = \omega$ in the Cayley transformation. Since $y = (A - \omega B)^{-1}(A - \omega B)x = x$, no new direction is added to the subspace. When the Cayley transform is computed exactly, e.g., with a direct linear solver (in practice, we have a small backward error on the solution of the linear system), then

$$y \equiv (A - \sigma B)^{-1}(A - \omega B)x = x + (\sigma - \omega)(A - \sigma B)^{-1}Bx. \tag{2.6}$$

The matrix $(A - \sigma B)^{-1}B$ is called the shift-and-invert transformation [21] or the spectral transformation [4]. The use of the Cayley transform or the spectral transformation in a projection method is equivalent since both add the same direction orthogonal to x to the subspace. The advantage of the shift-and-invert transform is that the transformation still works when $\sigma = \omega$. When an iterative solver is used, it is usually advantageous to use the Cayley transform instead of shift-and-invert [10, 17].

For the quadratic eigenvalue problem, the shift-and-invert transformation is defined as follows. The Cayley transform can be rewritten as

$$\begin{aligned} y &\equiv (K + i\sigma C - \sigma^2 M)^{-1}(K + i\omega C - \omega^2 M)u \\ &= u + (\omega - \sigma)(K + i\sigma C - \sigma^2 M)^{-1}(iC - (\omega + \sigma)M)u. \end{aligned}$$

The shift-and-invert transformation is defined as

$$(K + i\sigma C - \sigma^2 M)^{-1}(iC - (\omega + \sigma)M) \tag{2.7}$$

and can also be used when $\sigma = \omega$.

3. The Jacobi–Davidson method. The Jacobi–Davidson method is an alternative to the shift-and-invert transformation when an iterative linear solver is used and $\sigma = \omega$.

The combination of the Newton method applied to the set of nonlinear equations

$$\begin{aligned} Ax - \omega Bx &= 0, \\ x^*x &= 1 \end{aligned}$$

in ω and x and the generalized Davidson method (Algorithm 2.1) is called the Jacobi–Davidson method [27, 26]. It adds the x component of the solution of the Newton iteration to a subspace and computes Ritz pairs by projection. The algorithm for the linear problem (1.2) is the same as Algorithm 2.1 except for the transformation in step 2.6. Instead, a "correction equation" is solved. Let (ω, x) be a Ritz pair; then the subspace is expanded with y satisfying

$$\left(I - \frac{Bxx^*}{x^*Bx}\right)(A - \omega B)\left(I - \frac{xx^*}{x^*x}\right)y = (A - \omega B)x. \tag{3.1}$$

The solution y is computed by an iterative method with a suitable preconditioner if available. For the quadratic problem (1.1), the Jacobi–Davidson method is Algorithm 2.3 where step 2.6 is replaced by

$$\begin{aligned} &\left(I - \frac{(-iC + 2\omega M)uu^*}{u^*(-iC + 2\omega M)u}\right)(K + i\omega C - \omega^2 M)\left(I - \frac{uu^*}{u^*u}\right)v \\ &\qquad = (K + i\omega C - \omega^2 M)u, \end{aligned} \tag{3.2}$$

where (ω, u) is a Ritz pair [26, 32]. This can be derived from one Newton iteration on (1.1) and $u^*u = 1$.

The Jacobi–Davidson method is strongly related to the Davidson method. When we require that $y \perp x$, we can rewrite (3.1) as

$$\begin{aligned} (A - \omega B)y - \epsilon Bx &= (A - \omega B)x \\ \text{or} \quad y &= x + \epsilon(A - \omega B)^{-1}Bx, \end{aligned} \tag{3.3}$$

where ϵ is so that $y \perp x$ [27]. Note the resemblance with (2.6). When y is added to the subspace, the new direction v_{k+1} is independent of ϵ, since x is in the subspace. We can write y also as

$$y = (A - \omega B)^{-1}(A - (\omega - \epsilon)B)x.$$

We immediately see the connection with the generalized Davidson method, where the pole (σ) is now ω and the zero (ω) is now $\omega + \epsilon$. It is shown in [17] that $\omega + \epsilon$ is the harmonic Ritz value with target ω. The value ϵ converges to zero when ω converges to an eigenvalue [27].

Similarly, with $v \perp u$, (3.2) can be rewritten as

$$\begin{aligned} (K + i\omega C - \omega^2 M)v - \eta(-iC + 2\omega M)u &= (K + i\omega C - \omega^2 M)u, \\ v &= u + \eta(K + i\omega C - \omega^2 M)^{-1}(-iC + 2\omega M)u, \end{aligned} \tag{3.4}$$

where η is so that $v \perp u$. Note the resemblence with (2.7) for $\sigma = \omega$.

In section 2, we showed the equivalence of the modified Davidson method in Algorithm 2.2 and quadratic residual iteration in Algorithm 2.3. The following lemma

shows that, when step 2.6 is replaced by the solution of the Newton correction equations (3.1) and (3.2), respectively, the subspace V is expanded in the same way for both Algorithms 2.2 and 2.3.

LEMMA 3.1. *Let (ω, u) be a Ritz pair for* (1.1), *and assume $\epsilon \neq 0$. Let $x = \begin{pmatrix} u \\ \omega u \end{pmatrix}$, and let $y = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}$ be the solution of* (3.1). *Then $(\eta/\epsilon)y_1$ is a solution of* (3.2).

Proof. Assume that $y \perp x$ and rewrite (3.3) as

$$\begin{bmatrix} K + i\omega C & -\omega M \\ -\omega M & M \end{bmatrix} \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} - \epsilon \begin{bmatrix} -iC & M \\ M & 0 \end{bmatrix} \begin{pmatrix} u \\ \omega u \end{pmatrix} = \begin{bmatrix} K + i\omega C & -\omega M \\ -\omega M & M \end{bmatrix} \begin{pmatrix} u \\ \omega u \end{pmatrix}$$

or

$$\begin{aligned} (K + i\omega C)y_1 - \omega M y_2 - \epsilon(-iC + \omega M)u &= Ku + i\omega Cu - \omega^2 Mu, \\ -\omega M y_1 + M y_2 - \epsilon M u &= 0, \end{aligned}$$

from which $y_2 = \omega y_1 + \epsilon u$. This gives

$$(K + i\omega C - \omega^2 M)y_1 - \epsilon(-iC + 2\omega M)u = Ku + i\omega Cu - \omega^2 Mu,$$

which looks like (3.4). The only difference is that $\eta \neq \epsilon$. Since $(\eta/\epsilon)y_1$ and v in (3.4) have the same component orthogonal to u, $(\eta/\epsilon)y_1$ satisfies (3.2). This gives the proof of the lemma. ∎

Finally, it is important to note that the projections used in (3.1) and (3.2) can also be used for $\sigma \neq \omega$. An example is given in [12, section 3.3.6]. However, when the linear systems are solved exactly (e.g., using a direct linear solver), there is no advantage in doing this. The search space V_k will be expanded with the same direction. This is discussed in more detail in [10, (6.3) and (6.4)] and [17]. When an iterative solver is used, the projections may help speed up the convergence of the iterative solvers, since the projection may remove a small eigenvalue of the matrix in the linear system. An example is given in section 9.

4. The Schur factorization. Methods for solving linear eigenvalue problems use the Schur factorization when more than one eigenvalue needs to be computed. The reasons are that a set of $2n$ independent Schur vectors always exists in contrast with the eigenvectors, Schur bases are orthogonal, and they can easily be used for locking and restarting purposes as we will discuss in the coming sections. This section is devoted to the Schur factorization for the linear problem and the quadratic problem. We show some properties of the Schur factorization of the linear problem and define one for the quadratic problem. We also show that the quadratic Schur form may not exist. Therefore, we suggest the use of the Schur form of the linearized problem.

4.1. Linear problems. When B is invertible, the Schur form or Schur factorization of $Ax = \lambda Bx$ is defined by

$$B^{-1}AX = XS \qquad \Leftrightarrow \qquad AX = BXS, \tag{4.1}$$

where X is an $n \times n$ unitary matrix and S is an $n \times n$ upper triangular matrix with the eigenvalues on the main diagonal. The columns of X are the Schur vectors, and S is the Schur matrix. The Schur form (4.1) can be computed by the QR method

applied to $B^{-1}A$ or the QZ method applied to the pair (A, B) [6]. Using the QR method assumes a nonsingular B and the formation of $B^{-1}A$. One could avoid this computation by using the generalized Schur form computed by the QZ method. This complicates the notation, but the concept is very similar. In this paper, we use the Schur form from (4.1) computed by the QR method.

A partial Schur form of order p with $1 \leq p \leq n$ is defined by

$$AX_p = BX_pS_p,$$

where $X_p \in \mathbf{C}^{n\times p}$ has orthonormal columns and $S_p \in \mathbf{C}^{p\times p}$ is upper triangular with eigenvalues of $Ax = \lambda Bx$ on the main diagonal.

When A and B are large and sparse, the QR method is not suitable for computing a partial Schur form. One usually employs a projection method, i.e., approximate Schur vectors $\mathcal{U}_k$ are computed in the subspace spanned by the (orthonormal) columns of the matrix $\mathcal{V}_k$ so that

$$A\mathcal{U}_k - B\mathcal{U}_kS_k = \mathcal{F}_k, \tag{4.2}$$

where $\mathcal{U}_k = \mathcal{V}_kX_k$ and $\mathcal{F}_k$ is a residual term. Using the concept of orthogonal projection, i.e., we force $\mathcal{V}_k^*\mathcal{F}_k = 0$, we find that X_k and S_k satisfy the $k \times k$ Schur problem

$$A_kX_k - B_kX_kS_k = 0 \quad \text{with} \quad A_k = \mathcal{V}_k^*A\mathcal{V}_k \ \text{ and } \ B_k = \mathcal{V}_k^*B\mathcal{V}_k, \tag{4.3}$$

which can easily be solved by the QR method on $H_k = B_k^{-1}A_k$. We call (4.2) the *approximate partial Schur form.*

We can describe an orthogonal projection method in terms of $\mathcal{U}_k$ instead of $\mathcal{V}_k$ since both form an orthonormal basis for the same subspace. The use of the Schur basis instead of $\mathcal{V}_k$ makes it easier to lock and restart as we shall see later. After the kth iteration we thus have (4.2) with $\mathcal{U}_k^*\mathcal{F}_k = 0$. In the $k+1$st iteration, a new vector v_{k+1} is added and a new projection matrix H_{k+1} is computed so that

$$A\begin{pmatrix} \mathcal{U}_k & v_{k+1} \end{pmatrix} - B\begin{pmatrix} \mathcal{U}_k & v_{k+1} \end{pmatrix} H_{k+1} = \mathcal{F}_{k+1}$$

with $(\mathcal{U}_k \quad v_{k+1})^*\mathcal{F}_{k+1} = 0$. Note that the basis $(\mathcal{U}_k \quad v_{k+1})$ has the Schur vectors in the front, which is useful for locking as we shall discuss in section 5. In practice, we never compute $\mathcal{U}_k$ at each iteration. Instead, we store $\mathcal{V}_k$ and X_k. Only X_k needs to be computed, which is much cheaper.

4.2. Quadratic problems. The existence of a partial Schur form is guaranteed for the linearized problem (1.2):

$$\begin{bmatrix} K & 0 \\ 0 & M \end{bmatrix} \begin{pmatrix} U_{2k} \\ Y_{2k} \end{pmatrix} = \begin{bmatrix} -iC & M \\ M & 0 \end{bmatrix} \begin{pmatrix} U_{2k} \\ Y_{2k} \end{pmatrix} S_{2k}, \tag{4.4}$$

with U_{2k}, $Y_{2k} \in \mathbf{C}^{n\times 2k}$, and S_{2k} a $2k \times 2k$ upper triangular matrix. Note that U_{2k} does not need to be of full rank. For example, when $C = 0$ and (ω, u) is an eigenpair, $(-\omega, u)$ is also one. The corresponding U matrix is $U_2 = (\alpha u \quad \beta u)$ for some constants α and β which has rank smaller than two.

We define the partial quadratic Schur form as

$$KW_{2k} + iCW_{2k}T_{2k} - MW_{2k}T_{2k}^2 = 0, \tag{4.5}$$

where $W_{2k} \in \mathbf{C}^{n\times 2k}$ has orthonormal columns and $T_{2k} \in \mathbf{C}^{2k\times 2k}$ is upper triangular with the eigenvalues on its main diagonal. The following lemma shows a condition for which a partial quadratic Schur form exists.

LEMMA 4.1. *If U_{2k} from* (4.4) *has full rank, then a partial Schur form* (4.5) *of* (1.1) *exists.*

Proof. From (4.4), it follows that $Y_{2k} = U_{2k}S_{2k}$ and

$$KU_{2k} + iCU_{2k}S_{2k} - MU_{2k}S_{2k}^2 = 0. \tag{4.6}$$

Consider the QR factorization $W_{2k}Z_{2k} = U_{2k}$, where $W_{2k} \in \mathbf{C}^{n\times 2k}$ has orthonormal columns and $Z_{2k} \in \mathbf{C}^{2k\times 2k}$ is upper triangular. Define the upper triangular matrix $T_{2k} = Z_{2k}S_{2k}Z_{2k}^{-1}$. Then (4.5) is satisfied. ∎

We propose computing (4.4) instead of (4.5), since this Schur form is guaranteed to exist. Due to the relationship between the quadratic problem and its linearization and the corresponding solvers, we can use quadratic residual iteration for building the subspace and computing Ritz pairs.

On the kth iteration, we project the linearized problem (1.2) on the subspace with basis

$$\mathcal{V}_{2k} = \begin{bmatrix} V_k & 0 \\ 0 & V_k \end{bmatrix} \tag{4.7}$$

for computing the Schur basis, where the Schur vectors have the form

$$\mathcal{U}_{2k} = \begin{pmatrix} U_{2k} \\ Y_{2k} \end{pmatrix} = \begin{bmatrix} V_k & 0 \\ 0 & V_k \end{bmatrix} \begin{pmatrix} X_1 \\ X_2 \end{pmatrix}. \tag{4.8}$$

With $K_k = V_k^*KV_k$, $C_k = V_k^*CV_k$, and $M_k = V_k^*MV_k$, this projection gives

$$\begin{bmatrix} K_k & \\ & M_k \end{bmatrix} \begin{pmatrix} X_1 \\ X_2 \end{pmatrix} = \begin{bmatrix} -iC_k & M_k \\ M_k & \end{bmatrix} \begin{pmatrix} X_1 \\ X_2 \end{pmatrix} S_{2k}, \tag{4.9}$$

from which $X_2 = X_1S_{2k}$. Hence, the approximate partial Schur form is

$$\begin{bmatrix} K & \\ & M \end{bmatrix} \begin{pmatrix} U_{2k} \\ U_{2k}S_{2k} \end{pmatrix} - \begin{bmatrix} -iC & M \\ M & \end{bmatrix} \begin{pmatrix} U_{2k} \\ U_{2k}S_{2k} \end{pmatrix} S_{2k} = \begin{pmatrix} F_{2k} \\ 0 \end{pmatrix}$$

and

$$KU_{2k} + iCU_{2k}S_{2k} - MU_{2k}S_{2k}^2 = F_{2k}. \tag{4.10}$$

Note that (even with $F_{2k} = 0$) (4.10) is not a partial quadratic Schur factorization since U_{2k} does not have orthogonal columns and is not guaranteed to have full rank.

As mentioned before, we can use the Schur basis $\mathcal{U}_{2k}$ instead of $\mathcal{V}_{2k}$. Instead of adding one vector at the $k+1$st iteration, we now add two vectors to the basis, so the basis at iteration $k+1$ becomes

$$\begin{pmatrix} \mathcal{U}_{2k} & \begin{pmatrix} v_{k+1} & 0 \\ 0 & v_{k+1} \end{pmatrix} \end{pmatrix}.$$

Using (4.8), we can rewrite this as

$$\begin{bmatrix} V_k & v_{k+1} & 0 & 0 \\ 0 & 0 & V_k & v_{k+1} \end{bmatrix} \begin{pmatrix} X_1 & 0 & 0 \\ 0 & 1 & 0 \\ X_2 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

It is thus sufficient to store V_k, v_{k+1} and X_1, X_2 to represent the basis.

For the computation of the Schur form of the linearized problem, we need $H_{2k} = B_{2k}^{-1} A_{2k}$. This is cheaply computed and in a numerically stable way: let

$$B_{2k} = \begin{bmatrix} -iC_k & M_k \\ M_k & 0 \end{bmatrix} \quad \text{and} \quad A_{2k} = \begin{bmatrix} K_k & \\ & M_k \end{bmatrix}.$$

Then $H_{2k} = B_{2k}^{-1} A_{2k}$ can be computed as follows.

ALGORITHM 4.1 (computation of $H_{2k} = B_{2k}^{-1} A_{2k}$). *Compute* $H_{2k} = \begin{bmatrix} 0 & I \\ M_k^{-1} K_k & iM_k^{-1} C_k \end{bmatrix}$, *where the action of* M_k^{-1} *is performed using the Cholesky factorization* $M_k = L_k L_k^*$ *with* L_k *lower triangular. The matrix* $M_k = V_k^* M V_k$ *is well conditioned since* M *is positive definite and has a small condition number.*

5. Locking. Locking of converged eigenvalues is widely used in linear eigenvalue calculations [30, 11, 20, 5]. We first explain the idea of locking for linear problems and then apply this to the quadratic problem (1.2).

5.1. Linear problems. The idea of locking is as follows. Suppose that, at the kth iteration, the Schur factorization is reordered so that we have the decomposition

$$A \begin{pmatrix} \mathcal{U}_q & \mathcal{U}_{k-q} \end{pmatrix} - B \begin{pmatrix} \mathcal{U}_q & \mathcal{U}_{k-q} \end{pmatrix} \begin{bmatrix} S_q & S_{q,k-q} \\ 0 & S_{k-q} \end{bmatrix} = \begin{pmatrix} \mathcal{F}_q & \mathcal{F}_{k-q} \end{pmatrix}, \tag{5.1}$$

with $\|\mathcal{F}_q\|$ smaller than the convergence tolerance. We consider the first q Ritz values and corresponding Schur vectors as converged. In fact, we assume that $\mathcal{F}_q = 0$. The first q Schur vectors $\mathcal{U}_q$ and the Schur matrix S_q are fixed or "locked" in the subsequent iterations, i.e., when new directions are added to the subspace and the Schur vectors are recomputed, the locked vectors are not changed. On the $k+1$st iteration, after the addition of v_{k+1}, we have a new basis $(\mathcal{U}_q \quad \mathcal{V}_{k-q+1}) = (\mathcal{U}_q \quad \mathcal{U}_{k-q} \quad v_{k+1})$, and, after projection, we have the projection equation

$$A \begin{pmatrix} \mathcal{U}_q & \mathcal{V}_{k-q+1} \end{pmatrix} - B \begin{pmatrix} \mathcal{U}_q & \mathcal{V}_{k-q+1} \end{pmatrix} \begin{bmatrix} H_q & H_{q,k-q+1} \\ H_{k-q+1,q} & H_{k-q+1} \end{bmatrix} = \begin{pmatrix} \mathcal{G}_q & \mathcal{G}_{k-q+1} \end{pmatrix} \tag{5.2}$$

with $(\mathcal{U}_q \quad \mathcal{V}_{k-q+1})^* (\mathcal{G}_q \quad \mathcal{G}_{k-q+1}) = 0$, and with the projection matrix

$$\begin{bmatrix} H_q & H_{q,k-q+1} \\ H_{k-q+1,q} & H_{k-q+1} \end{bmatrix} = B_k^{-1} A_k \quad \text{and} \tag{5.3a}$$

$$A_k = \begin{pmatrix} \mathcal{U}_q & \mathcal{V}_{k-q+1} \end{pmatrix}^* A \begin{pmatrix} \mathcal{U}_q & \mathcal{V}_{k-q+1} \end{pmatrix}, \tag{5.3b}$$

$$B_k = \begin{pmatrix} \mathcal{U}_q & \mathcal{V}_{k-q+1} \end{pmatrix}^* B \begin{pmatrix} \mathcal{U}_q & \mathcal{V}_{k-q+1} \end{pmatrix}. \tag{5.3c}$$

Since $\|F_q\|$ is small, we can replace the projection matrix by

$$\begin{bmatrix} S_q & H_{q,k-q+1} \\ & H_{k-q+1} \end{bmatrix}$$

without making a big error. This is deflation or locking. The error made is given by the following theorem.

THEOREM 5.1. *Let*

$$\mathcal{U}_q^* A \mathcal{U}_q - \mathcal{U}_q^* B \mathcal{U}_q S_q = 0, \tag{5.4}$$

$$A\mathcal{U}_q - B\mathcal{U}_q S_q = \mathcal{F}_q, \tag{5.5}$$

and let $(\mathcal{U}_q \quad \mathcal{V}_{k-q+1})$ *have orthonormal columns. Then, with the definition of* (5.3),

$$\left\| \begin{bmatrix} H_q & H_{q,k-q+1} \\ H_{k-q+1,q} & H_{k-q+1} \end{bmatrix} - \begin{bmatrix} S_q & H_{q,k-q+1} \\ 0 & H_{k-q+1} \end{bmatrix} \right\|_F \leq \|B_k^{-1}\| \|\mathcal{V}_{k-q+1}^* \mathcal{F}_q\|_F.$$

Proof. We drop the subscripts for $\mathcal{U}_q$ and $\mathcal{V}_{k-q+1}$. From (5.3), it follows that

$$\begin{aligned} \mathcal{U}^* A\mathcal{U} - \mathcal{U}^* B\mathcal{U} H_q - \mathcal{U}^* B\mathcal{V} H_{k-q+1,q} &= 0, \\ \mathcal{V}^* A\mathcal{U} - \mathcal{V}^* B\mathcal{U} H_q - \mathcal{V}^* B\mathcal{V} H_{k-q+1,q} &= 0. \end{aligned}$$

From (5.4) and (5.5), we have that

$$\begin{aligned} \mathcal{U}^* B\mathcal{U}(S_q - H_q) - \mathcal{U}^* B\mathcal{V} H_{k-q+1,q} &= 0, \\ \mathcal{V}^* B\mathcal{U}(S_q - H_q) - \mathcal{V}^* B\mathcal{V} H_{k-q+1,q} &= -\mathcal{V}^* \mathcal{F}_q. \end{aligned}$$

This gives the linear system

$$\left(\begin{pmatrix} \mathcal{U} & \mathcal{V} \end{pmatrix}^* B \begin{pmatrix} \mathcal{U} & \mathcal{V} \end{pmatrix} \right) \begin{pmatrix} S_q - H_q \\ H_{k-q+1,q} \end{pmatrix} = \begin{pmatrix} 0 \\ -\mathcal{V}^* \mathcal{F}_q \end{pmatrix},$$

which implies that

$$\|S_q - H_q\|_F^2 + \|H_{k-q+1,q}\|_F^2 \leq \left\| \left(\begin{pmatrix} \mathcal{U} & \mathcal{V} \end{pmatrix}^* B \begin{pmatrix} \mathcal{U} & \mathcal{V} \end{pmatrix} \right)^{-1} \right\|_2^2 \|\mathcal{V}^* \mathcal{F}_q\|_F^2.$$

This completes the proof. ∎

The projection equation can now be decomposed as

$$\begin{aligned} A \begin{pmatrix} \mathcal{U}_q & \mathcal{V}_{k-q} \end{pmatrix} - B \begin{pmatrix} \mathcal{U}_q & \mathcal{V}_{k-q+1} \end{pmatrix} \begin{bmatrix} S_q & H_{q,k-q+1} \\ 0 & H_{k-q+1} \end{bmatrix} \\ = \begin{pmatrix} \mathcal{F}_q & \mathcal{F}_{k-q+1} \end{pmatrix}. \end{aligned} \tag{5.6}$$

The new Schur vectors must have the form

$$\begin{pmatrix} \mathcal{U}_q & \mathcal{V}_{k-q+1} \end{pmatrix} \begin{bmatrix} I & 0 \\ 0 & Z_{k-q+1} \end{bmatrix}.$$

By multiplying (5.6) on the right by $\left(\begin{smallmatrix} I & 0 \\ 0 & Z_{k-q+1} \end{smallmatrix}\right)$, it follows that with $\mathcal{U}_{k-q+1} = \mathcal{V}_{k-q+1} Z_{k-q+1}$

$$\begin{aligned} A \begin{pmatrix} \mathcal{U}_q & \mathcal{U}_{k-q+1} \end{pmatrix} - B \begin{pmatrix} \mathcal{U}_q & \mathcal{U}_{k-q+1} \end{pmatrix} \begin{bmatrix} S_q & H_{q,k-q+1} Z_{k-q+1} \\ & S_{k-q+1} \end{bmatrix} \\ = \begin{pmatrix} \mathcal{F}_q & \mathcal{F}_{k-q+1} Z_{k-q+1} \end{pmatrix}, \end{aligned}$$

where $H_{k-q+1} Z_{k-q+1} = Z_{k-q+1} S_{k-q+1}$ is the Schur form of H_{k-q+1}. This allows us to compute Z_{k-q+1} and S_{k-q+1}.

5.2. Quadratic problems. Locking in quadratic residual iteration can be performed via the linearized problem. The mathematics in the last section remains, but two vectors instead of only one are added at each iteration. It is also important to note that it is not necessary to store $\mathcal{U}_{2k}$, but only V_k, X_{2k}, and S_{2k}. Locking then corresponds to locking the first q columns of X_{2k} and S_{2k}.

6. Restarting. The number of basis vectors, k, grows as the algorithms proceed. In practice, this number is limited by storage and computational costs for the Gram–Schmidt orthogonalization. It is possible that convergence to the desired eigenpairs has not occurred when this limit is reached. One way to solve this problem is to reduce the dimension of the subspace by throwing away the uninteresting part. This is called purging.

6.1. Linear problems. We can keep the locked Schur vectors as well as the Schur vectors of interest in the subspace and throw away those we are not interested in. This idea was used for restarting the block Arnoldi method [24], the implicitly restarted Arnoldi method [11, 16], the Jacobi–Davidson method [5], and the rational Krylov method [20, 3]. The main idea is to reorder the Schur form so that unwanted Schur vectors are moved toward the end of the matrix of Schur vectors and to cut the Schur factorization after the pth column. So

$$A \begin{pmatrix} \mathcal{U}_p & \mathcal{U}_{k-p} \end{pmatrix} - B \begin{pmatrix} \mathcal{U}_p & \mathcal{U}_{k-p} \end{pmatrix} \begin{bmatrix} S_p & S_{p,k-p} \\ 0 & S_{k-p} \end{bmatrix} = \begin{pmatrix} \mathcal{F}_p & \mathcal{F}_{k-p} \end{pmatrix}$$

is reduced to

$$A\mathcal{U}_p - B\mathcal{U}_p S_p = \mathcal{F}_p$$

by purging the last $k-p$ Schur vectors. This equation can be expanded by adding new vectors.

In practice, the Schur vectors are computed from the projected Schur form (4.3) as $\mathcal{U}_k = \mathcal{V}_k X_k$ by decomposing $X_k = \begin{pmatrix} X_p & X_{k-p} \end{pmatrix}$ and by using $\mathcal{V}_p = \mathcal{V}_k X_p$ as the new basis.

6.2. Quadratic problems. We want to reduce the approximate Schur form

$$A \begin{pmatrix} U_{2k} \\ U_{2k} S_{2k} \end{pmatrix} - B \begin{pmatrix} U_{2k} \\ U_{2k} S_{2k} \end{pmatrix} S_{2k} = \begin{pmatrix} F_{2k} \\ 0 \end{pmatrix}$$

to

$$A \begin{pmatrix} U_{2p} \\ U_{2p} S_{2p} \end{pmatrix} - B \begin{pmatrix} U_{2p} \\ U_{2p} S_{2p} \end{pmatrix} S_{2p} = \begin{pmatrix} F_{2p} \\ 0 \end{pmatrix}$$

by chopping off the last $2k-2p$ columns of U_{2k} and S_{2k}. This is not possible since the new basis must have the form

$$\mathcal{Y}_{2p} = \begin{bmatrix} W_p & 0 \\ 0 & W_p \end{bmatrix} \begin{pmatrix} Z_{11} & Z_{12} \\ Z_{21} & Z_{22} \end{pmatrix}.$$

We can, however, keep the first p columns of U_{2k}, denoted by U_p, as follows. Consider the QR factorization $U_p = W_p T_p$, where W_p has orthonormal columns and T_p is upper triangular. Let $Z_{11} = T_p$ and $Z_{21} = T_p S_p$ with S_p as the upper left $p \times p$ submatrix of S_{2p}. The remaining blocks Z_{12} and Z_{22} must be chosen so that

$$\begin{pmatrix} T_p & Z_{12} \\ T_p S_p & Z_{22} \end{pmatrix} \tag{6.1}$$

is square and unitary. The last p columns do not represent Schur vectors but are necessary to keep the basis in the form (4.7).

In practice, we proceed as follows.

ALGORITHM 6.1.

1. *Let the first p columns of $U_{2k} = V_k X_1$ be U_p and let $U_p = V_k X_{11}$.*
2. *Compute the QR factorization $X_{11} = QT_p$.*
3. *Decompose*

$$\begin{pmatrix} X_1 \\ X_2 \end{pmatrix} = \begin{pmatrix} X_{11} & X_{12} \\ X_{21} & X_{22} \end{pmatrix}$$

with $X_{11}, X_{21} \in \mathbf{C}^{k\times p}$ and $X_{12}, X_{22} \in \mathbf{C}^{k\times 2k-p}$.
4. *Apply Q*

$$\begin{pmatrix} Q^*X_1 \\ Q^*X_2 \end{pmatrix} = \begin{pmatrix} T_p & Q^*X_{12} \\ T_pS_p & Q^*X_{22} \end{pmatrix}.$$

Remove the last $2k-2p$ columns of X_1 and X_2.
5. *Orthogonalize this matrix to get* (6.1).

Note that the first p columns of $\mathcal{Y}_{2p}$ are the same as those of $\mathcal{U}_{2k}$. When the first q columns of $\mathcal{U}_{2k}$ are locked Schur vectors and $q \le p$, then

$$A\mathcal{Y}_{2p} - B\mathcal{Y}_{2p}\begin{bmatrix} S_q & H_{q,2p-q} \\ 0 & H_{2p-q} \end{bmatrix} = \begin{pmatrix} \mathcal{F}_q & \mathcal{G}_{2p-q} \end{pmatrix},$$

where the lower left block in the Schur matrix is forced to be zero. (See Theorem 5.1.)

7. Building the Schur factorization. We have discussed how to compute an approximate partial Schur form by projection on a subspace and how to use the partial Schur form for restarting purposes and for locking eigenvectors (or Schur vectors). We still have to discuss which vectors will be added to the subspace. Instead of building a basis of eigenvectors in the subspace Range(V_k), we can build a partial Schur form. A Schur form always exists, while the eigendecomposition does not. Moreover, Schur vectors are orthogonal, which is in favor of the numerical stability of the method. In this section, we discuss how we can expand a partial Schur form.

7.1. Linear problems. The Jacobi–Davidson QR (JDQR) and Jacobi–Davidson QZ (JDQZ) methods [5] for solving linear eigenvalue problems compute the partial Schur form $A\mathcal{U}_k = B\mathcal{U}_k S_k$ column by column. Decompose the partial Schur form into

$$A\begin{pmatrix} \mathcal{U}_{k-1} & x \end{pmatrix} = B\begin{pmatrix} \mathcal{U}_{k-1} & x \end{pmatrix}\begin{bmatrix} S_{k-1} & s \\ & \omega \end{bmatrix}.$$

In order to simplify the notation, we use S to denote S_{k-1} and $\mathcal{U}$ for $\mathcal{U}_{k-1}$. Define $\mathcal{Q}$ so that $\mathcal{Q}^*B\mathcal{U} = I$. As a result, $I - B\mathcal{U}Q^*$ is a projector that maps $B\mathcal{U}$ to zero. Also note that $B\mathcal{U}\mathcal{Q}^*(A-\omega B)x = B\mathcal{U}s$. We will discuss later how to choose $\mathcal{Q}$. Since $(I - \mathcal{U}\mathcal{U}^*)x = x$, we can consider (ω, x) as an eigenpair of

$$(I - B\mathcal{U}\mathcal{Q}^*)A(I - \mathcal{U}\mathcal{U}^*)x = \omega(I - B\mathcal{U}\mathcal{Q}^*)B(I - \mathcal{U}\mathcal{U}^*)x. \tag{7.1}$$

Note the resemblance with the Jacobi–Davidson correction equation in (3.1). Suppose $\mathcal{U}$ and S are already computed and we wish to compute x, s, and ω. Suppose an approximation for (ω, x) is available that satisfies the approximate Schur factorization

$$A\begin{pmatrix} \mathcal{U} & x \end{pmatrix} - B\begin{pmatrix} \mathcal{U} & x \end{pmatrix}\begin{bmatrix} S & s \\ & \omega \end{bmatrix} = \begin{pmatrix} 0 & r \end{pmatrix},$$

with $\mathcal{Q}^*r = 0$. We want to compute an improvement by using the Cayley transform on (7.1). The Cayley transform y is solved from the system

$$(I - B\mathcal{U}\mathcal{Q}^*)(A - \sigma B)(I - \mathcal{U}\mathcal{U}^*)y = (I - B\mathcal{U}\mathcal{Q}^*)(A - \omega B)(I - \mathcal{U}\mathcal{U}^*)x.$$

Since $\mathcal{U}^*x = 0$ and assuming that $\mathcal{U}^*y = 0$, we get

$$(I - B\mathcal{U}\mathcal{Q}^*)(A - \sigma B)y = (I - B\mathcal{U}\mathcal{Q}^*)r, \tag{7.2}$$

and so

$$(A - \sigma B)y = r + B\mathcal{U}z, \tag{7.3}$$

where $z = \mathcal{Q}^*(A - \sigma B)y$ is chosen so that $\mathcal{U}^*y = 0$. If $A - \sigma B$ is invertible, the solution can be written as

$$y = (A - \sigma B)^{-1}B\mathcal{U}z + (A - \sigma B)^{-1}r.$$

Since $(A - \sigma B)^{-1}B\mathcal{U} = \mathcal{U}(S - \sigma I)^{-1}$,

$$y = \mathcal{U}\tilde{z} + (A - \sigma B)^{-1}r, \tag{7.4}$$

where $\tilde{z}$ is such that $y \perp \mathcal{U}$. The second term in the right-hand side can be considered as the Cayley transform for the Schur vector x.

7.2. Quadratic problems. In quadratic residual iteration, we want to expand the subspace by the first n components of y without explicitly solving a linear system with $A - \sigma B$. With U denoting U_{k-1}, recall from $\mathcal{U}_k = \left(\begin{smallmatrix} U_k \\ U_k S_k \end{smallmatrix}\right)$ that $\mathcal{U} = \left(\begin{smallmatrix} U \\ US \end{smallmatrix}\right)$ and $x = \left(\begin{smallmatrix} u \\ Us + \omega u \end{smallmatrix}\right)$. Decompose $r = \left(\begin{smallmatrix} r_1 \\ r_2 \end{smallmatrix}\right)$, and then from $r = (A - \omega B)x - B\mathcal{U}s$, we derive

$$\begin{aligned} r_1 &= Ku + i\omega Cu - \omega^2 Mu + iCUs - \omega MUs - MUSs, \\ r_2 &= 0. \end{aligned}$$

Note that r_1 is the last column of the approximate partial Schur factorization

$$K \begin{pmatrix} U & u \end{pmatrix} + iC \begin{pmatrix} U & u \end{pmatrix} \begin{bmatrix} S & s \\ & \omega \end{bmatrix}$$
$$-M \begin{pmatrix} U & u \end{pmatrix} \begin{bmatrix} S & s \\ & \omega \end{bmatrix}^2 = \begin{pmatrix} 0 & r_1 \end{pmatrix}.$$

With $y = \left(\begin{smallmatrix} y_1 \\ y_2 \end{smallmatrix}\right)$, (7.3) becomes

$$(K + i\sigma C)y_1 - \sigma My_2 = r_1 - iCUz + MUSz \tag{7.5a}$$
$$-\sigma My_1 + My_2 = MUz. \tag{7.5b}$$

Decompose $\mathcal{Q} = \left(\begin{smallmatrix} Q_1 \\ Q_2 \end{smallmatrix}\right)$; then $\mathcal{Q}^*r = 0 = Q_1^*r_1$ and

$$z = \begin{pmatrix} Q_1 \\ Q_2 \end{pmatrix}^* \begin{pmatrix} (K + i\sigma C)y_1 - \sigma My_2 \\ -\sigma My_1 + My_2 \end{pmatrix}.$$

By multiplying (7.5b) by σ and adding to (7.5a), we get

$$\begin{aligned} (K + i\sigma C - \sigma^2 M)y_1 &= r_1 + (-iCU + MU(S + \sigma I))z \\ -\sigma My_1 + My_2 &= MUz, \end{aligned} \tag{7.6}$$

with, similarly,

$$z = \begin{pmatrix} Q_1 \\ Q_2 - \sigma Q_1 \end{pmatrix}^* \begin{pmatrix} (K + i\sigma C - \sigma^2 M)y_1 \\ -\sigma M y_1 + M y_2 \end{pmatrix}.$$

The dependence on y_2 disappears when $Q_2 = \sigma Q_1$. Equation (7.4) becomes

$$\begin{aligned} y_1 &= U\tilde{z} + (K + i\sigma C - \sigma^2 M)^{-1} r_1, \\ y_2 &= US\tilde{z} + \sigma(K + i\sigma C - \sigma^2 M)^{-1} r_1 \end{aligned}$$

and shows that y_1 and y_2 point in the same direction orthogonal to U. In a projection method, it is only this direction that matters, so it is sufficient to add y_1 to the subspace. By rearranging (7.6) we obtain

$$(I - (-iCU + MU(S + \sigma I))Q_1^*)(K + i\sigma C - \sigma^2 M)y_1 = r_1,$$

where we still have the condition that $\mathcal{U}^* y = 0$. Since $\tilde{z}$ is determined by this condition but since $\mathcal{U}\tilde{z}$ does not contribute to the expansion of the subspace, we can just as well use the orthogonality condition $U^* y_1 = 0$, which produces another $\tilde{z}$ but produces the same component of y_1 orthogonal to U. This also produces y_2 with the same direction orthogonal to U.

In practice we solve the equation

$$(I - PQ^*)(K + i\sigma C - \sigma^2 M)(I - ZZ^*)y_1 = r_1 \tag{7.7}$$

by an iterative method with $P = iCU - MU(S + \sigma I)$ and $ZT = U$, the QR factorization of U. We still have to make a choice of Q. When U and S are computed by orthogonal projection within quadratic residual iteration or Jacobi–Davidson, the obvious choice is $Q = ZR$ since $Z^* r_1 = 0$, where R is chosen so that $Q^* P = I$. The problem is that PQ^* is an oblique projector and is not guaranteed to exist. A more robust choice is $Q = PR$, but then we first must orthogonalize r_1 against the columns of Q. In our numerical experiments we used $Q = ZR$.

8. Algorithm and numerical examples. The first two examples are easily generated and are included to allow the reader to reproduce the results. The last example is the numerical simulation of poro-elastic material and is less easily reproduced. This is added to illustrate the method on a more realistic problem.

The algorithm that we use for the computation of Ritz pairs is the following one. It uses locking of converged Schur vectors and purging for restarting purposes. On each iteration, the Schur basis of the projected linearized problem is computed. The basis vectors are obtained by Gram–Schmidt orthogonalization with reorthogonalization [2]. The major work lies in the solution of the linear system in step 2.10 and the Gram–Schmidt orthogonalization in step 2.11. When the number of vectors becomes large, the operations on the small scale linearized problem may also become significant. We allow σ to be chosen differently at each iteration. When a direct linear solver is used, however, we prefer to keep σ unchanged for a number of iterations in order to avoid a factorization on each iteration.

ALGORITHM 8.1. *In this algorithm, k is the actual dimension of the subspace, q is the number of locked Schur vectors, satisfying $\|KU_q e_j + iCU_q S_q e_j - MU_q S_q^2 e_j\| \leq$ TOL for $j = 1, \ldots, q$, where $\left\|\begin{pmatrix} U_q \\ U_q S_q \end{pmatrix}\right\| = 1$, p is the dimension after a restart, and m is the maximum subspace dimension.*

1. *Given* $v_1 \in \mathbf{C}^n$ *with* $\|v_1\| = 1$.
   Let $k = 1$ *and* $q = 0$.
2. *Until* $q \geq$ *number of wanted eigenvalues:*
   2.1. *Compute the* k*th row and column of* $K_k = V_k^* K V_k$, $C_k = V_k^* C V_k$, *and* $M_k = V_k^* M V_k$.
   2.2. *Compute* H_{2k} *by Algorithm* 4.1.
   2.3. *Map* H_{2k} *in the Schur basis:*

$$H_{2k} = \begin{pmatrix} X_1 & 0 & 0 \\ 0 & 1 & 0 \\ X_2 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}^* H_{2k} \begin{pmatrix} X_1 & 0 & 0 \\ 0 & 1 & 0 \\ X_2 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

   2.4. *Compute the Schur form* $H_{2k}\binom{X_1}{X_2} = \binom{X_1}{X_2} S_{2k}$, *where the first* q *Schur vectors are locked.*
   2.5. *Order the Schur form so that the main diagonal elements of* S_{2k} *nearest* σ *are in the upper left position.*
   2.6. *For* $j = q + 1$ *to* k*:*
      2.6.1. *Compute the* j*th column of* $R_k = KV_kX_1 + iCV_kX_1S_{2k} - MV_kX_1S_{2k}^2$.
      2.6.3. *If* $\|R_k e_j\| \leq$ TOL, *then* $q = q + 1$.
      Else Go to 2.7.
      End for
   2.7. *If* $k \geq m$, *then*
      2.7.2. *Compute the QR factorization* $X_1 = Q_k T_k$.
      2.7.3. *Compute the basis vectors* $V_p = V_k Q_p$.
      2.7.4. *Update the Schur vectors:* $X_1 = T_p$ *and* $X_2 = Q_p^* X_2$.
      2.7.5. *Add columns to* X_1 *and* X_2 *so that they have appropriate size and form an orthonormal basis* (*see Algorithm* 6.1).
      2.7.6. *Update:* $K_p = Q_p^* K_k Q_p$, $C_p = Q_p^* C_k Q_p$, $M_p = Q_p^* M_k Q_p$.
   2.8. *Increase the subspace dimension* $k = k + 1$.
   2.9. *Select a pole* σ.
   2.10. *Solve the linear system* $(K + i\sigma C - \sigma^2 M)w = R_k e_{q+1}$.
   2.11. *Orthogonalize* w *against* $v_1, \ldots, v_k$ *by Gram–Schmidt into* v_{k+1}.

For our examples, K, C, and M are real symmetric matrices, so, in step 2.1 it is sufficient to compute the kth column. The subspace is expanded by a quadratic Cayley transform applied to a Schur vector. When an iterative solver is used, we can expand the subspace by solving the correction equation (3.2) instead of the Cayley linear system in step 2.10, and we also take into account the locked Schur vectors by employing (7.7), where Z also contains the last (unlocked) Schur vector and Q is the corresponding left-hand side basis. (See sections 9 and 10 for examples.)

8.1. A "linear" problem. For the first example, M is the identity matrix, C is zero, and K is the diagonal matrix with $1^2, 2^2, \ldots, n^2$ on its main diagonal for $n = 1000$. The 10 eigenvalues nearest zero are $\pm j$ for $j = 1, \ldots, 5$, and the corresponding eigenvectors are e_j for the eigenvalue $\pm j$, where e_j is the jth identity vector. Note that the Schur form, defined by (4.5), does not exist, but it does exist for the linearized problem. Incidentally, the linearized problem has $2n$ eigenvectors. We can solve this problem by the spectral transformation Lanczos method [4] since $C = 0$, but we want to illustrate that the algorithm is able to compute linearly dependent eigenvectors of the quadratic eigenvalue problem.

We computed the 10 eigenvalues nearest zero by the use of Algorithm 8.1 with a fixed pole $\sigma = 0$ and $m = 30$ iteration vectors. The initial vector v_1 was $1/\sqrt{n}$ everywhere. All the eigenvalues were computed within 20 iterations to a tolerance (TOL) of 10^{-7}. When only $m = 17$ vectors are used and we compress the dimension to $p = 15$, then before the first restart, 8 eigenvalues have converged. For the convergence of the remaining two eigenvalues, two restarts are required. In total, this corresponds to 20 iterations, which is exactly the same number as for the run with $m = 30$ vectors. This shows that the purging of the Ritz values far away from the pole σ does not necessarily slow down the convergence. A similar result was shown for the implicitly restarted Arnoldi method [16] and for the implicitly filtered rational Krylov method [3].

8.2. A simple quadratic problem. We use the same K and M as for the previous example, but now $C = \alpha I$ with $\alpha = 0.1$. We again computed the 10 eigenvalues nearest zero. We used the same parameters as for the previous example, i.e., $m = 17$, $p = 15$, $\sigma = 0$, and a tolerance of 10^{-7}. The eigenvalues of this example and the previous one are related as follows.

LEMMA 8.1. *Let $u \neq 0$, and let $Ku + i\omega Cu - \omega^2 Mu = 0$; then there is an eigenvalue λ of $Kv = \lambda^2 Mv$ so that*

$$|\lambda - \omega| \leq |\omega| \|M^{-1}\|^{1/2} \frac{\|Cu\|}{u^* Mu}.$$

Proof. Let L be so that $M = L^* L$. Define $v = Lu$ so that

$$\begin{aligned} Ku - \omega^2 Mu &= -i\omega Cu, \\ L^{-*} K L^{-1} v - \omega^2 v &= -i\omega L^{-*} Cu. \end{aligned}$$

The proof follows from the Bauer–Fike theorem (Theorem 2.4) with $\kappa = 1$, since $L^{-*} K L^{-1}$ is Hermitian. Note that $\|L^{-*}\| = \|M^{-1}\|^{1/2}$. ∎

Since $\|C\| = 0.1$ the eigenvalues lie close to those of the previous example. Therefore, the convergence is very similar. It took 20 iterations to compute the ten Ritz values. The first ten Ritz values are given by Table 8.1.

TABLE 8.1
Ritz values for the example in section 8.2.

Real part	Imaginary part	Residual norm
0.99874921	0.0500000	$6.94\,10^{-8}$
−0.99874921	0.0500000	$6.94\,10^{-8}$
1.9993749	0.0500000	$9.58\,10^{-8}$
−1.9993749	0.0500000	$9.57\,10^{-8}$
−2.9995833	0.0500000	$2.88\,10^{-8}$
2.9995833	0.0500000	$2.88\,10^{-8}$
3.9996875	0.0500000	$9.75\,10^{-9}$
−3.9996875	0.0500000	$9.87\,10^{-9}$
−4.9997500	0.0500000	$2.84\,10^{-8}$
4.9997500	0.0500000	$2.85\,10^{-8}$

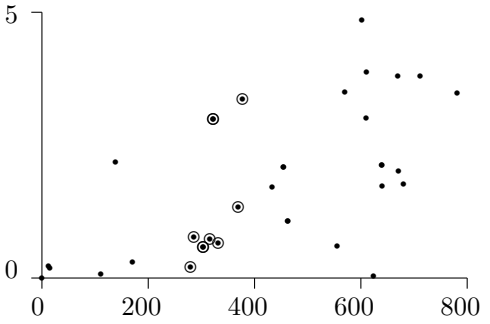


FIG. 8.1. *The eigenvalues of the acoustic simulation of a poro-elastic material between* 0 *and* 800 *Hz are shown as dots. The Ritz values computed by Algorithm* 8.1 *are denoted as circles. Only eight circles are shown, since there are two pairs of double eigenvalues. Notice the different scales for real and imaginary axes.*

8.3. Acoustic simulation of poro-elastic material. This example is related to the acoustic simulation of a $0.4m \times 0.4m \times 0.06m$ sample made of a poro-elastic material. The material is modelled using a two-phase Biot model accounting for kinematic and mechanical interactions between the (elastic) skeleton and the pore (acoustic) fluid [23, 25]. The following material properties have been selected: for the skeleton, the Young modulus is $140000N/m^2$, the Poisson ratio is $0.35(-)$, and the density is $1300kg/m^3$. The pore fluid has density $1.225kg/m^3$, the sound speed is $340m/s$, the porosity is $0.95(-)$, the flow resistivity is $5000Ns/m^4$, the Biot factor is $1(-)$, the fluid bulk modulus is $141600N/m2$, and the tortuosity is $1.2(-)$. The discrete finite-element model relies on a *u-w* formulation [25], where skeleton displacement components (u) and relative fluid displacement components (weighted by the local porosity) (w) are selected as nodal variables. The finite-element mesh has 324 nodes and 192 HEXA8 elements. The total number of degrees of freedom is 1944.

We used Algorithm 8.1 with $m = 30$, $p = 15$, and the number of wanted Ritz values 10. The first pole was 300. A new pole σ was selected after every 10 iterations, as

$$\sigma = \min_{j \geq q+1} \{\mu = \mathrm{Re}((\omega_j + \omega_{j+1})/2), |\omega_j - \mu| \geq 10\},$$

where q is the number of converged Ritz values so that the pole is never close to a Ritz value. This prevents the matrix factorization of almost singular matrices. The tolerance for the eigenvalue problem was $\mathrm{TOL} = 10^{-5}$. (This tolerance is relatively large, since the initial residual norm $\|r_P\|$ is of the order 1.)

The results are obtained by a direct and an iterative method for solving the linear system in step 2.12 of Algorithm 8.1. The direct solver is the package ME47 from the Harwell Subroutine Library (HSL) [8]. The iterative solver is GMRES [22]. The linear systems are solved with a relative residual tolerance $\tau = 10^{-2}$ and $\tau = 10^{-4}$, i.e., $(K + i\sigma C - \sigma^2 M)w = r_P + s$ and $\|s\| \leq \tau \|r_P\|$, where s is the residual of the linear system and $r_P = R_k e_{q+1}$ is the quadratic residual of the Ritz pair. The real part $K - \sigma^2 M$ was used as preconditioner, where the HSL package MA47 was used for the solution of the corresponding linear system. The preconditioner is perhaps not very advisable for practical computations, but this example shows that an iterative method can be used. Recall from section 2 that the quadratic Cayley transform need not be computed very accurately for fast convergence.

The eigenvalues and computed Ritz values are displayed in Figure 8.1. The iterative solver for $\tau = 10^{-2}$ required between 21 and 27 iterations to attain the required

residual tolerance. From the numerical results in Table 8.2, it can be seen that about the same number of Ritz values have converged after 40 iterations, independent of the choice of linear solver. From the 40th iteration on, there is a significantly different convergence behavior. Quadratic residual iteration requires 46 iterations when a direct linear solver is used and 59 iterations when the iterative solver with $\tau = 10^{-2}$ is used. Note that the relative residual tolerance $\tau = 10^{-2}$ for GMRES is quite large. Accurate solves are not necessary. With $\tau = 10^{-4}$, the convergence speed is the same as when a direct linear system solver is used, but the cost per iteration is higher than when $\tau = 10^{-2}$.

TABLE 8.2
Number of converged Ritz pairs for some iteration numbers when a direct and iterative linear solver are used. τ is the relative residual tolerance of the iterative method.

Iteration	Number of converged Ritz values		
	Direct	Iterative $\tau = 10^{-2}$	Iterative $\tau = 10^{-4}$
10	1	1	1
20	3	3	4
30	6	5	6
40	8	7	8
46	10		10
50		9	
59		10	

9. Jacobi–Davidson for the quadratic eigenvalue problem. In this section, we compare the Jacobi–Davidson method with quadratic residual iteration. The only difference from Algorithm 8.1 is in step 2.12, where we solve the correction equation (3.2) instead of computing a Cayley transformation. Since we compute more than one eigenpair, we also add the locked Schur vectors to this basis, i.e., we solve a problem of the form (7.7), where Z spans the $q+1$ first columns of $V_k X_1$ and Q the first $q+1$ columns of $iC(V_k X_1) - M(V_k X_1)(S_{2k} + \sigma I)$.

We now compare the following methods for the application from section 8.3. The purpose is to compute the ten eigenvalues nearest 300.

QRI-D. Quadratic residual iteration with a direct linear solver. The solver used is ME47 from HSL. The pole changes every ten iterations. The first pole is 300.

QRI-G. Quadratic residual iteration with GMRES(30) as linear solver, preconditioned by the direct solver MA47 from HSL for $K - \sigma^2 M$. The pole σ changes every ten iterations. The first pole is 300.

QRI-GP. Quadratic residual iteration with GMRES(30) applied to (7.7), with Z and Q defined as above, preconditioned by MA47. The pole σ changes every ten iterations. The first pole is 300.

QJD. Jacobi–Davidson, where the linear systems are solved by GMRES(30) preconditioned by MA47. For the first ten iterations, the pole was kept constant to 300. From the eleventh iteration on, the pole $\sigma = \omega$ and changes each iteration. The preconditioner changes every ten iterations, so we factorize only a few times.

TABLE 9.1
The number of linear solves for the different algorithms.

Method	Linear solver tolerance τ	Direct linear solves	Eigensolver iterations
QRI-D	—	**45**	46
QRI-G	10^{-2}	**1412**	58
QRI-G	10^{-3}	1545	48
QRI-G	10^{-4}	1976	45
QRI-G	10^{-6}	2998	47
QRI-G	10^{-8}	4130	47
QRI-GP	10^{-2}	**1432**	57
QRI-GP	10^{-3}	1499	47
QRI-GP	10^{-4}	1739	46
QRI-GP	10^{-6}	2318	46
QRI-GP	10^{-8}	3503	45
QJD	10^{-2}	—	—
QJD	10^{-3}	—	—
QJD	10^{-4}	**1469**	33
QJD	10^{-6}	2208	27
QJD	10^{-8}	2918	25

Table 9.1 contains results for the methods listed above for different values of the tolerance of the linear solver. The bold values give the minimum number of linear solves for each method. Clearly, QRI-D is much faster than any other method, but this is because the linear systems with $K + i\sigma C - \sigma^2 M$ are solved exactly. Note, however, that a direct solver for complex linear systems is used while the other methods use a direct solver for real matrices. Let us concentrate on the results obtained with GMRES. The preconditioner is the real part of $K + i\sigma C - \sigma^2 M$, which is in this case $K - \sigma^2 M$. Note that the factorization cost is about the same for all methods since factorization is only performed every ten outer iterations. For QRI-G and QRI-GP, the linear systems need not be solved very accurately to obtain fast convergence. It is remarkable that the number of outer iterations is much larger for the optimal cases. For $\tau = 10^{-6}$ and $\tau = 10^{-8}$ the linear systems are solved more accurately than necessary, since the number of outer iterations does not change significantly. Using a larger τ gives roughly the same number of outer iterations, but less global work. Also note that QRI-GP is more efficient than QRI-G when τ is smaller. We also tried QRI-G with $\sigma \equiv \omega$ as for QJD, but then GMRES stagnates rather quickly. Without the projections in the correction equation (3.2), the matrix of the linear system has an eigenvalue near zero that hinders the convergence. The projections filter away this eigenvalue. For small τ, QJD is definitely faster than any other algorithm. For large τ, QJD stagnates after the locking of the first Ritz value. The first and second eigenvalue form a double one at $303.624 + 0.589i$, and it seems to be difficult to find the second of the double eigenvalue when τ is large. A more important danger arises in the following situation. When we use $\tau = 10^{-8}$ and change the pole after the 6th iteration instead of after the 10th, the Ritz value is $\omega \simeq 303.624 + 0.589i$ and QJD tries to improve the corresponding Ritz vector by solving (7.7). The problem is that there is an eigenvalue at $303.624 + 0.589i$ with multiplicity two, while only one eigenvector is filtered away by the projector $I - ZZ^*$. This means that the linear system is still nearly singular and GMRES stagnates.

10. Shift-and-invert Arnoldi for the linearized problem. The most widely used method in applications is probably the shift-and-invert Arnoldi method applied to the linearized problem (1.2). This method is criticized because it doubles the

size of the problem, leading to an increase in storage cost for the iteration vectors $\mathcal{V}_k$ and in the overall computational cost. The shift-and-invert transformation, $y = (A - \sigma B)^{-1}Bx$, however, can be efficiently computed by the solution of the block structured linear system

$$(A - \sigma B)y = Bx,$$
$$\begin{bmatrix} K + i\sigma C & -\sigma M \\ -\sigma M & M \end{bmatrix} \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} -iCx_1 + Mx_2 \\ Mx_1 \end{pmatrix}.$$

By multiplying the last row by σ and adding to the first row, we get

$$\begin{bmatrix} K + i\sigma C - \sigma^2 M & 0 \\ -\sigma M & M \end{bmatrix} \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} -iCx_1 + M(x_2 + \sigma x_1) \\ Mx_1 \end{pmatrix},$$

from which

$$y_1 = (K + i\sigma C - \sigma^2 M)^{-1}(-iCx_1 + M(x_2 + \sigma x_1)),$$
$$y_2 = x_1 + \sigma y_1.$$

This requires the matrix factorization of $K + i\sigma C - \sigma^2 M$ as in the solution of the quadratic problem.

The Arnoldi method does not produce Ritz vectors that satisfy (2.5) exactly, so the relationship between Algorithms 2.2 and 2.3 is lost. On the other hand, there is a link with Algorithm 2.1, since in exact arithmetic and with the exact linear solves the Davidson method produces the same subspace as the shift-and-invert Arnoldi method when σ is fixed. The major difference between Algorithms 2.1 and 2.3 lies in the construction of the subspaces. In the Algorithms 2.2 and 2.3, the subspace is expanded so that the two components from the Cayley transform applied to a Ritz vector are added. Both algorithms are designed to improve one Ritz vector at a time. In the Arnoldi method, all Ritz vectors converge together.

This is illustrated by the following example. Consider the example from section 8.3. We performed 30 steps of Arnoldi's method applied to $(A - \sigma B)^{-1}B$ starting with a random initial vector. The left-hand picture in Figure 10.1 shows the residual norms $\rho_j = \|Ax_j - BX_jS_j\|$ of the six Ritz values nearest $\sigma = 300$ as a function of the iteration number. The Ritz values nearest σ converge faster, where $X_j = [x_1, \ldots, x_j]$ denote the first j Schur vectors and S_j denotes the corresponding Schur matrix. Most of the Ritz values have decreasing residuals from the first to the last iteration. The central picture shows the results of 30 iterations of quadratic residual iteration. We used Algorithm 8.1 without a restart. We consider a Ritz value as converged when the residual norm $\rho_j = \|r_P\| = \|(KU_j + iCU_jS_j - MU_jS_j^2)e_j\|$ is smaller than the convergence tolerance, 10^{-7}. The horizontal dashed line indicates the tolerance TOL used by Algorithm 8.1 in step 2.3.6. Two different kinds of convergence behavior can be observed. The residual norm of the third Ritz value (dotted line) makes a significant decrease during the convergence of the first and the second. This looks like the convergence behavior of the Arnoldi method. The other Ritz values show a completely different behavior. Each time a Ritz value has converged, it is locked and another Ritz value is targeted and starts converging. This is very clear from the dashed convergence curves. The residuals decrease at the beginning, but most of them stagnate until a new eigenpair is targeted. Peaks in the curves indicate a Ritz value that starts converging to another eigenvalue.

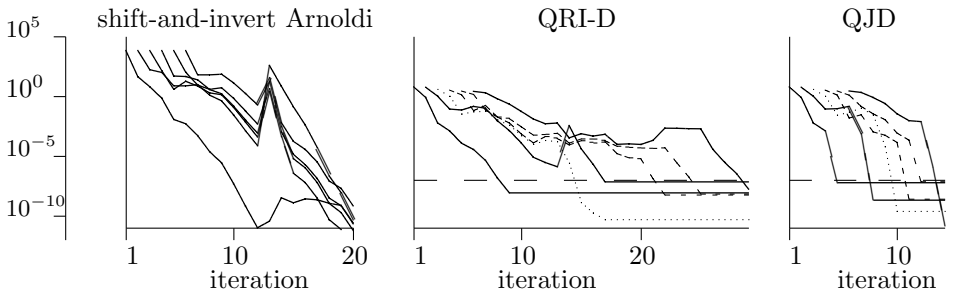


FIG. 10.1. *Convergence behavior of shift-and-invert Arnoldi, quadratic residual iteration (QRI-D), and Jacobi–Davidson (QJD) for the problem from section* 8.3. *The lines show the evolution of* $\rho_j = \|(KU_j + iCU_jS_j - MU_jS_j^2)e_j\|$ *of the six Ritz values nearest* σ.

The tolerance TOL plays an important role in the overall convergence speed. In the example mentioned above, Arnoldi requires 21 iterations for finding six eigenvalues with the required accuracy when TOL$= 10^{-7}$ and quadratic residual iteration 28 iterations. When TOL$= 10^{-4}$ instead, Arnoldi's method requires only 18 iterations, but quadratic residual requires no more than 14. This illustrates that we cannot make a decision on which method is the best.

The right-hand picture shows the results for the Jacobi–Davidson method preconditioned with $K + i\sigma C - \sigma^2 M$ with $\sigma = 300$ and linear solver tolerance $\tau = 10^{-5}$. The first three iterations used a fixed pole $\sigma = 300$, and then $\sigma = \omega$, while the preconditioner is computed once for $\sigma = 300$. The picture shows quadratic convergence for the Ritz values. The algorithm converges in 13 iterations but requires 118 complex linear solves, while less than 30 for the other two algorithms. Stagnation is not as pronounced as for the QRI-D method.

The quadratic residual iteration method has some advantages. First, the modified Davidson framework (Algorithm 2.2) uses a larger subspace than the Davidson approach (Algorithm 2.1) constructed by adding two vectors per iteration step to the subspace. This may lead to some minor gain in convergence speed. The potential gain is in the storage of the iteration vectors. Often, however, one stores the matrices KV_k, MV_k, and CV_k in order to calculate the residuals more efficiently or the projection matrices K_k, M_k, and C_k, and then the advantage is lost. In our implementation, these additional vectors are not stored.

11. Conclusions. In this paper, we have shown there is a close connection between solution methods for the quadratic eigenvalue problem and its linearized form. Solution methods that solve the quadratic problem without linearization appear to be equivalent to methods that solve the linearized problem by projection on a larger subspace. The Jacobi–Davidson correction equation for the linearized problem is connected to the correction equation of the quadratic problem.

If direct linear system solvers are used for the Cayley transform, the gain of the residual iteration method is small compared to the shift-and-invert Arnoldi method. The Gram–Schmidt orthogonalization cost is significantly lower due to the smaller dimension. When the vectors KV_k, CV_k, and MV_k are not stored, about half of the memory is needed, but otherwise the memory consumption is higher than for the shift-and-invert Arnoldi method. The convergence of the Arnoldi method is not focused on a single eigenvalue, but all eigenvalues start converging from the beginning. The quadratic method targets one eigenvalue, which may lead to a fast local

convergence, but a slow global convergence. Therefore, we suggest the use of the shift-and-invert Arnoldi method when direct linear system solvers are used. However, as the last example illustrates, when the convergence tolerance is large, quadratic residual iteration is faster. If iterative linear system solvers are used, we suggest the use of quadratic residual iteration or the Jacobi–Davidson method for the quadratic problem in conjunction with a deflation and restarting scheme based on the linearized case. It is important to note that none of the discussed methods can be flagged as optimal: there are always situations for which one method outperforms the other.

Finally, we want to stress that a practical code should use a block version, i.e., a number of Cayley transforms is applied to more than one Ritz vector at a time. This allows us to compute multiple or clustered eigenvalues more efficiently and may improve the reliability of the method since more than one eigenvalue is targeted simultaneously. When Jacobi–Davidson is used, some care with the choice of poles σ is in order when multiple eigenvalues are expected, since this may lead to the solution of nearly singular linear systems.

Acknowledgments. The author thanks Jennifer Scott, Nick Gould, and Iain Duff from the Rutherford Appleton Laboratory and Françoise Tisseur from the University of Manchester for helpful comments that improved the presentation of the paper. We are also grateful to Jean-Pierre Coyette from Free Field Technologies for the matrices of the acoustic simulation of a poro-elastic material. The author also thanks the referee for suggestions that improved the comparisons between the various methods.

REFERENCES

[1] M. Crouzeix, B. Philippe, and M. Sadkane, *The Davidson method*, SIAM J. Sci. Comput., 15 (1994), pp. 62–76.

[2] J. Daniel, W. Gragg, L. Kaufman, and G. Stewart, *Reorthogonalization and stable algorithms for updating the Gram-Schmidt QR factorization*, Math. Comp., 30 (1976), pp. 772–795.

[3] G. De Samblanx, K. Meerbergen, and A. Bultheel, *The implicit application of a rational filter in the RKS method*, BIT, 37 (1997), pp. 925–947.

[4] T. Ericsson and A. Ruhe, *The spectral transformation Lanczos method for the numerical solution of large sparse generalized symmetric eigenvalue problems*, Math. Comp., 35 (1980), pp. 1251–1268.

[5] D. R. Fokkema, G. L. G. Sleijpen, and H. A. van der Vorst, *Jacobi–Davidson style QR and QZ algorithms for the reduction of matrix pencils*, SIAM J. Sci. Comput., 20 (1998), pp. 94–125.

[6] G. Golub and C. Van Loan, *Matrix Computations*, 3rd ed., The Johns Hopkins University Press, Baltimore, MD, 1996.

[7] R. G. Grimes, J. G. Lewis, and H. D. Simon, *A shifted block Lanczos algorithm for solving sparse symmetric generalized eigenproblems*, SIAM J. Matrix Anal. Appl., 15 (1994), pp. 228–272.

[8] *Harwell Subroutine Library. A Catalogue of Subroutines (Release* 12*)*, 1996. Information: Nick Brealey, AEA Technology, 477 Harwell, Didcot, Oxon, OX11 0RA, UK (hsl@aeat.co.uk).

[9] J. Huitfeldt and A. Ruhe, *A new algorithm for numerical path following applied to an example from hydrodynamical flow*, SIAM J. Sci. Statist. Comput., 11 (1990), pp. 1181–1192.

[10] R. B. Lehoucq and K. Meerbergen, *Using generalized Cayley transformations within an inexact rational Krylov sequence method*, SIAM J. Matrix Anal. Appl., 20 (1998), pp. 131–148.

[11] R. B. Lehoucq and D. C. Sorensen, *Deflation techniques for an implicitly restarted Arnoldi iteration*, SIAM J. Matrix Anal. Appl., 17 (1996), pp. 789–821.

[12] K. Meerbergen, *Robust Methods for the Calculation of Rightmost Eigenvalues of Large Eigenvalue Problems*, Ph.D. thesis, Department of Computer Science, K. U. Leuven, Leuven, Belgium, 1996.

[13] K. Meerbergen and D. Roose, *Matrix transformations for computing rightmost eigenvalues of real nonsymmetric matrices*, IMA J. Numer. Anal., 16 (1996), pp. 297–346.

[14] K. Meerbergen and D. Roose, *The restarted Arnoldi method applied to iterative linear system solvers for the computation of rightmost eigenvalues*, SIAM J. Matrix Anal. Appl., 18 (1997), pp. 1–20.

[15] R. Morgan, *Generalizations of Davidson's method for computing eigenvalues of large nonsymmetric matrices*, J. Comput. Phys., 101 (1992), pp. 287–291.

[16] R. Morgan, *On restarting the Arnoldi method for large nonsymmetric eigenvalue problems*, Math. Comp., 65 (1996), pp. 1213–1230.

[17] R. Morgan and K. Meerbergen, *Inexact methods*, in Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide, Z. Bai, J. Demmel, J. Dongarra, A. Ruhe, and H. van der Vorst, eds., SIAM, Philadelphia, 2000.

[18] R. Natarajan, *An Arnoldi-based iterative scheme for nonsymmetric matrix pencils arising in finite element stability problems*, J. Comput. Phys., 100 (1992), pp. 128–142.

[19] A. Neumaier, *Residual inverse iteration for the nonlinear eigenvalue problem*, SIAM J. Numer. Anal., 22 (1985), pp. 914–923.

[20] A. Ruhe, *Rational Krylov: A practical algorithm for large sparse nonsymmetric matrix pencils*, SIAM J. Sci. Comput., 19 (1998), pp. 1535–1551.

[21] Y. Saad, *Numerical Methods for Large Eigenvalue Problems. Algorithms and Architectures for Advanced Scientific Computing*, Manchester University Press, Manchester, UK, 1992.

[22] Y. Saad and M. H. Schultz, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 856–869.

[23] R. Sandhu and K. Pister, *A variational principle for linear coupled field problems in continuum mechanics*, Internat. J. Engrg. Sci., 8 (1970), pp. 989–999.

[24] J. Scott, *An Arnoldi code for computing selected eigenvalues of sparse unsymmetric matrices*, ACM Trans. Math. Software, 21 (1995), pp. 432–475.

[25] B. Simon, J. Wu, O. Zienkiewicz, and D. Paul, *Evaluation of u-w and u-p finite element methods for the dynamic response of saturated porous media using one-dimensional models*, Int. J. Numer. Anal. Methods Geomech., 10 (1986), pp. 461–482.

[26] G. L. G. Sleijpen, G. Booten, D. Fokkema, and H. A. van der Vorst, *Jacobi Davidson type methods for generalized eigenproblems and polynomial eigenproblems*, BIT, 36 (1996), pp. 595–633.

[27] G. L. G. Sleijpen and H. A. van der Vorst, *A Jacobi–Davidson iteration method for linear eigenvalue problems*, SIAM J. Matrix Anal. Appl., 17 (1996), pp. 401–425.

[28] G. L. G. Sleijpen, H. A. van der Vorst, and M. van Gijzen, *Quadratic eigenproblems are no problem*, SIAM News, 29 (1996), pp. 8–9.

[29] D. C. Sorensen, *Implicit application of polynomial filters in a k-step Arnoldi method*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 357–385.

[30] G. Stewart, *Simultaneous iteration for computing invariant subspaces of non-Hermitian matrices*, Numer. Math., 25 (1976), pp. 123–136.

[31] F. Tisseur, *Backward error and condition of polynomial eigenvalue problems*, Linear Algebra Appl., 309 (2000), pp. 339–361.

[32] M. B. van Gijzen, *The parallel computation of the smallest eigenpair of an acoustic problem with damping*, Internat. J. Numer. Methods Engrg., 45 (1999), pp. 765–777.

SIAM J. SCI. COMPUT.
Vol. 22, No. 5, pp. 1840–1864
© 2001 Society for Industrial and Applied Mathematics

KRYLOV SUBSPACE ESTIMATION*

MICHAEL K. SCHNEIDER† AND ALAN S. WILLSKY†

Abstract. Computing the linear least-squares estimate of a high-dimensional random quantity given noisy data requires solving a large system of linear equations. In many situations, one can solve this system efficiently using a Krylov subspace method, such as the conjugate gradient (CG) algorithm. Computing the estimation error variances is a more intricate task. It is difficult because the error variances are the diagonal elements of a matrix expression involving the inverse of a given matrix. This paper presents a method for using the conjugate search directions generated by the CG algorithm to obtain a convergent approximation to the estimation error variances. The algorithm for computing the error variances falls out naturally from a new estimation-theoretic interpretation of the CG algorithm. This paper discusses this interpretation and convergence issues and presents numerical examples. The examples include a 10^5-dimensional estimation problem from oceanography.

Key words. Krylov subspaces, linear least-squares estimation, error variances, conjugate gradient

AMS subject classifications. 65U05, 65F10, 93E10

PII. S1064827599357292

1. Introduction.

The goal of finite-dimensional linear least-squares estimation is to estimate an l-dimensional random vector x with a linear function of another m-dimensional random vector y so as to minimize the mean squared error [11, Chapter 4]. That is, one minimizes $\mathrm{E}[\|x-\hat{x}(y)\|^2]$ over $\hat{x}(y)$ to find the linear least-squares estimate (LLSE).

Write the relationship between x and y as $y = z+n$, where n is noise, uncorrelated with x, and

$$z = Cx \tag{1.1}$$

for a matrix C reflecting the type of measurements of x. In the Bayesian framework, x, z, and n have known means and covariances. The covariance matrices are denoted by Λ_x, Λ_z, and Λ_n, respectively, and, without loss of generality, the means are assumed to be zero. The LLSE of x given y is

$$\hat{x}(y) = \Lambda_x C^T \Lambda_y^{-1} y, \tag{1.2}$$

where $\Lambda_y = \Lambda_z + \Lambda_n = C\Lambda_x C^T + \Lambda_n$ is the covariance of y.

Direct computation of $\hat{x}(y)$ is difficult if x and y are of high dimension. In particular, the work in this paper was motivated by problems in which x represents a spatially distributed phenomenon and y represents measurements encountered in applications ranging from image processing to remote sensing. For example, when x and y represent natural images, they typically consist of $256 \times 256 = 65536$ pixels.

*Received by the editors June 17, 1999; accepted for publication (in revised form) August 30, 2000; published electronically January 31, 2001. This material is based upon work supported by a National Science Foundation graduate research fellowship, by ONR grant N00014-00-1-1004, and by AFOSR grant F49620-00-0362. A very preliminary paper describing the Krylov subspace estimation algorithm appeared as *A Krylov subspace method for large estimation problems* in Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, IEEE, New York, NY, 1999.

http://www.siam.org/journals/sisc/22-5/35729.html

†Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, Office 35-439, Cambridge, MA 02139 (mikesch@mit.edu, willsky@mit.edu).

In problems from physical oceanography, the dimensions of x and y are typically upwards of 10^5 and 10^4, respectively (e.g., see [7]). Furthermore, in applications such as remote sensing in which the measurement sampling pattern is highly irregular, Λ_z is typically a full matrix that is far from Toeplitz. This prevents one from solving the linear system (1.2) by spectral or sparse matrix methods. However, Λ_y often has a considerable amount of structure. For example, the covariance, Λ_x, of the full spatial field, is often either Toeplitz or well-approximated by a very sparse matrix in an appropriate basis, such as a local cosine basis [12]. The measurement matrix C is often sparse, and the noise covariance Λ_n is often a multiple of the identity. Thus, multiplying vectors by Λ_y is often efficient, and an iterative method for solving linear systems that makes use of multiplications by Λ_y, such as a Krylov subspace method, could be used to compute $\hat{x}(y)$.

For linear least-squares estimation problems, one is interested not only in computing the estimates but also some portion of the estimation error covariance matrix. The covariance of the estimation error,

$$\Lambda_{e_x}(y) = \Lambda_x - \Lambda_x C^T \Lambda_y^{-1} C \Lambda_x, \tag{1.3}$$

is the difference between the prior covariance and the error reduction. The terms on the diagonal of this matrix are the estimation error variances, the quantities most sought after for characterizing the errors in the linear estimate. A natural question to ask is whether a Krylov subspace method for computing the linear estimate $\hat{x}(y)$, such as the method of conjugate gradients (CG), could be adapted for computing portions of the error covariance matrix. This paper presents an interpretation of CG in the context of linear least-squares estimation that leads to a new algorithm for computing estimation error variances.

Many researchers in the geosciences have used CG for computing LLSEs. In particular, Bennett, Chua, and Leslie [1, 2, 3] and da Silva and Guo [4] use CG for computing LLSEs of atmospheric variables. The structures of these estimation problems are very similar to the ones considered here. In particular, the quantities to be estimated, x, are spatially varying processes, and the measurement matrices, C, are sparse. However, they do not consider using a Krylov subspace method for the computation of error variances. We not only propose such a method in this paper but also provide a detailed convergence analysis.

Paige and Saunders [13] and Xu et al. [19, 20, 21, 22] have developed Krylov subspace methods for solving statistical problems that are closely related to linear least-squares estimation. The LSQR algorithm of Paige and Saunders solves a regression problem and can compute approximations to the standard errors. The regression problem is a more general version of linear least-squares estimation in which a prior model is not necessarily specified. In the special case of linear least-squares estimation, the standard errors of the regression problem are the estimation error variances. Thus, LSQR can compute approximations to the error variances. The novelty of our work is that it focuses specifically on linear least-squares estimation and takes advantage of the structure inherent in many prior models for image processing problems. In particular, many such prior models imply a covariance of the data, $\Lambda_y = \Lambda_z + \Lambda_n$, in which the signal covariance matrix, Λ_z, has eigenvalues that decay rapidly to zero and the noise covariance matrix, Λ_n, is a multiple of the identity. Such properties are exploited by our algorithm. These assumptions were also made in the work of Xu, Kailath, et al. for signal subspace tracking. For that problem, one is interested in computing the dominant eigenvectors and eigenvalues of Λ_z. Although computing

the dominant eigenvectors and eigenvalues of Λ_z is sufficient to compute an approximation to the estimation error variances, it is not necessary. We do not explicitly compute eigenvectors or eigenvalues. This provides us with the opportunity to exploit preconditioning techniques in a very efficient manner.

Section 2 discusses our interpretation of CG as used to compute LLSEs. This naturally leads to the presentation of a new iterative algorithm for computing estimation error variances. Section 3 proposes two alternative stopping criteria. The main convergence result is presented in section 4. Techniques for accelerating convergence, including preconditioned and block algorithmic forms, are discussed in section 5. The main convergence result is proved in section 6. Finally, section 7 illustrates the proposed techniques with various numerical examples.

2. The estimation algorithm. The primary difficulty in computing the LLSE $\hat{x}(y)$ in (1.2) is the large dimension of the data y. The signal in the data, however, typically lies in a much lower dimensional subspace. One can take advantage of this fact to compute an approximation to $\hat{x}(y)$ by computing, instead of $\hat{x}(y)$, the LLSE of x given a small number of linear functionals of the data, $p_1^T y, p_2^T y, \ldots, p_k^T y$. For a particular sequence of linearly independent linear functionals, $p_1^T, p_2^T, \ldots, p_k^T$, let $\hat{x}_k(y)$ denote the LLSE of x given $p_1^T y, p_2^T y, \ldots, p_k^T y$. If most of the signal components in y lie in the span of $p_1, p_2, \ldots, p_k$, then the estimate $\hat{x}_k(y)$ approximates $\hat{x}(y)$. In this case, the covariance of the error in the estimate $\hat{x}_k(y)$, $\Lambda_{e_x,k}(y) \triangleq \mathrm{Cov}(x - \hat{x}_k(y))$, approximates the optimal error covariance, $\Lambda_{e_x}(y) \triangleq \mathrm{Cov}(x - \hat{x}(y))$.

The principal novelty of the algorithm we propose in this paper is the use of linear functionals that form bases for Krylov subspaces. The use of Krylov subspaces for solving linear algebra problems is not new, but the application of Krylov subspaces to the computation of error covariances is new. A Krylov subspace of dimension k, generated by a vector s and the matrix Λ_y, is the span of $s, \Lambda_y s, \ldots, \Lambda_y^{k-1} s$ and is denoted by $\mathcal{K}(\Lambda_y, s, k)$ [8, section 9.1.1]. The advantage of using linear functionals that form bases for Krylov subspaces is twofold. One reason is theoretical. Specifically, one can consider the behavior of the angles between $\mathcal{K}(\Lambda_y, s, k)$ and the dominant eigenvectors, u_i, of Λ_y: $\arcsin \|(I - \pi_k)u_i\|/\|u_i\|$, where π_k is the orthogonal projection onto $\mathcal{K}(\Lambda_y, s, k)$. As noted in [16], these angles are rapidly decreasing as k increases. Thus, linear functionals from Krylov subspaces will capture most of the dominant components of the data. Another reason for using functionals from Krylov subspaces is computational. As discussed in the introduction, the structure of Λ_y in many problems is such that multiplying a vector by Λ_y is efficient. A consequence of this fact is that one can generate bases for the Krylov subspaces efficiently.

The specific linear functionals used in this paper are the search directions generated by standard CG for solving a linear system of equations involving the matrix Λ_y. The conjugate search directions, $p_1, \ldots, p_k$, form a basis for $\mathcal{K}(\Lambda_y, s, k)$ and are Λ_y-conjugate [8, section 10.2]. The Λ_y-conjugacy of the search directions implies that $\mathrm{Cov}(p_i^T y, p_j^T y) = \delta_{ij}$; so, these linear functionals of the data are white. Thus, we can draw the novel conclusion that CG whitens the data. The whiteness of the linear functionals of the data allows one to write

$$\hat{x}_k(y) = \sum_{j=1}^{k} \left(\Lambda_x C^T p_i\right) p_i^T y, \tag{2.1}$$

$$\Lambda_{e_x,k}(y) = \Lambda_x - \sum_{j=1}^{k} \left(\Lambda_x C^T p_i\right) \left(\Lambda_x C^T p_i\right)^T, \tag{2.2}$$

which follows from $\mathrm{Cov}(p_1^T y, \ldots, p_k^T y) = I$.[1] One can now write recursions for the estimates and error variances in terms of the quantities $b_{y,k} = \Lambda_x C^T p_k$. We call these the *filtered backprojected* search directions because the prior covariance matrix Λ_x typically acts as a low-pass filter and C^T is a backprojection (as the term is used in tomography) since C is a measurement matrix. In terms of the $b_{y,k}$, the recursions have the following form:

$$\hat{x}_k(y) = \hat{x}_{k-1}(y) + b_{y,k} p_k^T y, \tag{2.3}$$

$$(\Lambda_{e_x,k}(y))_{ii} = (\Lambda_{e_x,k-1}(y))_{ii} - ((b_{y,k})_i)^2, \tag{2.4}$$

with initial conditions

$$\hat{x}_0(y) = 0, \tag{2.5}$$

$$(\Lambda_{e_x,0}(y))_{ii} = (\Lambda_x)_{ii}, \tag{2.6}$$

where $i = 1, \ldots, l$. Unfortunately, the vectors $p_1, p_2, \ldots$ generated by standard CG are not Λ_y-conjugate to a reasonable degree of precision because of the numerical properties of the method.

The numerical difficulties associated with standard CG can be circumvented using a Lanczos iteration, combined with some form of reorthogonalization, to generate the conjugate search directions [8, sections 9.1 and 9.2]. The Lanczos iteration generates a sequence of vectors according to the following recursion:

$$\alpha_k = q_k^T \Lambda_y q_k, \tag{2.7}$$

$$h_k = \Lambda_y q_k - \alpha_k q_k - \beta_k q_{k-1}, \tag{2.8}$$

$$\beta_{k+1} = \|h_k\|, \tag{2.9}$$

$$q_{k+1} = \frac{h_k}{\beta_{k+1}}, \tag{2.10}$$

which is initialized by setting q_1 equal to the starting vector s, $q_0 = 0$, and $\beta_1 = 0$. The Lanczos vectors, $q_1, q_2, \ldots$, are orthonormal and such that

$$\begin{bmatrix} q_1 & q_2 & \cdots & q_k \end{bmatrix}^T \Lambda_y \begin{bmatrix} q_1 & q_2 & \cdots & q_k \end{bmatrix} \tag{2.11}$$

is tridiagonal $\forall k$. Let $T_{y,k}$ denote this tridiagonal matrix, and let $L_{y,k}$ denote the lower bidiagonal Cholesky factor. Then, the vectors defined by

$$\begin{bmatrix} p_1 & p_2 & \cdots & p_k \end{bmatrix} = \begin{bmatrix} q_1 & q_2 & \cdots & q_k \end{bmatrix} L_{y,k}^{-T} \tag{2.12}$$

are equal, up to a sign, to the conjugate search directions generated by CG in exact arithmetic. That $L_{y,k}$ is lower bidiagonal allows one to use a simple one-step recursion to compute the p_i from the q_i. Note also that the $b_{y,k} = \Lambda_x C^T p_i$ can be computed easily in terms of a recursion in $\Lambda_x C^T q_i$. These latter quantities are available since the computation of q_{k+1} requires the product $\Lambda_y q_k = C(\Lambda_x C^T) q_k + \Lambda_n q_k$.

One of the main advantages to using the Lanczos iteration followed by the Cholesky factorization is that one can use a variety of reorthogonalization schemes to ensure

[1] Specifically, (2.1) and (2.2) follow from (1.2) and (1.3) with the substitution of I for Λ_y and $p_1^T C, \ldots, p_k^T C$ for the rows of C.

that the Lanczos vectors remain orthogonal and, in turn, that the associated conjugate search directions are Λ_y-conjugate. The simplest scheme is full orthogonalization [5, section 7.4]. This just recomputes h_k as

$$h_k := h_k - \begin{bmatrix} q_1 & \cdots & q_k \end{bmatrix} \begin{bmatrix} q_1 & \cdots & q_k \end{bmatrix}^T h_k \tag{2.13}$$

between the steps in (2.8) and (2.9). This is typically sufficient to ensure orthogonality among the q_i. However, one can also use more complicated schemes that are more efficient such as selective orthogonalization [15]. A discussion of the details can be found in [18, Appendix B]. We have found that the type of orthogonalization used does not significantly affect the quality of the results.

Although one must use an orthogonalization scheme in conjunction with the Lanczos iteration, the added complexity is not prohibitive. Specifically, consider counting the number of floating point operations (flops) required to perform k iterations. We will assume that full orthogonalization is used and that the number of flops required to multiply vectors by Λ_y is linear in either the dimension m of the data or the dimension l of the estimate. Then, the only contribution to the flop count that is second order or higher in k, l, and m is from the orthogonalization, $2mk^2$. For comparison, consider a direct method for computing the error variances that uses Gaussian elimination to invert the symmetric positive definite Λ_y. The flop count is dominated by the elimination, which requires $m^3/3$ flops [8, p. 146]. Thus, our algorithm typically provides a gain if $k < m/6$. For many estimation problems, a reasonable degree of accuracy is attained for $k \ll m$. Some examples are given in section 7.

A summary of the steps outlined above to compute an approximation to the optimal linear least-squares estimate and associated estimation error variances is as follows.

ALGORITHM 2.1.

1. *Initialize* $\hat{x}_0(y) = 0$, $(\Lambda_{e_x,0}(y))_{ii} = (\Lambda_x)_{ii}$ *for* $i = 1, \ldots, l$.
2. *Generate a random vector* s *to initialize the Lanczos iteration.*
3. *At each step* k,
   (a) *compute the conjugate search direction* p_k *and filtered backprojection* $b_{y,k}$ *using a reorthogonalized Lanczos iteration, and*
   (b) *update*

$$\hat{x}_k(y) = \hat{x}_{k-1}(y) + b_{y,k} p_k^T y, \tag{2.14}$$

$$(\Lambda_{e_x,k}(y))_{ii} = (\Lambda_{e_x,k-1}(y))_{ii} - ((b_{y,k})_i)^2 \quad \text{for } i = 1, \ldots, l. \tag{2.15}$$

3. Stopping criteria. A stopping criterion is needed to determine when a sufficient number of iterations has been run to obtain an adequate approximation to the error variances. Two alternative stopping criteria are proposed in this section. The first is a simple scheme that we have found works well. However, there is no systematic method for setting the parameters of the criterion to guarantee that a specified level of accuracy is achieved. The second stopping criterion is a more complicated scheme for which one can establish bounds on the approximation error. However, the criterion tends to be overly conservative in establishing the number of iterations needed to achieve a specified level of accuracy.

3.1. Windowed-maximal-error criterion. Under this first criterion, the algorithm stops iterating after k steps if

$$\tau_{k,\varepsilon_{\min}} \triangleq \max_{k-K_{\text{win}} \le j \le k} \max_i \frac{((b_{y,j})_i)^2}{\max((\Lambda_{e_x,k}(y))_{ii}, \varepsilon_{\min})} < \varepsilon_{\text{tol}}, \tag{3.1}$$

where K_{win}, $\varepsilon_{\min}$, and ε_{tol} are parameters. This criterion guarantees that no components of the error variances have been altered over the last $K_{\text{win}}+1$ iterations by more than ε_{tol} relative to the current approximation to the error variances. The motivation for this criterion is the theorem in section 4 which implies that the vectors $b_{y,k}$, representing the contribution to error reduction from $p_k^T y$, get smaller as k increases. However, this behavior is not always monotone; so, the criterion takes into account gains over a window of the last few iterations.

3.2. Noiseless-estimation-error criterion. The second stopping criterion examines how well the Krylov subspace at the kth step, $\mathcal{K}(\Lambda_y, s, k-1)$, captures the significant components of the signal z, as defined in (1.1). The motivation for such a criterion is the convergence analysis of section 4. A portion of the analysis examines the optimal error covariance for estimating z from y, $\Lambda_{e_z}(y)$, and its relation to the optimal error covariance for estimating z from $p_1^T y, \ldots, p_k^T y$, $\Lambda_{e_z,k}(y)$. The implication is that as $\Lambda_{e_z,k}(y) - \Lambda_{e_z}(y)$ gets smaller, the difference between $\Lambda_{e_x,k}(y)$ and $\Lambda_{e_x}(y)$ also decreases, albeit possibly at a slower rate. So, a relatively small difference between $\Lambda_{e_z,k}(y)$ and $\Lambda_{e_z}(y)$ implies a relatively small difference between $\Lambda_{e_x,k}(y)$ and $\Lambda_{e_x}(y)$. This fact motivates the interest in efficiently computable bounds for $\Lambda_{e_z,k}(y) - \Lambda_{e_z}(y)$. One such bound can be written, as follows, in terms of the error covariance for the noiseless estimation problem of estimating x from z.

PROPOSITION 3.1. *Suppose $\Lambda_n = \sigma^2 I$ for $\sigma^2 > 0$. Let $\Lambda_{e_z,k}(z)$ be the optimal estimation error covariance for estimating z from $p_1^T z, \ldots, p_k^T z$. Then, the difference between the error covariance for estimating z from y and z from $p_1^T y, \ldots, p_k^T y$ is bounded by*

$$\Lambda_{e_z,k}(y) - \Lambda_{e_z}(y) \le \Lambda_{e_z,k}(z) + f_k f_k^T, \tag{3.2}$$

where

$$\|f_k\|^2 \le \|\Lambda_z p_{k-1}\|^2 + \|\Lambda_z p_k\|^2 + \|\Lambda_z p_{k+1}\|^2 + \|\Lambda_z p_{k+2}\|^2. \tag{3.3}$$

Proof. The proof makes use of the Lanczos vectors q_i discussed at the end of section 2. The Lanczos vectors are useful because they form bases for the Krylov subspaces, and they tridiagonalize both Λ_y and Λ_z since $\Lambda_n = \sigma^2 I$, by assumption. Since the Lanczos vectors tridiagonalize Λ_y, $q_i^T y$ is correlated with $q_j^T y$ if and only if i and j differ by at most one. Let $\Lambda_{r_z,k+1}(y)$ denote the error reduction obtained from estimating z with $q_{k+2}^T y, q_{k+3}^T y, \ldots$. Furthermore, let $\Lambda^{\perp}_{r_z,k+1}(y)$ denote the error reduction obtained from estimating z with the random variable formed by making $q_{k+1}^T y$ uncorrelated with $q_i^T y$ for $i \ne k+1$. Then,

$$\Lambda_{e_z}(y) - \Lambda_{e_z,k}(y) = \Lambda_{r_z,k+1}(y) + \Lambda^{\perp}_{r_z,k+1}(y). \tag{3.4}$$

Since y is simply a noisy version of z, $\Lambda_{r_z,k+1}(y) \le \Lambda_{r_z,k+1}(z)$, where $\Lambda_{r_z,k+1}(z)$ is the error reduction obtained from estimating z with $q_{k+2}^T z, q_{k+3}^T z, \ldots$. Furthermore, $\Lambda_{r_z,k+1}(z) \le \Lambda_{e_z,k}(z)$ because $\Lambda_{e_z}(z) = 0$ and $q_i^T z$ is uncorrelated with $q_j^T z$ if i and j differ by more than one. Combining the last two inequalities with (3.4) yields

$$\Lambda_{e_z,k}(y) - \Lambda_{e_z}(y) \le \Lambda_{e_z,k}(z) + \Lambda^{\perp}_{r_z,k+1}(y). \tag{3.5}$$

The matrix $\Lambda^{\perp}_{r_z,k+1}(y)$ in (3.5) is bounded above by the optimal error reduction for estimating z from $q_k^T y$, $q_{k+1}^T y$, and $q_{k+2}^T y$ since $\Lambda^{\perp}_{r_z,k+1}(y)$ is the error reduction

for an estimator that is linear in these three functionals of y. Furthermore, $\Lambda^{\perp}_{r_z,k+1}(y)$ is bounded above by the optimal error reduction for estimating z from $p_{k-1}^T y, \ldots, p_{k+2}^T y$ since q_k, q_{k+1}, and q_{k+2} are linear combinations of $p_{k-1}, \ldots, p_{k+2}$. Now, write the rank-one matrix $\Lambda^{\perp}_{r_z,k+1}(y)$ as $f_k f_k^T$. Then, the latter bound on $\Lambda^{\perp}_{r_z,k+1}(y)$ implies (3.3). □

Although Proposition 3.1 provides a bound on $\|f_k\|^2$, the argument in the proof suggests that the bound is very weak. Recall from the proof that $f_k f_k^T = \Lambda^{\perp}_{r_z,k+1}(y)$, the error reduction obtained for estimating z from the random variable formed by making $q_{k+1}^T y$ uncorrelated with $q_k^T y$ and $q_{k+2}^T y$. Both q_k and q_{k+2}, as vectors from a Krylov subspace generated by Λ_y, are such that $q_k^T y$ and $q_{k+2}^T y$ are significantly correlated with z. Thus, making $q_{k+1}^T y$ uncorrelated with $q_k^T y$ and $q_{k+2}^T y$ will often significantly reduce the correlation of the resulting quantity with z. As a result, $\Lambda^{\perp}_{r_z,k+1}(y)$ is typically much smaller than the error reduction for estimating z from $q_{k+1}^T y$ alone, which, in turn, is smaller than the right-hand side of (3.3). Thus, the bound on $\|f_k\|^2$ is weak, and $\Lambda_{e_z,k}(z)$, the dominant term in (3.2), could be used alone as the basis of a stopping criterion.

One of the main advantages of the bound in Proposition 3.1 is that the diagonal elements of $\Lambda_{e_z,k}(z)$ are easily computable. As discussed in the proof of Proposition 3.1, the Lanczos vectors $q_1, q_2, \ldots$ generated by Algorithm 2.1 not only tridiagonalize Λ_y; they also tridiagonalize Λ_z:

$$\begin{bmatrix} q_1 & q_2 & \cdots & q_k \end{bmatrix}^T \Lambda_z \begin{bmatrix} q_1 & q_2 & \cdots & q_k \end{bmatrix} = T_{z,k}. \tag{3.6}$$

Let $L_{z,k}$ be the lower bidiagonal Cholesky factor of $T_{z,k}$, and let the vectors $r_1, r_2, \ldots$ be defined by

$$\begin{bmatrix} r_1 & r_2 & \cdots & r_k \end{bmatrix} = \begin{bmatrix} q_1 & q_2 & \cdots & q_k \end{bmatrix} L_{z,k}^{-T}. \tag{3.7}$$

Then, the linear functionals of the signal $r_1^T z, r_2^T z, \ldots$ are white. So, a simple recursion can be used to compute $\Lambda_{e_z,k}(z)$:

$$(\Lambda_{e_z,k}(z))_{ii} = (\Lambda_{e_z,k-1}(z))_{ii} - ((b_{z,k})_i)^2 \tag{3.8}$$

with the initialization

$$(\Lambda_{e_z,0}(z))_{ii} = (\Lambda_z)_{ii}, \tag{3.9}$$

where $i = 1, \ldots, m$ and $b_{z,k} = \Lambda_z r_k$. Note that $b_{z,k}$ can be computed without an additional multiplication by Λ_z since Algorithm 2.1 computes $\Lambda_z q_i$. The computations for calculating $\Lambda_{e_z,k}(z)$ are summarized as follows.

ALGORITHM 3.2.

1. *Initialize* $(\Lambda_{e_z,0}(z))_{ii} = (\Lambda_z)_{ii}$.
2. *At each iteration* k*:*
   (a) *compute* $b_{z,k}$ *using* q_k *and the one-step recursion specified by* $L_{z,k}^T$*, and*
   (b) *update*

$$(\Lambda_{e_z,k}(z))_{ii} = (\Lambda_{e_z,k-1}(z))_{ii} - ((b_{z,k})_i)^2. \tag{3.10}$$

Stopping Algorithm 2.1 when a function of $(\Lambda_{e_z,k}(z))_{ii}$ falls below some threshold has a variety of advantages and disadvantages. Although it may appear that one of the main disadvantages is the requirement that Λ_n must be a multiple of the identity,

this is not the case. There is an extension to the nonwhite case that makes use of preconditioning ideas, as discussed in section 5. In fact, the main disadvantage stems from the bound in Proposition 3.1 being based on the noiseless estimation problem (i.e., $\Lambda_n = 0$). If Λ_n is not small, the bound may not be tight. Thus, a stopping criterion based on $\Lambda_{e_z,k}(z)$ may be conservative in determining the number of iterations needed to guarantee a specified level of accuracy. On the other hand, the bound is easy to compute and provides a good indication of the fraction of error reduction that has been attained by a specific iteration.

4. The main convergence result. In this section, we state the main convergence result. It establishes a bound on the rate at which the approximation to the error variances, in exact arithmetic, converges to the optimal estimation error variances. The result leads naturally to a consideration of the two acceleration techniques discussed in the next section. The proof of the main result is left for section 6.

Establishing the convergence result requires making a few assumptions concerning the estimation problem and starting vector for the algorithm. The first is that the starting vector s in Algorithm 2.1 is a zero-mean Gaussian random vector. This assumption is needed to guarantee the independence of uncorrelated components of s. The covariance matrix of s, Λ_s, is assumed to equal Λ_y or to be proportional to the identity. As regards the estimation problem for the purposes of this section, Λ_n is not necessarily a multiple of the identity. However, we do assume that Λ_y and Λ_z have the same eigenvectors $u_1, u_2, \ldots, u_m$ and that the corresponding eigenvalues $\lambda_{y,1} \geq \lambda_{y,2} \geq \cdots \geq \lambda_{y,m}$ and $\lambda_{z,1} \geq \lambda_{z,2} \geq \cdots \geq \lambda_{z,m}$ satisfy the inequality, $\lambda_{z,i}/\lambda_{y,i} \leq \bar{\lambda}_i/\sigma^2$ for some $\sigma^2 > 0$ and sequence $\bar{\lambda}_i$. Note that both of these statements would hold for $\bar{\lambda}_i = \lambda_{z,i}$ if Λ_n were $\sigma^2 I$. The conditions are stated this generally because Λ_n may not be a multiple of the identity if some of the preconditioning techniques of section 5.1 are used. We also assume that the eigenvalues of Λ_y are distinct and have a relative separation $(\lambda_{y,i} - \lambda_{y,i+1})/(\lambda_{y,i+1} - \lambda_{y,m})$ that is bounded below by a constant $\lambda_{\text{sep}} > 0$. Furthermore, the $\lambda_{y,i}$ are assumed to decrease slowly enough (not faster than a geometric decay) that one can find constants $\zeta > 0$ and $0 < \Gamma < 1$ of reasonable magnitude (ζ not much larger than $\|\Lambda_y\|$) for which $1/(\lambda_{y,k}\gamma^k) < \zeta\Gamma^k$, where

$$\gamma \triangleq 1 + 2\left(\lambda_{\text{sep}} + \sqrt{\lambda_{\text{sep}} + \lambda_{\text{sep}}^2}\right). \tag{4.1}$$

This last assumption is a very weak assumption that is almost never violated. All of these assumptions concerning the estimation problem are not restrictive because they can be guaranteed using appropriate preconditioning techniques, as described in section 5. The assumptions are summarized as follows.

Assumptions.

1. The starting vector s in Algorithm 2.1 is a zero-mean Gaussian random vector, and $\Lambda_s = \Lambda_y$ or $\Lambda_s \propto I$.
2. There exist constants $\zeta > 0$ and $0 < \Gamma < 1$ such that $1/(\lambda_{y,k}\gamma^k) < \zeta\Gamma^k$.
3. Λ_y and Λ_z have the same eigenvectors.
4. There exist a constant $\sigma^2 > 0$ and a sequence $\bar{\lambda}_i$ such that $\lambda_{z,i}/\lambda_{y,i} \leq \bar{\lambda}_i/\sigma^2$.
5. There exists a constant $\lambda_{\text{sep}} > 0$ such that $(\lambda_{y,i} - \lambda_{y,i+1})/(\lambda_{y,i+1} - \lambda_{y,m}) \geq \lambda_{\text{sep}} > 0$.

These assumptions lead to the main convergence result, as stated next in Theorem 4.1. The theorem consists of two bounds, one concerning the error variances for estimating x, and one concerning the error variances for estimating only the measured

components of x, $z = Cx$. Two bounds are given because one may need fewer iterations to obtain a good estimate of z than of x. Moreover, the rate of convergence of the error variance for estimating z is of interest since z is often a subsampled version of x.[2]

THEOREM 4.1. *If Assumptions* 1–5 *hold, then*

(4.2)
$$\sum_{j=1}^{m}(\Lambda_{e_x,k}(y) - \Lambda_{e_x}(y))_{jj} \leq \frac{\|s\|^2\zeta\eta\|\Lambda_x\|\|\Lambda_y\|}{\sigma^2(1-\frac{1}{\gamma^2})(1-\frac{1}{\sqrt[4]{\gamma}})}\gamma^{-k/4} + \frac{\|\Lambda_x\|}{\sigma^2}\sum_{i=k}^{m-1}(i-k+4)\bar{\lambda}_{\lfloor\frac{i}{4}\rfloor}$$

and

$$\sum_{j=1}^{m}(\Lambda_{e_z,k}(y) - \Lambda_{e_z}(y))_{jj} \leq \frac{\|s\|^2\zeta\eta\|\Lambda_y\|}{(1-\frac{1}{\gamma^2})(1-\frac{1}{\sqrt{\gamma}})}\gamma^{-k/2} + \sum_{i=k}^{m-1}(i-k+4)\min\left(\frac{\bar{\lambda}_{\lfloor\frac{i}{4}\rfloor}\lambda_{z,\lfloor\frac{i}{4}\rfloor}}{\sigma^2}, \bar{\lambda}_{\lfloor\frac{i}{4}\rfloor}\right), \tag{4.3}$$

where γ *is given by* (4.1) *and* η *is a random variable whose statistics depend only on* λ_{sep}, γ, *and* Γ.

The bounds in Theorem 4.1 provide a characterization of the difference between the optimal error variances and the computed approximation. The bounds are a sum of two terms. The first terms on the right-hand sides of (4.2) and (4.3) characterize how well the Krylov subspaces have captured the dominant components of Λ_y. The bigger λ_{sep} is, the larger γ is, and the smaller the first terms in (4.2) and (4.3) become. Thus, the more separated the eigenvalues (as measured by λ_{sep}) are, the better the algorithm will perform. The second term is a sum of bounds $\bar{\lambda}_i$ on the ratio of eigenvalues $\lambda_{z,i}/\lambda_{y,i}$. The sum is over those $\bar{\lambda}_i$ corresponding to eigenvectors of Λ_z that are not well captured by the Krylov subspaces at step k. Note that the sum is over the more rapidly decreasing $\bar{\lambda}_i\lambda_{z,i}$ in (4.3).

The bounds are useful principally for two reasons. First, they indicate how the errors will scale as s, σ^2, $\|\Lambda_x\|$, $\|\Lambda_y\|$, and the eigenvalues of Λ_z change. In particular, note that the only dependence on the starting vector s is through the norm $\|s\|$. Thus, the performance of the algorithm does not depend strongly on s. Second, the bounds indicate that the rate of convergence can be increased by transforming the estimation problem in order to make γ big enough so that the second terms in (4.2) and (4.3) dominate. Such transformations are discussed next in section 5.1.

5. Techniques for improving convergence properties. This section presents two different techniques for improving the convergence properties of the proposed algorithm for computing error variances. These techniques can be used to guarantee convergence in the case that a given estimation problem violates any of the assumptions of Theorem 4.1. One can also use these techniques to increase γ so as to improve the theoretical convergence rates.

5.1. Preconditioning. In the estimation context, preconditioning consists of determining an invertible transformation B such that estimating x from the transformed data By can be theoretically done more efficiently by the proposed algorithm

[2]That the two bounds differ is a consequence of the fact that, for a given number of iterations k, we are not computing the best k linear functionals of the data for estimating x.

than estimating x directly from y. This will be the case if the covariances of the transformed data, $B\Lambda_y B^T$, and of the transformed signal, $B\Lambda_z B^T$, satisfy Assumptions 3 and 5 of Theorem 4.1 but Λ_y and Λ_z do not. The convergence properties will also be improved if γ for the transformed problem is higher than for the untransformed problem. The principal novelty of the preconditioning approaches described here is that they focus on these particular goals, which are very different than those of standard CG preconditioning and differ significantly from those of preconditioning for eigenvector algorithms [17, Chapter 8]. Although the goals of the preconditioning discussed here are different than for standard CG, the implementation details are very similar. In particular, explicit specification of a transformation B is not necessarily required for preconditioning techniques because preconditioning can be implemented in such a way that only multiplications by $B^T B$ are needed instead of multiplications by B and B^T.

There are three different implementations of preconditioning, each of which is mathematically equivalent in exact arithmetic. Symmetric preconditioning simply consists of applying the Krylov subspace algorithm to estimating x from $By = BCx + Bn$. Essentially, x is estimated given linear functionals from Krylov subspaces $\mathcal{K}(B\Lambda_y B^T, Bs, k)$ applied to By. There are also left and right preconditioning techniques. The following discussion focuses on right preconditioning, and analogous statements can be made concerning left preconditioning. Right preconditioning differs from symmetric preconditioning in that it involves estimating x given linear functionals from the Krylov subspaces $\mathcal{K}(\Lambda_y B^T B, s, k)$ applied to $B^T By$. Note that this is equivalent to the estimation performed in the case of symmetric preconditioning. Although $\Lambda_y B^T B$ is not symmetric, it is self-adjoint with respect to the $B^T B$ inner product. As in Algorithm 2.1, we do not compute the conjugate search directions for the preconditioned estimation problem using a standard preconditioned CG iteration. Instead, we use Lanczos iterations that compute a series of $B^T B$-conjugate vectors that tridiagonalize $B^T B\Lambda_y B^T B$, as follows:

$$\alpha_k = t_k^T \Lambda_y t_k, \tag{5.1}$$

$$h_k = \Lambda_y t_k - \alpha_k q_k - \beta_k q_{k-1}, \tag{5.2}$$

$$d_k = B^T B h_k, \tag{5.3}$$

$$\beta_{k+1} = \sqrt{d_k^T h_k}, \tag{5.4}$$

$$q_{k+1} = \frac{h_k}{\beta_{k+1}}, \tag{5.5}$$

$$t_{k+1} = \frac{d_k}{\beta_{k+1}}, \tag{5.6}$$

where $t_1 = B^T Bs$, $q_1 = s$, $q_0 = 0$, and $\beta_1 = 0$. The q_k are the $B^T B$-conjugate Lanczos vectors that tridiagonalize $B^T B\Lambda_y B^T B$, and the $t_k = B^T Bq_k$ tridiagonalize Λ_y. This latter tridiagonal matrix can be factored, as in Algorithm 2.1, to compute the Λ_y-conjugate search directions p_k. The only difference is that the t_k replace the q_k in (2.11) and (2.12). Moreover, one can compute the filtered backprojected search directions $b_{y,k} = \Lambda_x C^T p_k$ as a by-product. Overall, the steps of the preconditioned Krylov subspace algorithm are the same as those in Algorithm 2.1 except that a preconditioned Lanczos iteration replaces the normal Lanczos iteration. Note that the Lanczos method for tridiagonalizing a left-preconditioned system is the same as the generalized Lanczos algorithm for solving generalized eigenvalue problems [14,

section 15.11]. What follows are some examples of preconditioners in squared up form, B^TB, that one can consider using in various contexts.

One choice for a preconditioner when the noise covariance Λ_n is not a multiple of the identity but is invertible is to choose $B^TB = \Lambda_n^{-1}$. This choice of preconditioner will effectively shape the noise covariance to be a multiple of the identity. The transformed data covariance, $B\Lambda_yB^T$, and signal covariance, $B\Lambda_zB^T$, will then satisfy Assumption 3. Multiplying a vector by Λ_n^{-1} is often easy because Λ_n is often diagonal.

If the noise covariance is, or has been transformed to be, a multiple of the identity, one can consider preconditioners that will maximally separate the eigenvalues of Λ_y. Such preconditioners can guarantee that the transformed data covariance, $B\Lambda_yB^T$, satisfies Assumption 5 and can increase γ to improve the bounds in Theorem 4.1. Note that such preconditioning will do little to change the bound $\bar{\lambda}_i$ on $\lambda_{z,i}/\lambda_{y,i}$ in Assumption 4 because the preconditioner will transform both $\lambda_{z,i}$ and $\lambda_{y,i}$. The ideal preconditioner would simply operate on the spectrum of Λ_y and force a geometric decay in the eigenvalues to the noise level σ^2. The geometric decay guarantees a constant relative separation in the eigenvalues as measured by the ratio in Assumption 5. However, operating on the spectrum is difficult because one doesn't know the eigendecomposition of Λ_y. When the rows of C are orthogonal (which is often the case in the applications mentioned in the introduction) and the eigendecomposition of Λ_x is known, one practical preconditioner is the following. Let Λ_p be a matrix whose eigenvectors are the same as those of Λ_x and whose eigenvalues decay geometrically. Then, let the preconditioner be given by $B^TB = C\Lambda_pC^T$. Although this preconditioner has worked well in practice, as described in section 7, we have no theoretical results concerning the properties of the transformed estimation problem.

One can use extensions of each of the stopping criteria of section 3 in conjunction with preconditioning; however, the preconditioner must satisfy certain assumptions for the extension of the noiseless-estimation stopping criterion of section 3.2 to be used. What follows is a discussion of the extension and the underlying assumptions concerning the preconditioner for the right-preconditioned case. Recall that the discussion in section 3.2 assumes that the noise covariance is a multiple of the identity. This assumption ensures that the Lanczos vectors tridiagonalize both Λ_y and Λ_z so that one can compute $\Lambda_{e_z,k}(z)$ efficiently. Now, suppose one is using a preconditioning transformation B. Let $\Lambda_{n'} = \Lambda_n - (B^TB)^{-1}$. Assume that $\Lambda_{n'}$ is positive semidefinite so that it is a valid covariance matrix. Let n' be a random vector with covariance $\Lambda_{n'}$ and uncorrelated with z. Then, $z' = z + n'$ has covariance $\Lambda_{z'} = \Lambda_z + \Lambda_{n'}$. One can compute $\Lambda_{e_z,k}(z')$ efficiently because the t_k in (5.1)–(5.6) tridiagonalize both Λ_y and $\Lambda_{z'}$. For $\Lambda_{e_z,k}(z')$ to be useful, the pseudosignal z' should not have any significant components not in z. Note that an example of a preconditioner satisfying the above two assumptions is given by $B^TB = \Lambda_n^{-1}$. For this preconditioner, $\Lambda_{n'} = 0$; so, $\Lambda_{e_z,k}(z) = \Lambda_{e_z,k}(z')$. Thus, one can use $\Lambda_{e_z,k}(z')$ as part of a stopping criterion in conjunction with preconditioning provided that the preconditioner satisfies the two assumptions outlined above.

5.2. Using multiple starting vectors. Another technique for improving convergence properties in the case where Λ_y has repeated eigenvalues is to use a block form of Algorithm 2.1. Block Krylov subspace algorithms have been developed for other computations, particularly eigendecompositions [8, section 9.2.6]. The principal novelty of the algorithm we present here is the application to estimation.

Now consider the subspace spanned by the columns of

$$\begin{bmatrix} S & \Lambda_y S & \Lambda_y^2 S & \cdots & \Lambda_y^{k-1} S \end{bmatrix}, \tag{5.7}$$

where S is an $m \times r$ matrix of independent identically distributed random starting vectors whose marginal statistics satisfy the restrictions for Algorithm 2.1 starting vectors. Denote this subspace by $\mathcal{K}(\Lambda_y, S, k)$. Then, one can consider forming $m \times r$ matrices $P_1, \ldots, P_k$ whose columns form bases for $\mathcal{K}(\Lambda_y, S, k)$ and which satisfy $P_i^T \Lambda_y P_j = \delta_{ij} I$. As for the single starting vector case in section 2, the LLSE of x given the random vectors $P_1^T y, \ldots, P_k^T y$ and the associated error variances can be computing using a recursion:

$$\hat{x}_k(y) = \hat{x}_{k-1}(y) + B_{y,k} P_k^T y, \tag{5.8}$$

$$(\Lambda_{e_x,k}(y))_{ii} = (\Lambda_{e_x,k-1}(y))_{ii} - \sum_{j=1}^{r} ((B_{y,k})_{ij})^2, \tag{5.9}$$

with initial conditions

$$\hat{x}_0(y) = 0, \tag{5.10}$$

$$(\Lambda_{e_x,0}(y))_{ii} = (\Lambda_x)_{ii}, \tag{5.11}$$

where $i = 1, \ldots, l$ and $B_{y,k} = \Lambda_x C^T P_k$.

The P_i and $B_{y,i}$ can be computed using a reorthogonalized block Lanczos algorithm [8, section 9.2.6]. The block Lanczos iteration generates, according to the following recursions, a sequence of orthogonal matrices Q_i that are orthogonal to each other:

$$A_k = Q_k^T \Lambda_y Q_k, \tag{5.12}$$

$$H_k = \Lambda_y Q_k - Q_k A_k - Q_{k-1} R_k, \tag{5.13}$$

$$Q_{k+1} R_{k+1} = H_k \quad \text{(QR factorization of } H_k\text{)}, \tag{5.14}$$

where Q_1 and R_1 are a QR factorization of the starting matrix S, and $Q_0 = 0$. The Q_i block tridiagonalize Λ_y; so, one can write

$$\begin{bmatrix} Q_1 & \cdots & Q_k \end{bmatrix}^T \Lambda_y \begin{bmatrix} Q_1 & \cdots & Q_k \end{bmatrix} = T_{y,k}, \tag{5.15}$$

where $T_{y,k}$ is a block tridiagonal matrix. Let $L_{y,k}$ be the lower block bidiagonal Cholesky factor of $T_{y,k}$. Then, the P_i are defined by

$$\begin{bmatrix} P_1 & \cdots & P_k \end{bmatrix} \triangleq \begin{bmatrix} Q_1 & \cdots & Q_k \end{bmatrix} L_{y,k}^{-T}. \tag{5.16}$$

Thus, the P_i can be computed from the Q_i using a one-step recursion. Moreover, the $B_i = \Lambda_x C^T P_i$ can be computed as a by-product, as with a single starting vector.

As for the single starting vector case in section 2, the block Lanczos iteration must be combined with some form of reorthogonalization. Unlike the previous case, however, there are not as many methods for reorthogonalizing the block Lanczos iteration. Full orthogonalization is very common and is the method we have used. This simply recomputes H_k as

$$H_k := H_k - \begin{bmatrix} Q_1 & \cdots & Q_k \end{bmatrix} \begin{bmatrix} Q_1 & \cdots & Q_k \end{bmatrix}^T H_k \tag{5.17}$$

between steps (5.12) and (5.13).

The algorithm is summarized as follows.

ALGORITHM 5.1.

1. *Initialize* $\hat{x}_0(y) = 0$, $(\Lambda_{e_x,0}(y))_{ii} = (\Lambda_x)_{ii}$ *for* $i = 1, \ldots, l$.
2. *Generate a random* $m \times r$ *matrix* S *to initialize the block Lanczos iteration.*
3. *At each step* k,
   (a) *compute the block of search directions* P_k *and filtered backprojections* $B_{y,k}$ *using a reorthogonalized block Lanczos iteration, and*
   (b) *update*

$$\hat{x}_k(y) = \hat{x}_{k-1}(y) + B_{y,k} P_k^T y, \tag{5.18}$$

$$(\Lambda_{e_x,k}(y))_{ii} = (\Lambda_{e_x,k-1}(y))_{ii} - \sum_{j=1}^{r} ((B_{y,k})_{ij})^2 \quad \text{for } i = 1, \ldots, l. \tag{5.19}$$

The advantage of using the block form is that there may be small angles between the subspaces $\mathcal{K}(\Lambda_y, S, k)$ and multiple orthogonal eigenvectors of Λ_y associated with the same repeated eigenvalue, even in exact arithmetic. This is because each of the columns of S may have linearly independent projections onto the eigenspace associated with a repeated eigenvalue. The following theorem establishes convergence rates for the block case when there may be repeated eigenvalues. It is an extension of Theorem 4.1 to the block case. The proofs of both theorems are very similar, so the proof of Theorem 5.2 is omitted.[3]

THEOREM 5.2. *Suppose the following.*

1. *There exists a constant* $\lambda_{\text{sep},r} > 0$ *such that* $(\lambda_{y,i} - \lambda_{y,i+r})/(\lambda_{y,i+r} - \lambda_{y,m}) \geq \lambda_{\text{sep},r}$.
2. *There exist constants* $\zeta > 0$ *and* $0 < \Gamma < 1$ *such that* $1/(\lambda_{y,i}\gamma_r^i) < \zeta\Gamma^i$, *where*

$$\gamma_r \triangleq 1 + 2\left(\lambda_{\text{sep},r} + \sqrt{\lambda_{\text{sep},r} + \lambda_{\text{sep},r}^2}\right). \tag{5.20}$$

3. Λ_y *and* Λ_z *have the same eigenvectors.*
4. *There exist a constant* $\sigma^2 > 0$ *and a sequence* $\bar{\lambda}_i$ *such that* $\lambda_{z,i}/\lambda_{y,i} \leq \bar{\lambda}_i/\sigma^2$.
5. $(\lambda_{y,i} - \lambda_{y,i_+})/(\lambda_{y,i_+} - \lambda_{y,m})$ *is bounded away from zero, where* i_+ *is the index of the next smallest distinct eigenvalue of* Λ_y *after* i, *and then,*

$$\sum_{j=1}^{m} (\Lambda_{e_x,k}(y) - \Lambda_{e_x}(y))_{jj} \leq \frac{\eta \|\Lambda_x\| \|\Lambda_y\|}{\sigma^2 (1 - \frac{1}{\gamma_r^2})(1 - \frac{1}{\sqrt[4]{\gamma_r}})} \gamma_r^{-k/4} + \frac{\|\Lambda_x\|}{\sigma^2} \sum_{i=k}^{m-1} (i - k + 4) \bar{\lambda}_{\lfloor \frac{i}{4} \rfloor} \tag{5.21}$$

and

$$\sum_{j=1}^{m} (\Lambda_{e_z,k}(y) - \Lambda_{e_z}(y))_{jj} \leq \frac{\eta \|\Lambda_y\|}{(1 - \frac{1}{\gamma_r^2})(1 - \frac{1}{\sqrt{\gamma_r}})} \gamma_r^{-k/2} + \sum_{i=k}^{m-1} (i - k + 4) \min\left(\frac{\bar{\lambda}_{\lfloor \frac{i}{4} \rfloor} \lambda_{z,\lfloor \frac{i}{4} \rfloor}^2}{\sigma^2}, \bar{\lambda}_{\lfloor \frac{i}{4} \rfloor} \right), \tag{5.22}$$

where the statistics of the random variable η *depend on the starting matrix* S.

There are two key differences between the statements of Theorems 4.1 and 5.2. The first addresses the possibility of repeated eigenvalues. Specifically, the bounds in

[3] A proof may be found in [18, Appendix A].

Theorem 5.2 depend on the eigenvalue separation through $\lambda_{\text{sep},r}$, which measures the relative separation between eigenvalues whose indices differ by r. Thus, the proposition establishes a convergence rate in the case where there may be groups of up to r repeated or clustered eigenvalues. The second key difference is that the bounds in Theorem 5.2 may have a strong dependence on the starting matrix. This contrasts with the bounds in Theorem 4.1 which depend on the starting vector s only through the norm $\|s\|$. However, our numerical results have not indicated that the block algorithm's performance depends strongly on the starting matrix S.

One can use natural extensions of the preconditioning techniques and either of the stopping criteria of section 3 with Algorithm 5.1. Thus, Algorithm 5.1 is a simple replacement for Algorithm 2.1 that can be used to obtain better convergence properties when Λ_y has repeated eigenvalues.

6. Convergence analysis. The bounds in Theorem 4.1 are proved in this section in several steps. The first few steps place bounds on the norms of the filtered backprojected conjugate search directions, $\|\Lambda_x C^T p_i\|$ and $\|C\Lambda_x C^T p_i\|$. The bounds are proved using Saad's convergence theory for the Lanczos algorithm [16]. These bounds are stated in terms of an extremum of independent random variables. The extremum arises because the starting vector affects the angles between the Krylov subspaces and the dominant components of Λ_y. However, we prove that the extremum is part of a sequence of extrema that are converging in probability to a finite random variable (η in Theorem 4.1). Thus, the starting vector has no strong effect on the quality of the approximation to the error variances. This result is the principal novelty of our convergence analysis. After establishing the convergence of the extrema, we prove Theorem 4.1.

6.1. Bounds on the filtered backprojected search directions. One is interested in bounding the norms of the filtered backprojected search directions because the quality of the approximation to the error variances depends on the norms as follows:

$$\sum_{j=1}^{l}(\Lambda_{e_x,k}(y) - \Lambda_{e_x}(y))_{jj} = \sum_{i=k+1}^{l} \|\Lambda_x C^T p_i\|^2, \tag{6.1}$$

$$\sum_{j=1}^{l}(\Lambda_{e_z,k}(y) - \Lambda_{e_z}(y))_{jj} = \sum_{i=k+1}^{l} \|C\Lambda_x C^T p_i\|^2. \tag{6.2}$$

PROPOSITION 6.1. *Write the conjugate search directions in the basis of eigenvectors of Λ_y as follows:*

$$p_i = \upsilon_{i,1} u_1 + \cdots + \upsilon_{i,m} u_m. \tag{6.3}$$

Then

$$\|\Lambda_x C^T p_i\|^2 \le \|\Lambda_x\| \sum_{j=1}^{m} \lambda_{z,j} \upsilon_{i,j}^2, \tag{6.4}$$

and

$$\|C\Lambda_x C^T p_i\|^2 = \sum_{j=1}^{m} \lambda_{z,j}^2 \upsilon_{i,j}^2. \tag{6.5}$$

Proof. $\|\Lambda_x C^T p_i\|^2 \le \|\Lambda_x\| \|\Lambda_x^{1/2} C^T p_i\|^2 = \|\Lambda_x\| \sum_{j=1}^m \lambda_{z,j} \upsilon_{i,j}^2$. This proves the first inequality. The second inequality follows from Parseval's theorem. ∎

As we now show, one can bound the coefficients $\upsilon_{i,j}$ in terms of $\|(I-\pi_i)u_j\|/\|\pi_i u_j\|$, where π_i is the operator that produces the orthogonal projection onto $\mathcal{K}(\Lambda_y, s, i)$ with respect to the standard inner product.

PROPOSITION 6.2. *Write* $p_i = \upsilon_{i,1}u_1 + \cdots + \upsilon_{i,m}u_m$ *as in Proposition* 6.1. *Then*

$$|\upsilon_{i+1,j}| \le \frac{\|\Lambda_y\|^{1/2}}{\lambda_{y,j}} \frac{\|(I-\pi_i)u_j\|}{\|\pi_i u_j\|}. \tag{6.6}$$

Proof. Note that

$$\begin{aligned} \lambda_{y,j}|\upsilon_{i+1,j}| &= |p_{i+1}^T \Lambda_y u_j| \\ &= |p_{i+1}^T \Lambda_y \pi_i u_j + p_{i+1}^T \Lambda_y (I-\pi_i) u_j| \\ &= |p_{i+1}^T \Lambda_y (I-\pi_i) u_j| \end{aligned} \tag{6.7}$$

since p_{i+1} is Λ_y-conjugate to vectors in the range of π_i. Thus, $\lambda_{y,j}|\upsilon_{i+1,j}| \le \|\Lambda_y p_{i+1}\| \cdot \|(I-\pi_i)u_j\| \le \|\Lambda_y\|^{1/2} \|(I-\pi_i)u_j\|$ because of the Cauchy–Schwarz inequality and the fact that p_{i+1} is Λ_y-normal. The inequality in (6.6) then follows from $\|\pi_i u_j\| \le 1$. ∎

The bound in Proposition 6.2 can be refined. In particular, a theorem due to Saad [16, Theorem 1] implies the following result concerning the ratio $\|(I-\pi_i)u_j\|/\|\pi_i u_j\|$, which we state without proof.

THEOREM 6.3. *Let* γ *be defined by* (4.1), *and let* K_j *be defined by*

$$K_j \triangleq \begin{cases} \prod_{k=1}^{j-1} \frac{\lambda_{y,k}-\lambda_{y,m}}{\lambda_{y,k}-\lambda_{y,j}} & \text{if } j \ne 1, \\ 1 & \text{if } j = 1. \end{cases} \tag{6.8}$$

Then

$$\frac{\|(I-\pi_i)u_j\|}{\|\pi_i u_j\|} \le \frac{2K_j}{\gamma^{i-j}} \frac{1}{\|\pi_1 u_j\|}. \tag{6.9}$$

Recall, from the definition of angles between subspaces given in section 2, that $\|(I-\pi_i)u_j\|/\|\pi_i u_j\|$ is the tangent of the angle between the Krylov subspace $\mathcal{K}(\Lambda_y, s, i)$ and the eigenvector u_j. Theorem 6.3 bounds the rate at which these angles decrease as the subspace dimension i increases. The bound has three components. The rate of decay is γ, the relative separation between eigenvalues as defined in (4.1). The constant in the numerator, $2K_j$, depends on the eigenvalues according to (6.8). The numerator, $\|\pi_1 u_j\|$, is the norm of the projection of the starting vector, s, onto u_j. The primary importance of the theorem is that it establishes the decay rate γ.

One can refine the bound in Proposition 6.2 by splitting the coefficients $\upsilon_{i,j}$ into two groups: those that are getting small by Proposition 6.2 and Theorem 6.3 and those that may be large but do not significantly affect $\|\Lambda_x C^T p_i\|$ because the corresponding eigenvalues of Λ_z are small. This idea leads to the following proposition.

PROPOSITION 6.4.

$$\|\Lambda_x C^T p_{i+1}\|^2 \le 4\|\Lambda_x\|\|\Lambda_y\| \sum_{j=1}^{\lfloor \frac{i}{4} \rfloor - 1} K_j^2 \frac{1}{\gamma^{2(i-j)} \|\pi_1 u_j\|^2} \frac{\lambda_{z,j}}{\lambda_{y,j}^2} + \|\Lambda_x\| \sum_{j=\lfloor \frac{i}{4} \rfloor}^{\infty} \frac{\lambda_{z,j}}{\lambda_{y,j}}, \tag{6.10}$$

and

$$\|C\Lambda_x C^T p_{i+1}\|^2 \le 4\|\Lambda_y\| \sum_{j=1}^{\lfloor \frac{i}{4} \rfloor - 1} K_j^2 \frac{1}{\gamma^{2(i-j)}\|\pi_1 u_j\|^2} \frac{\lambda_{z,j}^2}{\lambda_{y,j}^2} + \sum_{j=\lfloor \frac{i}{4} \rfloor}^{\infty} \frac{\lambda_{z,j}^2}{\lambda_{y,j}}. \tag{6.11}$$

Proof. The first term in each of (6.10) and (6.11) follows immediately from Propositions 6.1 and 6.2 and Theorem 6.3. The second term follows from Proposition 6.1 and the fact that $p_{i+1}^T \Lambda_y p_{i+1} = \sum_{j=1}^m \lambda_{y,j} v_{i+1,j}^2 = 1$. ∎

The first terms in the bounds of Proposition 6.4 may get large if $1/(\gamma^i \|\pi_1 u_j\|^2)$ or K_j are not well behaved. However, the standing assumptions concerning the eigenvalues of Λ_y, Λ_z, and Λ_s imply that K_j and $1/(\gamma^i \|\pi_1 u_j\|^2)$ are bounded by quantities of a reasonable magnitude, as we now show.

6.2. Convergence of infinite products and extrema of independent sequences. The main result regarding the convergence of infinite products and extrema of independent sequences is the following.

PROPOSITION 6.5. *Let $F_i(v)$, $i = 1, 2, \ldots$, be a sequence of functions such that*

1. *$1 - F_i(v)$ is a cumulative distribution function, i.e., right-continuous and monotonically increasing from zero to one;*
2. *for every interval $[V, \infty)$ over which $1 - F_i(v)$ are positive, there exist a constant $A(V)$ and an absolutely summable sequence $\bar{F}_i(V)$ such that $F_i(V) \le \bar{F}_i(V) \le A(V) < 1$ $\forall i$; and*
3. *$\lim_{v \to \infty} \sum_{i=1}^{\infty} F_i(v) = 0$.*

Then, $F(v) = \prod_{i=1}^{\infty}(1 - F_i(v))$ is a distribution function. Moreover, $F(v)$ is positive over every interval such that $1 - F_i(v)$ is positive $\forall i$.

Proof. For $F(v)$ to be a distribution function, it must be right-continuous and monotonically increasing from zero to one.

Consider the interval $[V, \infty)$. Now, $\sum_{i=1}^{I} \log(1 - F_i(v))$ is right-continuous for each I since each $F_i(v)$ is right-continuous. Furthermore,

$$\begin{aligned} \left| \log(F(v)) - \sum_{i=1}^{I} \log(1 - F_i(v)) \right| &= \left| \sum_{i=I+1}^{\infty} \log(1 - F_i(v)) \right| \le \left| \sum_{i=I+1}^{\infty} \log(1 - \bar{F}_i(V)) \right| \\ &= \left| \sum_{i=I+1}^{\infty} \sum_{j=1}^{\infty} \frac{\bar{F}_i^j(V)}{j} \right| \le \left| \sum_{i=I+1}^{\infty} \frac{\bar{F}_i(V)}{1 - A(V)} \right|. \end{aligned} \tag{6.12}$$

Since $\bar{F}_i(V)$ is absolutely summable, $\sum_{i=1}^{I} \log(1 - F_i(v))$ converges to $\log(F(v))$ uniformly for $v \in [V, \infty)$. Thus, $\log(F(v))$ and, in turn, $F(v)$ are right-continuous.

That $F(v)$ is monotonic follows from the monotonicity of the $1 - F_i(v)$. Now, $\lim_{v \to -\infty} F(v) = 0$ since $\lim_{v \to -\infty}(1 - F_1(v)) = 0$. Moreover,

$$\lim_{v \to \infty} \log(F(v)) \ge \lim_{v \to \infty} \sum_{i=1}^{\infty} \frac{-F_i(v)}{1 - A(V)} = 0, \tag{6.13}$$

where V is such that $1 - F_i(v)$ is positive over $[V, \infty)$ $\forall i$. So, $\lim_{v \to \infty} F(v) = 1$.

Furthermore, if $1 - F_i(v)$ is positive $\forall i$ over an interval $[V, \infty)$, then

$$\log(F(v)) \ge \sum_{i=1}^{\infty} \frac{-\bar{F}_i(V)}{1 - A(V)} > -\infty. \tag{6.14}$$

Hence, $F(v)$ is positive over every interval such that $1 - F_i(v)$ is positive $\forall i$. □

A particular example of such a sequence of functions $F_i(v)$ satisfying the assumptions of Proposition 6.5 is

$$F_i(v) = \begin{cases} 1, & v < 0, \\ (1-v)^i, & 0 \le v \le 1, \\ 0, & v > 1. \end{cases} \tag{6.15}$$

Thus, any product of numbers converging geometrically fast towards one is bounded away from zero, and the product is continuously varying from zero to one as the geometric rate changes from one to zero. This fact is used in the proof of the following proposition, which bounds the constants K_j.

PROPOSITION 6.6. *There exists a function $K(v)$ which is continuous and monotonically decreasing from infinity to one as v ranges from zero to infinity and satisfies*

$$K_j \le K(\lambda_{\text{sep}}). \tag{6.16}$$

Proof.

$$\begin{aligned} \frac{1}{K_j} &= \prod_{k=1}^{j-1} \frac{\lambda_{y,k} - \lambda_{y,j}}{\lambda_{y,k} - \lambda_{y,m}} \\ &\ge \prod_{k=1}^{j-1} \left(1 - \left(\frac{1}{1 + \lambda_{\text{sep}}} \right)^k \right), \end{aligned} \tag{6.17}$$

where the inequality follows from Assumption 5. By Proposition 6.5, the product is monotonically decreasing to a limit as j tends to infinity. The limit is a continuous function of λ_{sep} that varies monotonically from zero to one as λ_{sep} increases from zero to infinity. Denote the limit by $1/K(\lambda_{\text{sep}})$. Then, $K_j \le K(\lambda_{\text{sep}})$, as desired. □

The bound on $1/(\gamma^i \|\pi_1 u_j\|^2)$ is stochastic because $\pi_1 = s^T/\|s\|$, where s is the starting vector. By Assumption 1, one can write $\|\pi_1 u_j\|^2 = \lambda_{s,j}|w_j|^2/\|s\|^2$, where $\lambda_{s,j}$ are eigenvalues of Λ_s and w_j are independent, zero-mean, unit variance Gaussian random variables. Thus,

$$\frac{1}{\gamma^i \|\pi_1 u_j\|^2} \le \|s\|^2 \max_{1 \le k \le m} \frac{1}{\lambda_{s,k} \gamma^k |w_k|^2} \tag{6.18}$$

for $m \ge i \ge j$. Suppose that the $\lambda_{y,k}$ satisfy

$$\frac{1}{\lambda_{y,k} \gamma^k} < \zeta \Gamma^k \tag{6.19}$$

for constants $\zeta > 0$ and $0 < \Gamma < 1$. Then, (6.19) holds for $\lambda_{s,k}$ for the same ζ and Γ if $\Lambda_s = \Lambda_y$ and for a different ζ and $\Gamma = 1/\gamma$ if $\Lambda_s \propto I$. Let

$$\mu_k = \max_{1 \le j \le k} \frac{\Gamma^j}{|w_j|^2}. \tag{6.20}$$

The quantity μ_k is an extremum of an independent, nonidentically distributed sequence of random variables. Bounding the rate at which extrema grow is a classic problem in statistics [10]. The following result states that the μ_k do not grow without bound but converge in probability.

PROPOSITION 6.7. *Suppose $w_1, w_2, w_3, \ldots$ is an independent sequence of zero-mean, unit variance Gaussian random variables. Let μ_k be as in* (6.20). *Then, the μ_k converge in probability to a finite-valued random variable.*

Proof. First, we show the μ_k converge in distribution.

$$\mathrm{P}\{\mu_k \le M\} = \prod_{i=1}^{k} \mathrm{P}\left\{|w_i| \ge \sqrt{\frac{\Gamma^i}{M}}\right\}. \tag{6.21}$$

Let

$$F_i(M) = \mathrm{P}\left\{|w_i| \ge \sqrt{\frac{\Gamma^i}{M}}\right\}. \tag{6.22}$$

Then

$$F_i(M) \le \sqrt{\frac{2}{\pi}}\sqrt{\frac{\Gamma^i}{M}}, \tag{6.23}$$

which satisfies the conditions of Proposition 6.5. Thus, $\lim_{k\to\infty} \mathrm{P}\{\mu_k \le M\} = F(M)$ for some distribution function F.

To show that the μ_k converge in probability, consider the following. For $n > k$ and $\varepsilon > 0$,

$$\mathrm{P}\{\mu_n - \mu_k > \varepsilon\} = \int \mathrm{P}\{\mu_n > \varepsilon + v | \mu_k = v\} dG_k(v), \tag{6.24}$$

where G_k is the distribution of μ_k. Now

$$\begin{aligned} \mathrm{P}\{\mu_n > \varepsilon + v | \mu_k = v\} &= \mathrm{P}\left\{\max_{1\le j\le n-k+1} \frac{\Gamma^j}{|w_j|^2} > \frac{\varepsilon + v}{\Gamma^{k-1}}\right\} \\ &\le 1 - F\left(\frac{\varepsilon + v}{\Gamma^{k-1}}\right). \end{aligned} \tag{6.25}$$

Let V be such that $1 - F(V) < \varepsilon/2$, and let N be such that

$$1 - F\left(\frac{\varepsilon + v}{\Gamma^{k-1}}\right) < \frac{\varepsilon}{2} \quad \text{for } k \ge N. \tag{6.26}$$

For $n > k \ge N$,

$$\begin{aligned} \int \mathrm{P}\{\mu_n > \varepsilon + v | \mu_k = v\} dG_k(v) &= \int_0^V \mathrm{P}\{\mu_n > \varepsilon + v | \mu_k = v\} dG_k(v) \\ &\quad + \int_V^\infty \mathrm{P}\{\mu_n > \varepsilon + v | \mu_k = v\} dG_k(v) \\ &\le \int_0^V \frac{\varepsilon}{2} dG_k(v) + \int_V^\infty dG_k(v) < \varepsilon. \end{aligned} \tag{6.27}$$

Thus, the μ_k satisfy the Cauchy criterion and converge in probability to a random variable whose distribution function is F [6, pp. 226–227]. □

6.3. Proof of Theorem 4.1. The results of the preceding two subsections combine to form a proof of Theorem 4.1 as follows.

Proof. By Propositions 6.4 and 6.6,

$$\sum_{j=1}^{m}(\Lambda_{e_x}(p_1^T y,\dots,p_k^T y))_{jj}-(\Lambda_{e_x}(y))_{jj}=\sum_{i=k+1}^{m}\|\Lambda_x C^T p_i\|^2 \tag{6.28}$$
$$\le 4\|\Lambda_x\|\|\Lambda_y\|\|s\|^2K^2(\lambda_{\mathrm{sep}})\zeta\mu_m\sum_{i=k}^{m-1}\sum_{j=1}^{\lfloor\frac{i}{4}\rfloor-1}\frac{\lambda_{z,j}}{\lambda_{y,j}^2}\frac{1}{\gamma^{(i-2j)}}+\|\Lambda_x\|\sum_{i=k}^{m-1}\sum_{j=\lfloor\frac{i}{4}\rfloor}^{m}\frac{\lambda_{z,j}}{\lambda_{y,j}},$$

and

$$\sum_{j=1}^{m}(\Lambda_{e_z}(p_1^T y,\dots,p_k^T y))_{jj}-(\Lambda_{e_z}(y))_{jj}=\sum_{i=k+1}^{m}\|\Lambda_x C^T p_i\|^2 \tag{6.29}$$
$$\le 4\|\Lambda_y\|\|s\|^2K^2(\lambda_{\mathrm{sep}})\zeta\mu_m\sum_{i=k}^{m-1}\sum_{j=1}^{\lfloor\frac{i}{4}\rfloor-1}\frac{\lambda_{z,j}^2}{\lambda_{y,j}^2}\frac{1}{\gamma^{(i-2j)}}+\sum_{i=k}^{m-1}\sum_{j=\lfloor\frac{i}{4}\rfloor}^{m}\frac{\lambda_{z,j}^2}{\lambda_{y,j}}.$$

By Assumptions 4 and 2, $\lambda_{z,j}/\lambda_{y,j}\le\bar{\lambda}_j/\sigma^2$ and $\bar{\lambda}_j/(\gamma^j\lambda_{y,j})\le\xi$ for a constant ξ. Moreover, $\lambda_{z,i}/\lambda_{y,j}\le 1$, in general. Thus

$$\sum_{j=1}^{m}(\Lambda_{e_x}(p_1^T y,\dots,p_k^T y))_{jj}-(\Lambda_{e_x}(y))_{jj}=\sum_{i=k+1}^{m}\|\Lambda_x C^T p_i\|^2 \tag{6.30}$$
$$\le\frac{4\|\Lambda_x\|\|\Lambda_y\|\|s\|^2K^2(\lambda_{\mathrm{sep}})\zeta\mu_m\xi}{\sigma^2(1-\frac{1}{\gamma^2})}\sum_{i=k}^{m-1}\frac{1}{\gamma^{i/4}}+\frac{\|\Lambda_x\|}{\sigma^2}\sum_{i=k}^{m-1}(i-k+4)\bar{\lambda}_{\lfloor\frac{i}{4}\rfloor},$$

and

$$\sum_{j=1}^{m}(\Lambda_{e_z}(p_1^T y,\dots,p_k^T y))_{jj}-(\Lambda_{e_z}(y))_{jj}=\sum_{i=k+1}^{m}\|C\Lambda_x C^T p_i\|^2 \tag{6.31}$$
$$\le\frac{4\|\Lambda_y\|\|s\|^2K^2(\lambda_{\mathrm{sep}})\zeta\mu_m}{(1-\frac{1}{\gamma^2})}\sum_{i=k}^{m-1}\frac{1}{\gamma^{i/2}}+\sum_{i=k}^{m-1}(i-k+4)\min\left(\frac{\bar{\lambda}_{\lfloor\frac{i}{4}\rfloor}\lambda_{z,\lfloor\frac{i}{4}\rfloor}}{\sigma^2},\bar{\lambda}_{\lfloor\frac{i}{4}\rfloor}\right).$$

The increasing μ_m converge in probability to a random variable μ by Proposition 6.7. Equations (4.2) and (4.3) follow immediately from (6.30) and (6.31). ∎

The analysis presented here predicts actual convergence behaviors, as illustrated next with the numerical examples in section 7.

7. Numerical examples. The following numerical examples illustrate the actual performance of the algorithm in relation to the theory of the previous sections. There are four different examples. Each one illustrates a different aspect of the theory. The estimation problems in each of the examples is different. The breadth of estimation problems provides a glimpse at the range of applicability of the Krylov subspace estimation algorithm. For each of the following problems, full orthogonalization was used, except as noted.

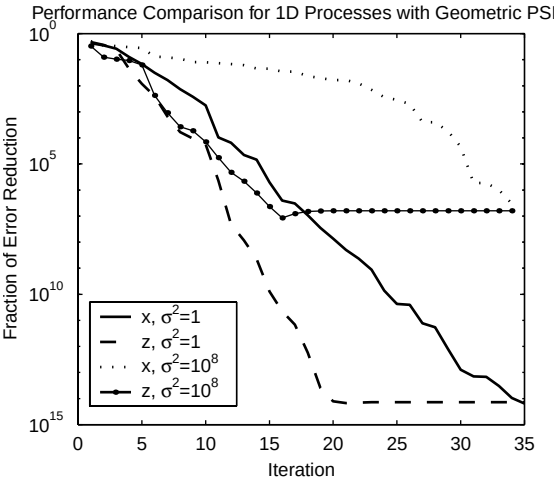


FIG. 7.1. *The four curves plotted here show the convergence behaviors when computing error variances for estimating two different quantities in two slightly different estimation problems. One of the quantities to be estimated is a* 1*-D process,* x*, and the other is a subsampled version of the same process,* z*. Both quantities are estimated from measurements consisting of* z *embedded in additive noise. The only difference between the two estimation problems is the variance of the noise,* σ^2*, which is* 1 *in one case and* 10^{-8} *in the other. The curves indicate that convergence is slower for lower* σ^2 *and for estimating* x*, as predicted by Theorem* 4.1.

The results in Figure 7.1 illustrate the relationship between the actual performance of the algorithm and that predicted by Theorem 4.1. The estimation problem consists of estimating 1024 samples of a stationary process, x, on a 1-D torus from 512 consecutive point measurements, y. The power spectral density (PSD) of x has a geometric decay, $S_{xx}(\omega) \propto (0.3)^{|\omega|}$, and is normalized so that the variance of x is one. Depicted in Figure 7.1 are the fractions of error reduction obtained for estimating x,

$$\frac{\sum_{i=1}^{l}(\Lambda_{e_x,k}(y) - \Lambda_{e_x}(y))_{ii}}{\sum_{i=1}^{l}(\Lambda_x - \Lambda_{e_x}(y))_{ii}}, \tag{7.1}$$

and z,

$$\frac{\sum_{i=1}^{l}(\Lambda_{e_z,k}(y) - \Lambda_{e_z}(y))_{ii}}{\sum_{i=1}^{l}(\Lambda_z - \Lambda_{e_z}(y))_{ii}}, \tag{7.2}$$

where $\Lambda_n = \sigma^2 I$ for $\sigma^2 = 1$ and $\sigma^2 = 10^{-8}$. Note that the numerators in (7.1) and (7.2) are the terms bounded in Theorem 4.1 and that the denominators are independent of the iteration index, k. The reference values $\Lambda_{e_x}(y)$ and $\Lambda_{e_z}(y)$ are computed using direct methods in MATLAB. The numerical errors in these direct methods tend to dominate after several iterations especially for $\sigma^2 = 10^{-8}$. Note that the eigenvalues of Λ_x and Λ_z satisfy $\lambda_{x,i} \geq \lambda_{z,i} \geq \lambda_{x,l-m+i}$ as a consequence of Cauchy's interlace theorem [9, Theorem 4.3.15] and the rows of the measurement matrix C being orthogonal. Since the PSD (collection of eigenvalues) display a two-sided geometric decay, Λ_z and, in turn, $\Lambda_y = \Lambda_z + \sigma^2 I$ may have eigenvalue multiplicities of order two. However, the plots show a geometric rate of convergence consistent with a geometrical decay of Λ_y despite the fact that the block form of the algorithm is not used. A block form is not necessary because roundoff error will introduce components of the

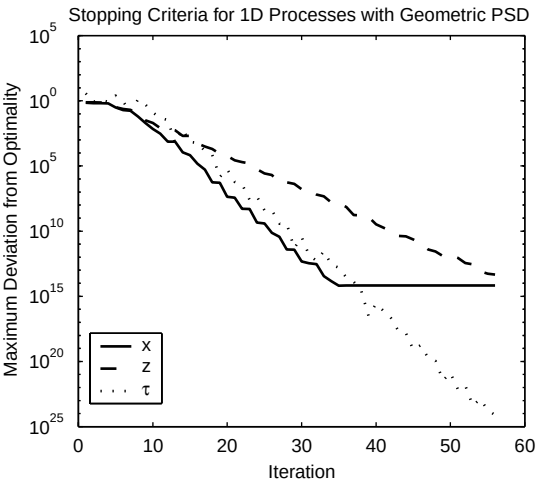


FIG. 7.2. *The results plotted here indicate how the computable quantities making up the two stopping criteria of section* 3 *relate to the difference between the computed approximation to the error covariance for estimating* x *at iteration* k *and the optimal error covariance,* $\Lambda_{e_x,k}(y) - \Lambda_{e_x}(y)$. *The solid line is the maximal difference between the computed and optimal error variances for estimating* x, $\max_i(\Lambda_{e_x,k}(y) - \Lambda_{e_x}(y))_{ii}$. *Each of the other two curves plot the quantities making up the two stopping criteria. The dashed line is the maximal error variance for estimating* z, $\max_i(\Lambda_{e_z,k}(z))_{ii}$, *and the dotted line is the maximum change made to the error variances at the current iteration,* $\tau_{k,0}$, *as defined in* (3.1) *for* $K_{\text{win}} = 0$.

eigenvectors of Λ_y into the Krylov subspaces that are not present in the starting vector [15, p. 228]. Note also that, as suggested by Theorem 4.1, the rate of convergence is faster for the error variances at measurement locations, i.e., for estimates of z, than away from measurement locations, i.e., for estimates of all of x. The theorem also suggests that convergence is slower for smaller σ^2, which is evident in Figure 7.1. Thus, Theorem 4.1 accurately predicts convergence behavior.

Figure 7.2 depicts how the two stopping criteria relate to the difference between the computed approximation to the error covariance for estimating x at iteration k and the optimal error covariance, $\Lambda_{e_x,k}(y) - \Lambda_{e_x}(y)$. The process to be estimated is the same one previously described. The measurement locations are chosen randomly. At any given location, the chance that there is a measurement is 50% and is independent of there being a measurement at any other sample point. The measurement noise covariance matrix is a diagonal matrix whose elements vary according to the following triangle function:

$$(\Lambda_n)_{ii} = \begin{cases} 9\frac{i-1}{\lfloor m/2 \rfloor - 1} + 1 & \text{for } 1 \le i \le \lfloor m/2 \rfloor, \\ 9\frac{m-i}{m - \lfloor m/2 \rfloor - 1} + 1 & \text{for } \lfloor m/2 \rfloor + 1 \le i \le m. \end{cases} \tag{7.3}$$

A whitening preconditioner, Λ_n^{-1}, is used. The figure contains plots of the maximal difference between the computed and optimal error variances for estimating x, $\max_i(\Lambda_{e_x,k}(y) - \Lambda_{e_x}(y))_{ii}$. There are also plots of the two quantities making up each of the two stopping criteria. One is of the maximal error variance for estimating z, $\max_i(\Lambda_{e_z,k}(z))_{ii}$, and the other is of the maximum change made to the error variances at the current iteration, $\tau_{k,0}$, as defined in (3.1). Note that $\Lambda_{e_z,k}(z)$ is a bound on $\Lambda_{e_x,k}(y) - \Lambda_{e_x}(y)$, but that the rates of convergence of these two quantities are different. The $\tau_{k,0}$, on the other hand, are more erratic but decrease at a rate close

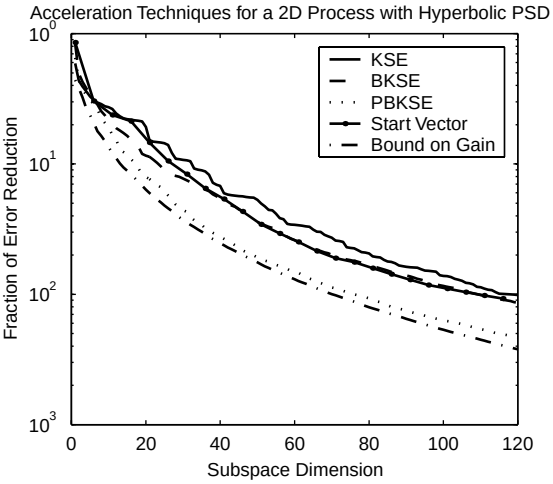


FIG. 7.3. *The results plotted here indicate that various acceleration techniques can be used to achieve nearly optimal performance. The curves depict the fraction of error reduction for estimating x for different methods of choosing linear functionals of the data. The figure shows the results for the standard Krylov subspace estimation algorithm (KSE), a block form with a block size of* 2 *(BKSE), and a preconditioned block form (PBKSE), also with a block size of* 2*. For comparison, the figure shows two additional curves. One (Start Vector) is of the results for Algorithm* 2.1 *modified to start with a linear combination of the first* 60 *eigenvectors of Λ_y. The other (Bound on Gain) is of the fraction of error reduction attained by using the optimal linear functionals of the data.*

to $\Lambda_{e_x,k}(y) - \Lambda_{e_x}(y)$. Stopping when $\tau_{k,\varepsilon_{\min}}$ falls below a threshold has been the most successful criterion because the $\tau_{k,\varepsilon_{\min}}$ give a good indication of the rate of decrease of $\max_i(\Lambda_{e_x,k}(y) - \Lambda_{e_x}(y))_{ii}$. However, stopping when $\max_i(\Lambda_{e_z,k}(z))_{ii}$ falls below a threshold is a preferable criterion when the noise intensity is small primarily because $\max_i(\Lambda_{e_z,k}(z))_{ii}$ provides a tight bound on $\max_i(\Lambda_{e_x,k}(y) - \Lambda_{e_x}(y))_{ii}$.

A comparison among various techniques to accelerate convergence is provided in Figure 7.3. The estimation problem consists of estimating a stationary random field, x, on a 32×32 toroidal grid from point measurements, y, of equal quality taken over one 32×16 rectangle. The PSD of x is proportional to $1/(|\omega| + 1)^3$ and is normalized so that the variance of x is one. The measurement noise covariance matrix Λ_n is $4I$. The plots are of the fraction of error reduction attained for estimating x, as defined by (7.1), versus the Krylov subspace dimensions. Both a right-preconditioned and a block form are considered. The preconditioner has the form $C\Lambda_p C^T$, as described in section 5.1. A simple block algorithm (BKSE) with a block size of 2 does not do much better than the standard algorithm (KSE). However, a preconditioned block form (PBKSE) requires considerably fewer iterations to achieve a given level of accuracy than the standard algorithm. The error reduction attained by using the optimal linear functionals of the data is also plotted in Figure 7.3. The performance of PBKSE is close to the optimal performance. Figure 7.3 also shows the results of an experiment to determine whether one can gain much by picking a good starting vector. A starting vector with components in each of the first 60 eigenvectors of Λ_y was used to start a run. The results are plotted in Figure 7.3 and are comparable to those of BKSE, indicating that one does not gain much by picking a good starting vector. That the choice of starting vector should have little impact on the results is a consequence of Proposition 6.7.

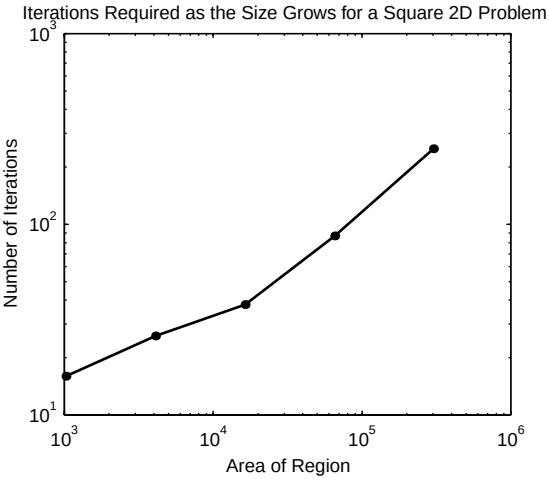


FIG. 7.4. *The number of iterations required for a practical* 2*-D problem of interest is not very large and grows no more than linearly with the area of the region of interest.*

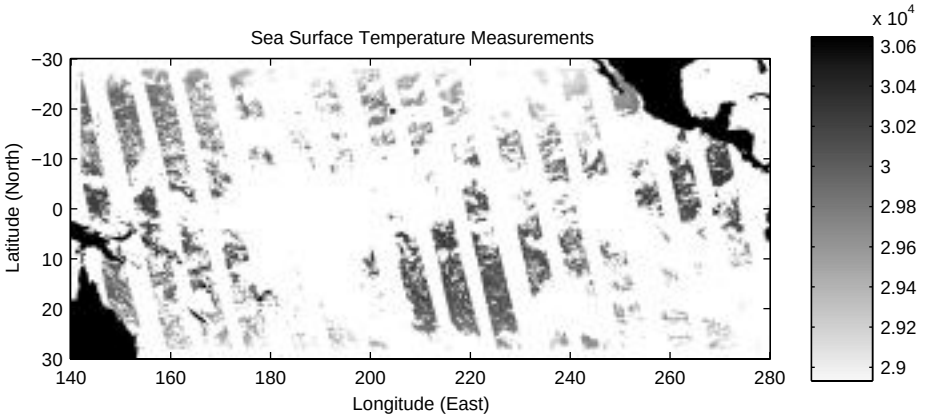


FIG. 7.5. *These data are satellite measurements of sea surface temperature. Measurements are taken only along satellite tracks with no obscuring cloud cover.*

Lastly, Figure 7.4 shows how the number of iterations grows with the region size for the problem of estimating deviations from mean sea surface temperature, x, from the satellite data, y, in Figure 7.5 [7]. The temperature deviations are estimated on a rectangular grid and are assumed to be stationary with a Gaussian-shaped covariance function. The width of the Gaussian is 60 pixels, and the height is 9×10^4. The measurements are very scattered because they exist only along the satellite tracks where there is no obscuring cloud cover (see Figure 7.5). The measurement noise covariance Λ_n is $400I$. Figure 7.4 shows how the number of iterations needed to satisfy $\tau_{k,10^{-2}} < 10^{-2}$ for $K_{\text{win}} = 8$ grows as a region of interest grows. Note that the measurement density in these regions varies from approximately $10 - 20\%$. The growth in the number of iterations is less than linear as the area of the region grows. One expects this behavior because one should need an increasing number of linear functionals as the region grows, but the growth should be no more than linear in the

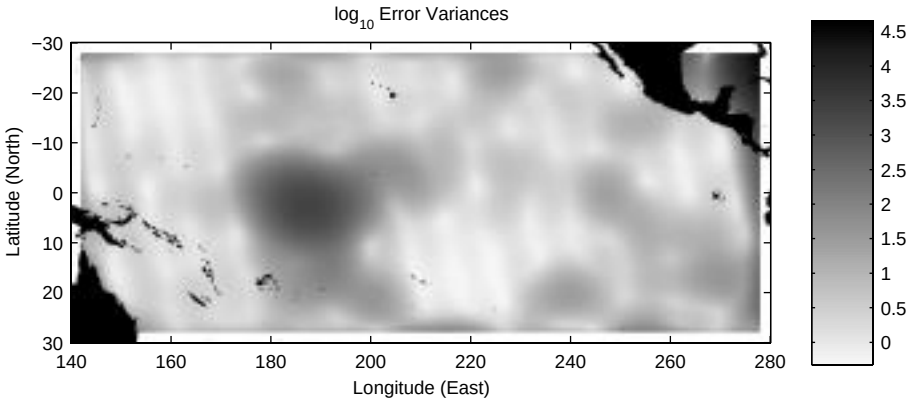


FIG. 7.6. *The Krylov subspace estimation algorithm generated these error variances on a* 1/6*-degree grid.*

area, provided that the process is stationary (as it is in this case). Figure 7.6 shows the error variances for estimating sea surface temperature given all 42,298 measurements in Figure 7.5. A selective orthogonalization scheme was used to generate this result [18, Appendix B]. Although the number of iterations is growing with problem size, the number of iterations needed for this moderately large 320,400-dimensional estimation problem is 249. That only a relatively small number of iterations was used indicates that the algorithm has found a very low-rank but very good estimator. Hence, the algorithm described here can be used to solve high-dimensional, practical problems with relatively few iterations.

8. Conclusion. In this paper, a statistical interpretation of CG has been used to derive a Krylov subspace estimation algorithm. The algorithm computes a low-rank approximation to the linear least-squares error reduction term which can be used to recursively compute LLSEs *and* error variances. An analysis of the convergence properties explains behaviors of the algorithm. In particular, convergence is more rapid at measurement locations than away from them when there are scattered point measurements. Furthermore, the analysis indicates that a randomly generated vector is a good starting vector. The theory also suggests preconditioning methods for accelerating convergence. Preconditioning has been found to increase the rate of convergence in those cases where convergence is not already rapid.

The low-rank approximation to the error reduction term is a very useful statistical object. The computation of estimates and error variances is just one application. Another is the simulation of Gaussian random processes. Simulation typically requires the computation of the square root of the covariance matrix of the process, a potentially costly procedure. However, the Krylov subspace estimation algorithm can be adapted to generate a low-rank approximation to the square root of the covariance matrix. Yet another application is the fusion of existing estimates with those generated by additional data. The resulting fusion algorithm can also be used as the engine of a Kalman filtering routine, thereby allowing the computation of estimates of quantities evolving in time. This is the subject of ongoing research.

Acknowledgments. The authors wish to thank Jacob White, Gilbert Strang, and Hamid Krim for helpful discussions, Paul Fieguth for his insight and for providing the sea surface temperature data, and the reviewers for their thoughtful suggestions.

REFERENCES

[1] A. F. Bennett, *Inverse Methods and Data Assimilation*, Lecture notes from the summer school at the College of Oceanic and Atmospheric Sciences, Oregon State University, Corvallis, OR, 1999.

[2] A. F. Bennett, B. S. Chua, and L. M. Leslie, *Generalized inversion of a global numerical weather prediction model*, Meteorology Atmospheric Phys., 60 (1996), pp. 165–178.

[3] A. F. Bennett, B. S. Chua, and L. M. Leslie, *Generalized inversion of a global numerical weather prediction model* II: *Analysis and implementation*, Meteorology Atmospheric Phys., 62 (1997), pp. 129–140.

[4] A. da Silva and J. Guo, *Documentation of the Physical-space Statistical Analysis System (PSAS) Part* I: *The Conjugate Gradient Solver Version PSAS*-1.00, DAO Note 96-02, Data Assimilation Office, Goddard Laboratory for Atmospheres, NASA, 1996; also available online from ftp://dao.gsfc.nasa.gov/pub/office_notes/on9602.ps.Z.

[5] J. W. Demmel, *Applied Numerical Linear Algebra*, SIAM, Philadelphia, 1997.

[6] R. M. Dudley, *Real Analysis and Probability*, Chapman and Hall, New York, 1989.

[7] P. W. Fieguth, M. R. Allen, and M. J. Murray, *Hierarchical methods for global-scale estimation problems*, in Proceedings of the IEEE Canadian Conference on Electrical and Computer Engineering, IEEE, New York, NY, 1998, pp. 161–164.

[8] G. Golub and C. V. Loan, *Matrix Computations*, Johns Hopkins University Press, Baltimore, MD, 1996.

[9] R. A. Horn and C. R. Johnson, *Matrix Analysis*, Cambridge University Press, Cambridge, UK, 1985.

[10] M. R. Leadbetter, G. Lindgren, and H. Rootzen, *Extremes and Related Properties of Random Sequences and Processes*, Springer-Verlag, New York, 1983.

[11] D. G. Luenberger, *Optimization by Vector Space Methods*, John Wiley and Sons, New York, 1969.

[12] S. Mallat, G. Papanicolau, and Z. Zhang, *Adaptive covariance estimation of locally stationary processes*, Ann. Statist., 26 (1998), pp. 1–47.

[13] C. C. Paige and M. A. Saunders, *LSQR: An algorithm for sparse linear equations and sparse least squares*, ACM Trans. Math. Software, 8 (1982), pp. 43–71.

[14] B. N. Parlett, *The Symmetric Eigenvalue Problem*, Prentice-Hall, Englewood Cliffs, NJ, 1980.

[15] B. N. Parlett and D. S. Scott, *The Lanczos algorithm with selective orthogonalization*, Math. Comp., 33 (1979), pp. 217–238.

[16] Y. Saad, *On the rates of convergence of the Lanczos and the block-Lanczos method*, SIAM J. Numer. Anal., 17 (1980), pp. 687–706.

[17] Y. Saad, *Numerical Methods for Large Eigenvalue Problems*, Manchester University Press, Manchester, UK, 1992.

[18] M. K. Schneider, *Krylov Subspace Estimation*, Ph.D. thesis, MIT, Cambridge, MA, 2001.

[19] G. Xu, Y. Cho, and T. Kailath, *Application of fast subspace decomposition to signal processing and communication problems*, IEEE Trans. Signal Process., 42 (1994), pp. 1453–1461.

[20] G. Xu and T. Kailath, *Fast estimation of principal eigenspace using Lanczos algorithm*, SIAM J. Matrix Anal. Appl., 15 (1994), pp. 974–994.

[21] G. Xu and T. Kailath, *Fast subspace decomposition*, IEEE Trans. Signal Process., 42 (1994), pp. 539–551.

[22] G. Xu, H. Zha, G. Golub, and T. Kailath, *Fast algorithms for updating signal subspaces*, IEEE Trans. Circuits Systems II Analog Digital Signal Process., 41 (1994), pp. 537–549.

SIAM J. SCI. COMPUT.
Vol. 22, No. 5, pp. 1865–1884
© 2001 Society for Industrial and Applied Mathematics

ON THE APPROXIMATION OF OPTIMAL STOPPING PROBLEMS WITH APPLICATION TO FINANCIAL MATHEMATICS*

MICHAEL D. MARCOZZI†

Abstract. The determination of the value function associated with a given reward and stochastic process represents an important class of stochastic control problem. In particular, the expectations of such processes may be represented as solutions of variational inequalities of evolutionary type typically characterized by their high number of degrees of freedom, unbounded domains, and lack of "natural" boundary conditions. In this paper, we introduce two methodologies for computing the value function of optimal stopping associated with general stochastic processes. Our results are implemented utilizing finite elements and are validated using problems taken from financial mathematics.

Key words. optimal stopping, variational inequalities, mathematical finance, option pricing

AMS subject classifications. 60G40, 35K65, 91B28

PII. S1064827599364647

1. Introduction.

We consider in this paper the outcome of making a decision based upon present information pertaining to a random process X_t relative to which there exist two possibilities at each moment of time $t > 0$: exit the process or continue on. We remark that the two decisions are generally not equally favorable but depend on efficiencies represented by a given reward function $g(\xi) \geq 0$. The issue is then to decide whether future gains will outweigh the loss due to stopping at the present moment or due to future unfavorable moves. Since the process on which these decisions are based is random, it is not possible to know at the time if the decision will turn out to be optimal. We therefore consider determining a stopping strategy which provides the best result "in the long run"; that is, which optimizes the expectation relative to the reward function. Such a scenario constitutes an optimal stopping problem. More specifically, if X_t is an Itô diffusion on $\mathbb{R}^n$ starting at $X_0 = x \in \mathbb{R}^n$ and g is continuous, we seek a stopping time τ^* such that

$$\mathbb{E}\left[g(X_{\tau^*})\right] = \sup_{\tau} \mathbb{E}\left[g(X_\tau)\right]$$

and the corresponding optimal expected reward

$$g^*(x) = \mathbb{E}\left[g(X_{\tau^*})\right],$$

where the supremum is taken over all stopping times τ for $\{X_t\}$ and $\mathbb{E}$ denotes the expectation with respect to the probability law associated with the process X_t.

The determination of the value function associated with a given reward g and stochastic process X_t represents an important class of stochastic control problem. When early stopping is anticipated, the value function may be approximated either through dynamic programming algorithms or their associated variational inequality. Indeed, traditionally, dynamic programming algorithms (i.e., approximating Markov

*Received by the editors December 10, 1999; accepted for publication (in revised form) July 26, 2000; published electronically January 31, 2001.
http://www.siam.org/journals/sisc/22-5/36464.html
†Department of Mathematical Sciences, University of Nevada, Las Vegas, NV 89154-4020 (marcozzi@nevada.edu).

chains, lattice techniques) have been developed by formally applying an explicit first-order accurate in time, second-order accurate in space finite difference approximation scheme to the Feynman–Kac partial differential equation. For single sources of uncertainty, dynamic programming has remained popular due more to its pedagogical simplicity than its computational efficiency.

The main difficulty toward implementing dynamic programming algorithms for multiple noise processes has largely stemmed from their inherent computational inefficiencies. That is, the discretization of the diffusion generator produces stiff systems which are more efficiently handled through implicit methods, possibly utilizing adaptive mesh refinement and time stepping techniques. However, precisely because they are based on explicit time stepping strategies, dynamic programming algorithms enjoy the feature that boundary information may be inherently propagated from the initial condition, thereby making any explicit statement of boundary conditions redundant, at the cost of possibly excessively large mesh sizes (a problem particularly acute for processes in higher dimensions). This has led, in particular, to the naïve belief that general so-called field equation methods (e.g., finite difference, finite element) are unsuitable for stopping problems. Indeed, while a few recent attempts have been made to apply field equation methods to some stopping problems involving two sources of uncertainty, these attempts have led exclusively to formulations which are neither well-posed in a mathematical sense nor may they be generalized.

The fundamental issue in applying field equation methods for computing the value function associated with a stopping process lies in the development of appropriate computationally viable formulations. To this end, we show that one poses restrictions of the original problem on a sequence of exhausting approximating domains such that boundary conditions are inferred from the reward function. Using these calculations as a benchmark, this paper also introduces an entirely new *implicit* formulation wherein boundary conditions are *not* employed. In particular, we note that this new "natural (no) boundary condition" formulation has the potential to reduce the size of the computational domain (thereby reducing the associate size of the discretization) by eliminating error introduced by the artificial boundary condition, a feature particularly important for multiple noise processes.

As an application, we consider stopping problems derived from financial mathematics. To this end, we note that a vanilla American option is a contract to buy or sell a prescribed asset (which follows a diffusion process) for a predetermined amount (the exercise price) up until a specified time in the future (the expiration date). The purpose of option pricing is to determine the present value of the option and its (expected) optimal exercise time. Options which may only be exercised at the expiration date are known as European. As represented by Black and Scholes [9] and Merton [42], the value of an option formally equates to the value of a portfolio consisting of a position in a safe asset, typically a money-market account, and the risky assets on which the option is written. The determination of the investment distribution which eliminates arbitrage opportunities in a risk-neutral measure is a stochastic control (stopping) problem and dictates the value of the portfolio and consequently the option.

The outline of this paper is as follows: in section 2, we detail the mathematical formulation of the stopping problem and the representation of the (deterministic) value function as the solution to a variational inequality. The interpretation of the mathematical framework in a financial context is presented. We introduce in section 3 two new methodologies for approximating the associated value functions. Estimates

are presented for a backward Euler finite element scheme which quantify the quality of the approximations. Finally, section 4 validates each method's performance for a number of problems arising in finance. To this end, we consider valuing options written on multiple (up to three) assets as well as an option written on a foreign currency acting within a stochastic interest rate global economy.

2. Stochastic equation; definition of the economy. We consider in section 2.1 the mathematical formulation of the stopping problem and the representation of its (deterministic) value function as the solution to a variational inequality. In section 2.2, we present a brief discussion of the mathematical framework vis-à-vis its financial context. Our intent here is to relate the assumptions necessary for the well-posedness of the variational formulation to those required for there to be unique existence of the option valuation problem.

2.1. Variational characterization of the stopping problem. In the following, we consider m independent exogenous sources of uncertainty represented in the model by the m-dimensional Brownian motion $\{B_t\}_{t\geq 0}$ defined on the probability space $(\Omega, \mathcal{F}, \mathbb{P})$. Here we assume that $\mathcal{F} = \{\mathcal{F}_t\}_{t\geq 0}$ denotes the standard augmentation of the natural filtration associated with $\{B_t\}_{t\geq 0}$. We suppose that the state variable $X_t \in \mathbb{R}^n$ evolves according to the Itô process

$$dX_s = b(X_s, s)\, ds + \sigma(X_s, s)\, dB_s, \quad s > t\,, \tag{2.1a}$$

$$X_t = x\,, \tag{2.1b}$$

such that x is deterministic. We make the following additional assumptions.

(**A1**) The drift vector $b : \mathbb{R}^n \times [0,\infty) \to \mathbb{R}^n$ is C^1 and has bounded derivatives.

(**A2**) The dispersion matrix $\sigma : \mathbb{R}^n \times [0,\infty) \to \mathbb{R}^{n\times m}$ is $C^{2,1}$, bounded, and has bounded derivatives.

(**A3**) The coercivity condition holds; that is, there exists a constant $\eta > 0$ such that

$$\xi^T\, a\, \xi \geq \eta\, \|\xi\|^2$$

for all $\xi \in \mathbb{R}^m$ and each $(x,t) \in \mathbb{R}^n \times (0,\infty)$. The $(n\times n)$-matrix $a(x,t) := \sigma(x,t)\,\sigma(x,t)^T$ is known as the diffusion matrix.

In particular, the functions b and σ satisfy the global Lipschitz condition

$$\|b(x,t) - b(y,t)\| + \|\sigma(x,t) - \sigma(y,t)\| \leq c\, \|x - y\|$$

for every $0 \leq t < \infty$, $x \in \mathbb{R}^n$, $y \in \mathbb{R}^n$, and for some constant $c > 0$, which then suffices to ensure that there exists a unique t-continuous strong solution of (2.1) (e.g., [35]). Formally, the characteristic operator $\mathcal{A}$ associated with (2.1) is given by

$$\mathcal{A}f := -\frac{1}{2}\sum_{i,j} \frac{\partial}{\partial x_i}\left(a_{ij}\frac{\partial f}{\partial x_j}\right) + \sum_j a_j \frac{\partial f}{\partial x_j}\,, \tag{2.2a}$$

where

$$a_j = \sum_i \frac{\partial a_{ij}}{\partial x_i} - b_j\ ;\ j = 1, \ldots, n\,. \tag{2.2b}$$

Associated with the flow (2.1), the decision variable (i.e., stopping time) θ, and the constraint ψ, the (expected) cost function is defined as

$$J_t^x(\theta) = \mathbb{E}_{\mathbb{P}}\left[\exp\left(-\int_t^\theta a_0(X_s, s)\, ds\right)\psi(X_\theta, \theta)\right] \tag{2.3a}$$

and the stopping problem is defined as

$$u(x,t) = \sup_{\theta \in \mathcal{T}_{[t,T]}} J_t^x(\theta)\,, \tag{2.3b}$$

where $T < \infty$ represents the (time) horizon and $\mathcal{T}_{[t,T]}$ the set of stopping times in $[t,T]$. The cost (or reward) function associated with (2.3) is

$$\exp\left(-\int_t^\theta a_0(X_s, s)\, ds\right)\psi(X_\theta, \theta),$$

where the exponential term may be interpreted as an actualization of the (final) reward.

In order to characterize the deterministic valuation function (2.3) as the solution of a variational inequality, we introduce the weighted Sobolev spaces $W^{m,p,\mu}$; the space of functions $u \in L^p(\mathbb{R}^n, e^{-\mu|x|}\,dx)$ whose weak derivatives of all orders $\leq m$ belong to $L^p(\mathbb{R}^n, e^{-\mu|x|}\,dx)$, where m denotes a nonnegative integer, $1 \leq p \leq \infty$, and $0 < \mu < \infty$. We equip $W^{m,p,\mu}$ with the norm

$$\|u\|_{m,p,\mu} = \left\{\sum_{k \leq m} \int_{\mathbb{R}^n} |D^k u(x)|^p \cdot e^{-\mu|x|}\, dx\right\}^{1/p}.$$

If $\mathcal{X}$ is equipped with norm $\|\cdot\|_{\mathcal{X}}$, the space $L^p([0,T];\mathcal{X})$ consists of the set of measurable functions $g : [0,T] \to \mathcal{X}$ such that $\int_{[0,T]} \|g(t)\|_{\mathcal{X}}^p\, dt < \infty$. Subject to the additional assumption

(**A4**) the constraint $\psi \in L^p([0,T]; W^{2,p,\mu})$, where $p > (n/2) + 1$, is nonnegative and bounded,

we consider the following (strong) variational inequality. Determine $u \in L^2([0,T]; H^1_{loc})$ $\cap\, L^p([0,T]; W^{2,p,\mu})$, $\partial u/\partial t \in L^2([0.T]; L^2_{loc}) \cap L^p([0,T]; W^{0,p,\mu})$ such that

$$\frac{\partial u}{\partial t} - \mathcal{A}\, u - a_0 u \geq 0, \qquad u \geq \psi\,, \tag{2.4a}$$

$$\left\{\frac{\partial u}{\partial t} - \mathcal{A}\, u - a_0 u\right\} \cdot \{u - \psi\} = 0\,, \tag{2.4b}$$

where

$$u(x,T) = \psi(x,T) \tag{2.4c}$$

for all $x \in \mathbb{R}^n$. Supposing (A1)–(A4), that a_0 satisfies (A1), and that $\partial\psi/\partial t - \mathcal{A}\psi - a_0\psi \in L^p([0,T]; W^{0,p,\mu})$, there exists a unique solution to (2.4) represented by (2.3) for all p sufficiently large and μ sufficiently small (cf. [8] for the case $a_0 \in \mathbb{R}$; see also [32], [38], [24]). In particular, it follows from the Sobolev embedding theorem

that $u(\cdot,t) \in C^0$ for all $t \in [0,T)$. For $n = 1$, the solution u may be seen to be infinitely smooth on the continuation region and C^1 globally (cf. [48]). The current assumptions are not strict. We note that the boundedness imposed in (A4) is a formal restriction and may be circumvented by the process indicated in the following paragraph. Subsequently, all examples in what follows may be considered to lie within the present framework.

Practically, the solution u may be realized as the limit of a (pointwise) convergent sequence obtained through approximating (2.4) on bounded exhausting domains. That is, let $\{\Omega_k\}$ denote an increasing sequence of bounded open domains for which $\cup \Omega_k = \mathbb{R}^n$ and $\Psi \in L^p([0,T]; W^{2,p,\mu})$ a regularization of ψ such that $\widetilde{\psi} := \psi - \Psi \to 0$ as $\|x\| \to \infty$. Of course, one may consider as a special case that $\Psi = \psi$, but this is not required and consequently $\widetilde{\psi}$ does not necessarily vanish on $\partial\Omega_k$. We consider $\widetilde{u}_k := u_k - \Psi$, where u_k approximates $u - \psi$ on Ω_k and $\widetilde{u}_k|_{\partial\Omega_k} \equiv \widetilde{\psi}|_{\partial\Omega_k}$, by construction. In particular, there exists a unique $\widetilde{u}_k \in L^2([0,T]; H^1_{\widetilde{\psi}}(\Omega_k) \cap H^2(\Omega_k))$, $\partial\widetilde{u}_k/\partial t \in L^2([0,T]; L^2(\Omega_k))$ such that

$$\frac{\partial \widetilde{u}_k}{\partial t} - \mathcal{A}\,\widetilde{u}_k - a_0 \widetilde{u}_k \geq \widetilde{f} \ \text{ almost everywhere (a.e.) on } \ \Omega_k \times [0,T)\ ;\ \widetilde{u}_k \geq \widetilde{\psi} \ \text{ on } \ \overline{\Omega}_k \times [0,T]\,, \tag{2.5a}$$

$$\left\{ \frac{\partial \widetilde{u}_k}{\partial t} - \mathcal{A}\,\widetilde{u}_k - a_0 \widetilde{u}_k - \widetilde{f} \right\} \cdot \left\{ \widetilde{u}_k - \widetilde{\psi} \right\} = 0 \ \text{ a.e. on } \ \Omega_k \times [0,T)\,, \tag{2.5b}$$

where $\widetilde{f} = -\partial\widetilde{\psi}/\partial\, t + \mathcal{A}\widetilde{\psi} + a_0\widetilde{\psi}$ and

$$\widetilde{u}_k(x,T) = \widetilde{\psi}(x,T) \tag{2.5c}$$

for all $x \in \Omega_k$, where $H^1_{\widetilde{\psi}}(\Omega_k) := \{v \in H^1(\Omega_k) \mid v = \widetilde{\psi} \text{ on } \partial\Omega_k\ \}$. It follows then for any compact set $\overline{G} \subset \mathbb{R}^n$ that

$$\max_{t\in[0,T]} \|u(x,t) - u_k(x,t)\|_{L^\infty(\overline{G})} \to 0 \ \text{ as } \ k \to \infty \tag{2.6}$$

(cf. [8], proof of Theorem 3.19 and section 3.4.9 or [32], Proposition 4.1 in the case $\operatorname{supp}\{\widetilde{\psi}\} \subseteq \Omega_k$ for k sufficiently large). The result (2.6) may be intuitively understood in that the asymptotic behavior (that is, $\widetilde{u}_k|_{\partial\Omega_k}$) cannot "appreciably" affect the solution in any fixed bounded region within a finite interval of time (n.b. the behavior of the Green's function; cf. also [23]).

Remark. It is precisely this result which justifies the essentially arbitrarily prescribed boundary information associated with the approximation (2.5) in use exclusively throughout the literature for multidimensional state spaces ($n > 1$) and single dimensional state spaces ($n = 1$) when the asymptotic behavior of the value function is not readily apparent. That is, generalizing the above, we see that *any* well-posed problem on Ω_k yields the estimate (2.6) and so suffices as an approximation to (2.4) *provided* that k is taken sufficiently large and the validity of the approximation is considered on the compact region $\overline{G}$ only. Indeed, in Barles, Daher, and Romano [5] the influence of the artificial boundary data $\widetilde{u}_k|_{\partial\Omega_k}$, which is noted to be local near $\partial\Omega_k$, is attributed to "boundary-layer" effects, presupposing an advection dominated region, which is not the case. This effect is, in fact, the natural consequence of the restriction of the problem from $\mathbb{R}^n$ to Ω_k and is essentially a purely local influence confined to a neighborhood of $\partial\Omega_k$.

The starting point for most applications typically assumes that the problem is posed on a bounded domain. For example, in Clarke and Parrot [15], Dirichlet data is utilized along one axis and Neumann data (set to vanish) along a second. Barles, Daher, and Romano [5] examine both Dirichlet and Neumann data for a single state variable application. Generally, the ad hoc nature of the boundary information causes the ill-posedness of the formulation such as in the studies by Zvan, Forsyth, and Vetzal [51] and Haber, Schönbucher, and Wilmott [28] involving two state variables and the one-dimensional studies of Dewynne and Wilmott [21] and Das [19]. In a footnote in Amin and Bordurtha [1] (referring to a problem to be considered in section 4 and justifying the need for a lattice type implementation which does not employ explicit boundary information), it is stated that "even if such a partial differential equation can be derived, the boundary conditions are extremely complicated with three state variables. In fact, no attempts have been made to numerically solve three state partial differential equations in the finance literature." We remark that all of the above cited studies may be formulated within the present unifying framework.

2.2. Economic context. In an economic context, we consider $(n-1)$ risky processes X_t within a given market; a_0 representing the rate of return from the nth asset which is supposed "risk-free," such as a U.S. government bond or money-market account, ψ a contract which specifies a payoff at the exercise time, and T the expiration date of the contract. The purpose of option pricing is to determine the "fair" value (and expected exercise time) of the contingent claim (cf. [3], [43], [34], [7]). Of course, the ability to value a claim lies within the assumptions imposed upon the economy. In particular, we require (cf. [6]) the following conditions.

- *Equivalent martingale measure.* An equivalent martingale measure is a measure $\mathbb{Q}$ equivalent to $\mathbb{P}$, under which the discounted securities (that is, normalized by one of the asset within the market) are all $\mathbb{Q}$-martingales.
- *Arbitrage-free.* A market is arbitrage-free if there exists no "wealth" process (self-financing trading strategy) having zero initial value which generates a "risk-free" profit (that is, a positive expectation under the $\mathbb{P}$ equivalent "risk-free" measure $\mathbb{Q}$).
- *Completeness.* A market is complete if any possible derivative claim can be "hedged" by trading with a "self-financing" portfolio of securities (that is, the value process lies within the span of the securities).

The ability to value options is predicated on the following result (cf. [29], [20]):

1. A market is arbitrage-free if and only if there exists an equivalent martingale measure $\mathbb{Q}$.
2. A market is complete if and only if it is arbitrage-free and the measure $\mathbb{Q}$ is unique.

From a mathematical perspective, there exists a unique value for a derivative security if and only if there exists a unique equivalent martingale measure $\mathbb{Q}$. In order to represent the unique "fair" value of the derivative security, the expectation (2.3) is necessarily considered with respect to the $\mathbb{P}$-equivalent measure $\mathbb{Q}$. In a financial context, a portfolio consisting of positions in the $n-1$ risky assets and the riskless security may be constructed to "replicate" (hedge) the behavior of the derivative security (the value process) in such a manner that an investor cannot secure a profit starting out with no investment in the market. Put another way, an investor cannot make a profit without incurring some risk (this means that the market is arbitrage-free). Martingales are necessary for there to be no arbitrage and hedging results in unique prices.

With respect to the above, we note that coercivity (A3) and the regularity assumptions (A1), (A2), and (A4) provide for the unique existence of an equivalent martingale measure through a Girsanov transformation and consequently for market completeness. These same arguments are essential for the well-posedness of (2.4) and consequently allow for the valuation of options through variational techniques.

In a complete market it is possible to define the so-called market price of risk γ_t as the change in drift associated with Girsanov's transformation from the measure $\mathbb{P}$ to $\mathbb{Q}$. In general, γ_t is dependent only on the traded assets X_t and time t, not on the underlying derivative security valuation u. As such, under the $\mathbb{Q}$-measure, an asset is tradable if and only if its market price of risk is zero. In this case, all (normalized) tradable assets in the economy have the same growth rate as the money-market account (and consequently it is not possible to generate a risk-free profit). It is for this reason that the $\mathbb{Q}$-measure is said to be risk-neutral or arbitrage-free. Financially, γ_t represents the return (per unit of risk) above the risk-free rate; an investor expects to be compensated for the uncertainty associated with his investment. By specifying a particular measure, such as the one which minimizes inherent risk, variational techniques may be used as well in incomplete markets, although in this case there is in general no unique option valuation (cf. [36]).

3. Approximation of the stopping problem. Standard results for the approximation of variational problems on bounded domains are utilized to construct an approximating sequence for the stopping problem (2.4) on unbounded domains via the exhausting sequence of restrictions (2.5). We implement this procedure as method (i) below. Clearly, any convergence realized beyond the estimate (2.6) will rely on the quality of the artificial boundary conditions imposed upon the frontier of Ω_k. In order to avoid this difficulty, we introduce a so-called natural (or no) boundary condition approximation. This constitutes method (ii) below. In section 3.1 we introduce the weak formulation of (2.5) followed in section 3.2 by an implicit differencing in time, finite element approximation in space. A pointwise approximating sequence is constructed in section 3.3 for (2.4) on $\overline{G}$. Estimates are provided which quantify the quality of the approximation.

3.1. Weak formulation on bounded domains. We introduce now the (weak) variational form of (2.5). Formally, we consider a class of so-called *test* functions v such that $v \geq \widetilde{\psi}$. It follows then from (2.5a) that for a given t

$$\left\{ \frac{\partial \widetilde{u}_k}{\partial t} - \mathcal{A}\,\widetilde{u}_k - a_0 \widetilde{u}_k - \widetilde{f} \right\} \cdot \left\{ v - \widetilde{\psi} \right\} \geq 0\,. \tag{3.1a}$$

Subtracting (2.5b) from (3.1a), we obtain

$$\left\{ \frac{\partial \widetilde{u}_k}{\partial t} - \mathcal{A}\,\widetilde{u}_k - a_0 \widetilde{u}_k - \widetilde{f} \right\} \cdot \{ v - \widetilde{u}_k \} \geq 0\,. \tag{3.1b}$$

The weak form of (2.5) then follows by integrating (3.1b) over Ω_k and applying the Green's formula in the usual manner (e.g., [8], [38], [49]). For simplicity, we suppose $a_{ij} \in \mathbb{R}$, then for each $t \in (0,T)$, $u, v \in H^1(\Omega_k)$, we define

$$a_k(t;u,v) := \frac{1}{2} \sum \int_{\Omega_k} a_{ij}\, \frac{\partial u}{\partial x_j}\, \frac{\partial v}{\partial x_i}\, dx + \sum \int_{\Omega_k} a_j\, \frac{\partial u}{\partial x_j}\, v\, dx + \int_{\Omega_k} a_0\, u\, v\, dx$$

and

$$b_k(t;u,v) := \sum \int_{\partial \Omega_k} a_{ij}\, \frac{\partial u}{\partial x_j}\, v\, n_i\, dx\,,$$

where n_i denotes the ith component of the outward unit normal vector to $\partial\Omega_k$. In what follows we denote the $L^2(\Omega_k)$ inner product by $(\cdot,\cdot)$ and recall that $H^{2,1}(\Omega_k \times (0,T)) := \{v \mid v_{x_i}, v_{x_{ij}}, v_t \in L^2\}$. The two formulations below differ solely in the manner in which boundary information is inherited from (2.4) by (2.5).

(i) *Constraint as boundary condition.* We consider here that $\operatorname{supp}\{\widetilde{\psi}\} \subseteq \Omega_k$ (in which case $\widetilde{u}_k|_{\partial\Omega_K} = 0$) for all k sufficiently large. In this case, the solution $\widetilde{u}_k \in \overset{\circ}{K}_t := \{v \in H^{2,1}(\Omega_k \times (0,T)) \,|\, v \geq \widetilde{\psi} \text{ a.e. in } \Omega_k \times (0,T);\ v = 0 \text{ a.e. on } \partial\Omega_k \times (0,T)\}$ of (2.5) is equivalently a solution to the evolutionary variational inequality

$$\left(\frac{\partial \widetilde{u}_k}{\partial t}, v\right) - a_k(t; \widetilde{u}_k, v) \geq \left(\widetilde{f}, v\right) \quad \text{a.e. in } t\,, \tag{3.2a}$$

such that

$$\widetilde{u}_k(x,T) = \widetilde{\psi}(x,T) \ \text{ on } \ \Omega_k \tag{3.2b}$$

for all $v \in \overset{\circ}{K}_t$. The equation (3.2) is the statement of (2.5) in weak form. The application of $\widetilde{u}_k|_{\partial\Omega_k} = 0$ represents a so-called (essential) artificial boundary condition for (3.2) or (2.5) relative to (2.4). We remark that the quality of the regularization Ψ affects the efficiency of this implementation through the Dirichlet condition.

(ii) *Natural (no) boundary condition.* Supposing that $\widetilde{u}_k \in K_t := \{v \in H^{2,1}(\Omega_k \times (0,T)) \,|\, v \geq \widetilde{\psi} \text{ a.e. in } \Omega_k \times (0,T)\}$ is a solution of (2.5), it is equivalently a solution of the variational inequality

$$\left(\frac{\partial \widetilde{u}_k}{\partial t}, v\right) - a_k(t; \widetilde{u}_k, v) - b_k(t; \widetilde{u}_k, v) \geq \left(\widetilde{f}, v\right) \quad \text{a.e. in } t\,, \tag{3.3a}$$

such that

$$\widetilde{u}_k(x,T) = \widetilde{\psi}(x,T) \ \text{ on } \ \Omega_k \tag{3.3b}$$

for all $v \in K_t$. Note that here the boundary conditions are formally represented by allowing $x \in \Omega_k \to x_o \in \partial\Omega_k$ such that $\partial u/\partial t - \mathcal{A}u - a_0 u - \widetilde{f} \to 0$; that is, by considering one-sided derivatives in $\mathcal{A}$ on $\partial\Omega_k$. Indeed, *no* explicit boundary information is transmitted by this condition.

We label this approach as "natural" in that boundary information is not imposed in the space of test functions K_t. Nor is boundary information transmitted through the variational equation (3.3a). In particular, we require only a growth constraint on the solution which is enforceable on the finite domain Ω_k, for all k sufficiently large, by the well-posedness of the formulation (3.3) alone. This "trivial (no)" boundary condition has important ramifications relative to the numerical implementation of the scheme (cf. section 4).

3.2. Discretization on bounded domains. When, for a given fixed k, the constraint is imposed as an artificial boundary condition in (2.5) (that is, when $\widetilde{u}_k|_{\partial\Omega_k} = 0$), numerical methods for the approximation of $\widetilde{u}_k$ are well known (cf. [27], [26], [14], [17]). As such, we now consider the approximation of the evolutionary variational inequality (2.5) by (implicit) semidiscretization. To this end, we let $\Delta t = T/M$, for some positive integer M, $t_m = m \cdot \Delta t$, and define the sequence $\{\widetilde{u}_k^m\}$, $\widetilde{u}_k^m \in V_k$, by recurrence starting with

$$\widetilde{u}_k^M = \widetilde{\psi}\,, \tag{3.4a}$$

where, for $m = 1, \ldots, M-1$,

$$-\left(\frac{1}{\Delta t}\widetilde{u}_k^m, v\right) - a_k(t_m; \widetilde{u}_k^m, v) - \iota \cdot b_k(t_m; \widetilde{u}_k^m, v) \geq \left(\widetilde{f} - \frac{1}{\Delta t}\widetilde{u}_k^{m+1}, v\right) \tag{3.4b}$$

for all $v \in V_k$, where $\iota = 0$, $V_k = \overset{\circ}{K} := \{v \in H^2(\Omega_k) \,|\, v \geq \widetilde{\psi} \text{ in } \Omega_k \,;\, v = 0 \text{ on } \partial\Omega_k\}$ and $\iota = 1$, $V_k = K := \{v \in H^2(\Omega_k) \,|\, v \geq \widetilde{\psi} \text{ in } \Omega_k\}$ when approximating (3.1) and (3.2), respectively. In particular, we note that for each $1 \leq m < M$, the relation (3.4b) is a stationary variational inequality which is known to be uniquely solvable in the case $V_k = \overset{\circ}{K}$, for all Δt sufficiently small (and, by transformation, for all Δt). Allowing piecewise continuation on the interval $[m\,\Delta t, (m+1)\,\Delta t)$ and imposing the constraint at the boundary, it can be shown in addition that the solution $\widetilde{u}_k^m$ converges weakly to $\widetilde{u}_k$ in $H^{2,1}(\Omega_k \times (0,T))$ (cf. [8], [26]).

In order to obtain a fully discrete approximation of (3.2) (resp., (3.3)), we let $\{S_h\}$ denote a family of finite-dimensional subspaces of $H_0^1(\Omega_k) \cap H^2(\Omega_k)$ (resp., $H^2(\Omega_k)$) such that $\{\phi_{h,1}, \phi_{h,2}, \ldots, \phi_{h,N}\}$ forms a basis for S_h. For definiteness, we suppose that Ω_k is rectangular and that S_h consists of piecewise linear (componentwise) functions on a quasi-uniform rectangular "triangulation" of Ω_k with mesh size (length of the longest side) h. Letting $\mathcal{N}_h$ denote the set of nodes of the space S_h (i.e., the set of all vertices), we define

$$K_h = \{v_h \in S_h \mid \text{ for all } b \in \mathcal{N}_h\,,\ v_h(b) \geq \widetilde{\psi}(b)\}\,.$$

In general, the set K_h is not contained in either $\overset{\circ}{K}$ or K. Replacing V_k with K_h, we obtain the following discretization in space and time of (3.4). Determine the sequence $\{\widetilde{u}_h^m\}$, $\widetilde{u}_h^m \in K_h$, by recurrence starting with

$$\widetilde{u}_h^M = \widetilde{\psi}\,, \tag{3.5a}$$

where, for $m = 1, \ldots, M-1$,

$$-\left(\frac{1}{\Delta t}\widetilde{u}_h^m, v\right) - a_k(t_m; \widetilde{u}_h^m, v) - \iota \cdot b_k(t_m; \widetilde{u}_h^m, v) \geq \left(\widetilde{f} - \frac{1}{\Delta t}\widetilde{u}_h^{m+1}, v\right) \tag{3.5b}$$

for all $v \in K_h$, where $\iota = 0$ (resp., $\iota = 1$) for the fully discrete approximation of (3.2) (resp., (3.3)). In particular, we have the representations

$$\widetilde{u}_h^m = \sum_{i=1}^{N} u_i(t) \cdot \phi_{h,i}(x) \tag{3.6a}$$

and

$$v = \sum_{i=1}^{N} v_i \cdot \phi_{h,i}(x)\,, \tag{3.6b}$$

where $u_i(t)$ and v_i are elements of $\mathbb{R}$. Substituting (3.6) into (3.5), we obtain the sequence (one for each m) of linear complimentary problems for $u^m = (u_1(t_m), u_2(t_m), \ldots, u_N(t_m))$ and $v = (v_1, v_2, \ldots, v_N)$: determine the sequence $\{u^m\}$ by recurrence such that

$$u^M = \widetilde{\psi}(b, t_M) \text{ for all } b \in \mathcal{N}_h, \tag{3.7a}$$

where

$$v \cdot A\, u^m \geq v \cdot F^{m+1} \tag{3.7b}$$

for $m = 1, \ldots, M-1$ (cf. [16], [39]). Here, $A = (A_{ij})$ such that

$$A_{ij} = -\frac{1}{\Delta t}(\phi_{h,i}, \phi_{h,j}) - (\nabla\phi_{h,i} \cdot e_i, \nabla\phi_{h,j} \cdot e_j) - \iota \cdot (\nabla\phi_{h,i} \cdot n_i, \phi_{h,j})\,,$$

$$F^{m+1} = \widetilde{f} - B\, u^{m+1}\,,$$

where $B = (B_{ij})$,

$$B_{ij} = \frac{1}{\Delta t}(\phi_{h,i}, \phi_{h,j})\,,$$

and e_i and n_i denote the ith component of the unit coordinate and unit outward normal vectors, respectively.

In particular, the sequence $\{\widetilde{u}_h^m\}$ constitutes an implicit Euler finite element approximation of $\widetilde{u}_k$ utilizing regular n-rectangles of type 1 (and globally of class C^0), where the rectangular partitions are hierarchical and consist of quasi-uniform elements parameterized by h (cf. [14], [27], [26]). The quality of the approximation (3.5) to (3.2), for fixed k, is known to satisfy the estimate

$$\max_{t\in[0,T]} \|\widetilde{u}_k - \widetilde{u}_h^m\|_{L^\infty(\Omega_k)} \leq \mathcal{O}(l + h^{2-\epsilon}) \tag{3.8}$$

for any $\epsilon > 0$ and all $l := \Delta t$, h sufficiently small (cf. [27]). The estimate (3.8) is a statement that the finite element discretization (3.5) is first-order accurate in time and second-order accurate in space over Ω_k.

3.3. Discrete approximation on unbounded domains. The approximating methods (i) and (ii) are then defined by considering the pointwise limit on $\overline{G}$ of the sequence $\widetilde{u}_h^m$ for $\iota = 0$ and 1, respectively, as $k \to \infty$. More precisely, for a given k, a finite element approximation $\widetilde{u}_{h_k}^m$ is constructed which satisfies (3.7) for a specific time/mesh size l_k and h_k. By induction, the approximation $\widetilde{u}_{h_{k+1}}^m$ is constructed with $\Omega_k \subset \Omega_{k+1}$, $l_k > l_{k+1}$, and $h_k > h_{k+1}$. Letting $u_{h_k}^m := \widetilde{u}_{h_k}^m|_{\overline{G}} + \Psi|_{\overline{G}}$, the *diagonalizing* sequence $\{u_{h_k}^m\}$ approximating the solution u of (2.4) is now fully specified on $\overline{G}$. Combining (3.8) with (2.6), it then follows that

$$\max_{t\in[0,T]} \|u(x,t) - u_{h_k}^m(x,t)\|_{L^\infty(\overline{G})} \leq \mathcal{O}(l_k + h_k^{2-\epsilon}) + o(\Pi_k^{-1}) \;\text{ as }\; k \to \infty \tag{3.9}$$

for any $\epsilon > 0$, where $\overline{G} \subset \Omega_k$ denotes the (fixed) so-called *approximation* domain and $\Pi_k := \mathrm{diam}\{\Omega_k\}$ is the diameter of the *computational* domain Ω_k. Clearly, the first term in the estimate bounds the time discretization error, the second bounds the space discretization error, and the third bounds the error due to the truncation of the domain. We note (cf. section 4) (a) that asymptotic performance may *only* be realized on the approximating region $\overline{G}$ and not over the entire computational domain Ω_k, (b) the futility of refining a mesh relative to a fixed computational domain, and (c) the detriment of stipulating a time/mesh sizing l_k and h_k without refining Π_k. The practices (a)–(c) are utilized exclusively in the literature (e.g., [25], [5]). Relative to method (ii), we remark that it is not known if (3.3) or its finite element approximation

(3.5) is solvable. However, we verify the estimate (3.9) in this case computationally in the next section.

Remark. We note that in Jaillet, Lamberton, and Lapeyre [32], Zhang [50], and Barles, Daher, and Romano [5], for example, the estimate (2.6) and a finite difference analogue of (3.8) relative to (3.2) are obtained. The estimate (3.9) addresses the more general case where l_k, $h_k \to 0$ *and* $\Pi_k \to \infty$. In particular, each of the cited works develops a "localization" of the expectation (cf. (2.3)) onto a bounded domain (that is, the diffusion is absorbed on $\partial\Omega_k$), the value function being then characterized as the solution to a variational problem such that $\widetilde{u}_k|_{\partial\Omega} = 0$ (for fixed k). This approach fails to develop the local approximation to u on $\overline{G}$ through the construction of the sequence $\{u^m_{h_k}\}$. In this context, the localization employed by method (ii) would be consistent with a specific reflected diffusion imposed on Ω_k.

4. Computational results. In order to validate the estimate (3.9), and consequently verify the convergence and efficiency of the proposed methods, we consider as an application a variety of optimal stopping problems derived from mathematical finance. To this end, we value options written on multiple assets (e.g., stocks) as well as an option written on a foreign currency. In section 4.1 we present the specifics of implementing the proposed methods. In section 4.2 we discuss case related details and summarize our general conclusion.

Since analytic solutions do not generally exist for optimal stopping problems, numerical methods are necessitated. In terms of approximating the value function of a stopping process, we remark that dynamic programming based algorithms have typically been employed, particularly in the single state space case when boundary information may be ignored (so-called lattice methods) without any great computational cost (cf. [25]). When the time horizon is known to be realized (e.g., European options), Monte Carlo and Quasi-Monte Carlo methods have been shown to be effective in higher dimensions (cf. [44], [10]). Recently, attempts at combining these two approaches have lead to dynamic programming simulation based techniques (cf. [12]). These appear attractive when possible early exit times are limited to a "small" set.

4.1. Numerical implementation. In the following, we solve the discretization (3.7) by projected successive-over-relaxation (SOR) (cf. [18]). Projection methods have been shown to be efficient solvers for (3.7) (cf. [11], [40], [30], [37]). An introductory treatment of finite elements and projected SOR can be found in [49], for example. Additionally, we note that for most numerical implementations it is not actually necessary to introduce the intermediate function $\widetilde{u}_k$ and the formulation (2.5). One simply constructs an approximation for $u|_{\Omega_k}$ directly; it is not necessary to introduce the regularization Ψ and consider that the Dirichlet data vanish on the computational boundary $\partial\Omega_k$. This is the procedure utilized unless otherwise noted. In the exceptional cases, we approximate $\widetilde{u}_k$ via (2.5) and determine $u|_{\Omega_k} = \widetilde{u}_k + \Psi$ assuming $\Psi = \psi$ and consequently that $u_k|_{\partial\Omega_k} = 0$ (cf. section 3.3).

Relative to the implementation of method (i) and the imposition of Dirichlet boundary data, the array A and vector F^{m+1} from (3.7) are constructed (assembled) and then the values of the discrete solution (overwrite) are set equal to the constraint Ψ at each boundary node. Method (ii) is implemented such that no overwriting occurs. In the single state variable case and a finite difference setting, method (ii) would be analogous to employing a second-order central differencing scheme for space discretization at the interior nodes and a second-order one-sided differencing scheme at the boundary nodes.

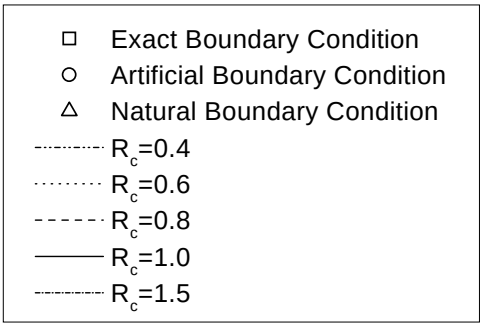


FIG. 1. *Figure index. Note: R_c is the radius of the computational domain in the maximum norm.*

When early exercise (stopping) is not anticipated (e.g., European options), comparisons will be made to the analytic solution where possible. In the case of early stopping (e.g., American options), or when analytic solutions are not available for processes which run to completion, we will compute solutions anticipated to be at least two orders of magnitude more accurate than the approximations and utilize these as benchmarks (the validity of this approach will be evidenced by the realized convergence rates) (cf. (III) below).

In order to computationally validate the estimate (3.9), we fix $l_k/h_k^2 = \beta$, where we choose for computational purposes $\beta = 1$, a sufficiently small constant (as verified by the asymptotic behavior obtained). As such, we expect to achieve asymptotic convergence as $k \to \infty$; that is, as $h_k \to 0$ and $\Pi_k \to \infty$. Note that the choice of β effects the relative performance (accuracy) of the method.

The approximation and computational domains were constructed as n-squares with (maximum norm) radius, R_a and R_c, respectively. The center of G and Ω_k were aligned. In particular, the radius $R_a = 0.1$ for all computations. Mesh spacing was uniform (that is, $\Delta x_i = \Delta x_j = \Delta x$) such that $h_k = \Delta x$.

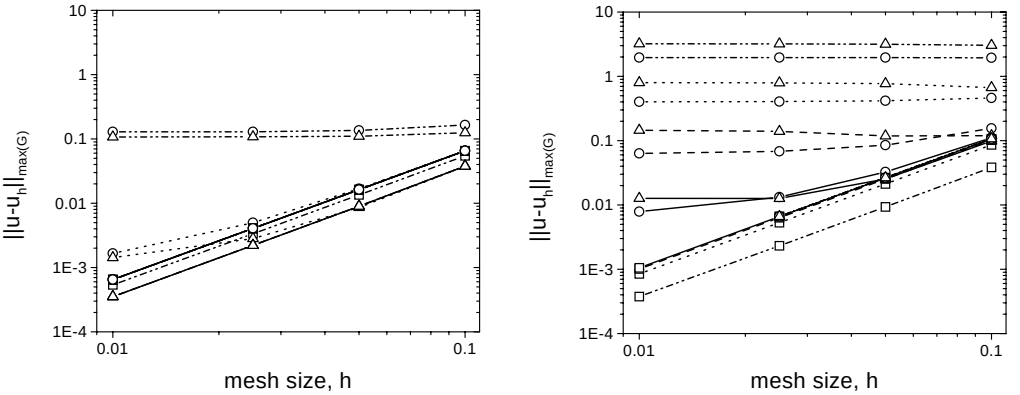
Computations were performed on a variety of platforms, the most powerful of which was a PIII 500MHz machine. The largest mesh size considered contained on the order of 36,000 nodes. The multivariate normal probabilities utilized for the multiple asset analytical solutions were computed using a double precision archived implementation of the code presented in [46].

4.2. Asymptotic analysis. The following formulations are provided in the so-called risk neutral measure $\mathbb{Q}$, designating complete markets and unique option prices (cf. section 2.2). We remark that the process (2.1) is fully specified by its drift and volatility, whereas the options themselves are uniquely determined by their payoff attributes represented by the function ψ in (2.3). Figure 1 provides the key which is utilized in all subsequent graphics.

(I) *European put option on the minimum of n-assets.* In this case, the logarithm of each stock's price is represented by a component of the diffusion X_t, the discounting factor a_0 and drifts b_j are taken to be the (fixed) risk-free rate of return $r \in \mathbb{R}$, and the payoff is defined such that

$$\psi(X_T, T) = \max\Big\{E - \min\left[e^{X_1}, e^{X_2}, \ldots, e^{X_n}\right], 0\Big\},$$

where E denotes the exercise price of the option. The constraint utilized for the



a. Single asset. *b. Two assets.*

FIG. 2. *European put option on the minimum of n assets.*

European options was the depreciated payoff

$$\psi(X_t, t) = \max\left\{E \cdot e^{-r\,(T-t)} - \min\left[e^{X_1}, e^{X_2}, \ldots, e^{X_n}\right], 0\right\}.$$

In the case of a single asset, the European put price is given by the celebrated Black–Scholes formula. For the two asset case, the Black–Scholes formula is generalized in Stultz [47]. The stochastic process (2.1) was defined such that $\sigma_{ii} = 0.3$, $\sigma_{ij} = 0$ $(i \neq j)$, $r = 0.1$, $T = 1.0$, while for the payoff we assumed $E = 100$. The center of the computational grid (and of G) was taken to be $\ln(E)$ in the single asset case and $(\ln(E), \ln(E))$ in the two asset case.

Figure 2 presents the convergence of methods (i) and (ii) on the approximating domain $\overline{G}$ for the one $(n = 1)$ and two $(n = 2)$ asset European cases. As European options do not exercise early, the variational inequality (2.5) reduces to a standard parabolic partial differential equation in this case. Computationally, when a European option is implemented as a variational inequality, the constraint would not be expected to enter into the projected SOR algorithm (thus, the problem would implicitly be solved as a parabolic partial differential equation). Indeed, we found that solving the equations via projected SOR, as opposed to SOR, resulted in slightly larger computed errors. The exception to this was the method (ii) formulation of the single asset put, which converged for significantly smaller computational domains in the variational setting (when projected SOR was employed), as opposed to a partial differential equation formulation (solved by SOR). The variational formulation (projected SOR) values are recorded in Figure 2 in the single asset case.

(II) *Call option on the minimum of n-assets.* As with the put, here we take the logarithm of each stock's price to be a component of the diffusion X_t, the discounting factor a_0 and drifts b_j again taken to be the risk-free rate of return r, and the payoff defined such that

$$\psi(X_T, T) = \max\Big\{\min\left[e^{X_1}, e^{X_2}, \ldots, e^{X_n}\right] - E, 0\Big\},$$

where E denotes the exercise price of the option. In this case, the time evolving constraint is the payoff

$$\psi(X_t, t) = \max\left\{\min\left[e^{X_1}, e^{X_2}, \ldots, e^{X_n}\right] - E, 0\right\}.$$

Calls, as defined here, do not exercise early. For a single asset, the sum of the European call plus the depreciated exercise price of the option equates to the European put price added to the asset's value; a principle known as put-call parity. For multiple assets, a variety of call options valuations are derived in Johnson [33] (cf. also [47]).

We found it necessary in the single asset case ($n = 1$) to run computations for the function $\widetilde{u}_k$ and to compute $u|_{\partial\Omega_k} = \widetilde{u}_k + \Psi$ (rather than calculate $u|_{\partial\Omega_k}$ directly, cf. section 4.1) in order to make method (ii) stable. We note that by (A4) we require the payoff function ψ to have bounded growth. Therefore, it is not surprising that we needed to compute $\widetilde{u}_k$ as per (2.5) rather than $u|_{\partial\Omega_k}$ directly. By contrast (and perhaps surprisingly), we were able to compute $u|_{\partial\Omega_k}$ directly for the multiple asset cases ($n = 2, 3$ trials), which afforded significant coding simplifications.

Figure 3 pertains to the convergence of the approximations on $\overline{G}$ to the analytic European call value for up to three assets. Here the stochastic process and payoff are defined by $\sigma_{ii} = 0.3$, $\sigma_{ij} = 0$ $(i \neq j)$, $r = 0.1$, $T = 1.0$, $E = 100$, and the computational grid centered about $\ln(E)$ in each component. As with the European put on a single asset, we utilized the variational inequality setting (i.e., projected SOR) only in the case of the call on a single asset for method (ii), convergence being obtained without the use of the constraint (that is, with SOR) for suitably large computational domains. The method (ii) implementation for calls on multiple assets ($n > 1$) was always stable (for any mesh and computational sizes) without the use of the constraint (the results represented in Figure 3). As with the European puts, utilizing the constraint for the European call options on multiple assets (that is, projected SOR as opposed to SOR) generally resulted in slightly larger computed errors.

(III) *Put option on the geometric average of n-assets.* As above, the logarithm of each stock's price is represented by a component of the diffusion X_t and the discounting factor a_0 and drifts b_j are taken to be the risk-free rate of return r. The payoff in this case is defined to be

$$\psi(X_T, T) = \max\left\{E - \left(e^{X_1} \cdot e^{X_2} \cdot \ldots \cdot e^{X_n}\right)^{1/n}, 0\right\},$$

the constraint for a European option is given by the depreciated payoff

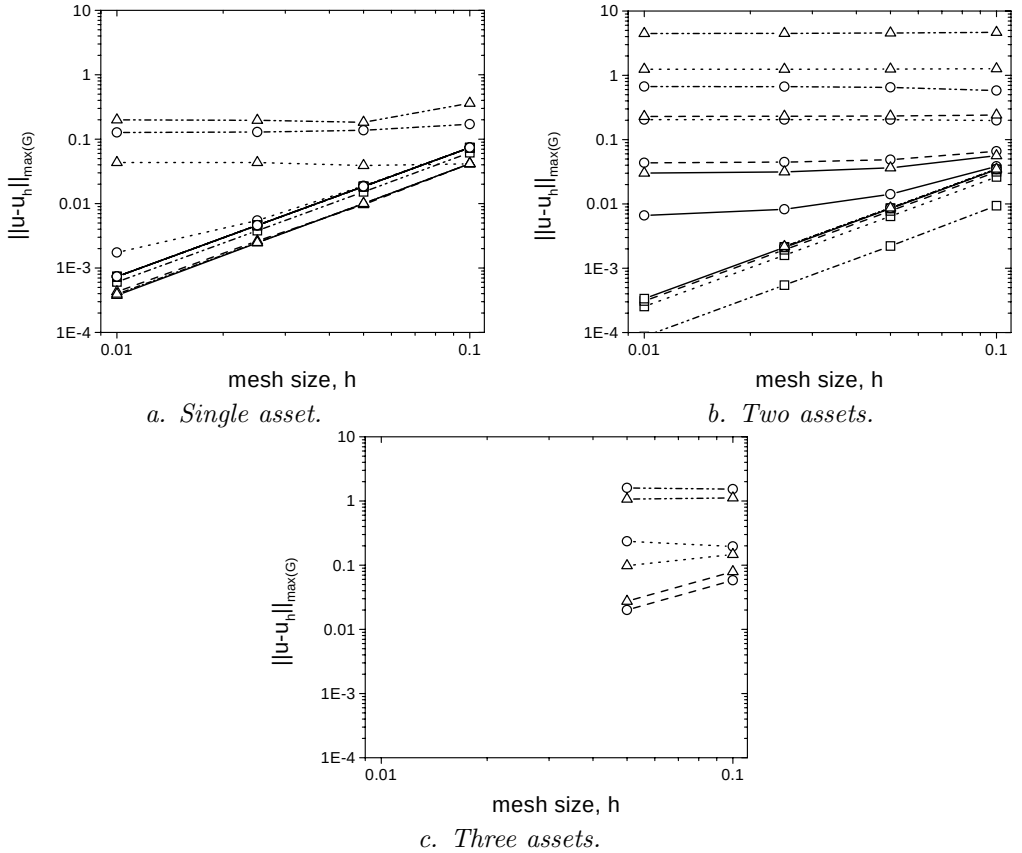
$$\psi(X_t, t) = \max\left\{E \cdot e^{-r\,(T-t)} - \left(e^{X_1} \cdot e^{X_2} \cdot \ldots \cdot e^{X_n}\right)^{1/n}, 0\right\},$$

while for an American option it is

$$\psi(X_t, t) = \max\left\{E - \left(e^{X_1} \cdot e^{X_2} \cdot \ldots \cdot e^{X_n}\right)^{1/n}, 0\right\},$$

where E again denotes the exercise price of the option. Unlike the previous cases, no analytic solution exists here for the European ($n > 1$) or American options (any $n > 0$). However, by introducing the process $Y_t = X_1 + X_2 + \cdots + X_n$ such that $\psi(Y_t) = \max\{E - e^{Y_t/n}, 0\}$, we are able to value these options by numerically approximating $4(Y_t)$, which has only a single source of uncertainty (that is, by solving the single state variable variational inequality for Y_t). The computed solutions for $4(Y_t)$ were then utilized as benchmarks for both the European and American cases when solved according to methods (i) and (ii) (with $h = 0.001$).

Figure 4 presents convergence results for the European case and Figure 5 the analogous results for the American case for up to three assets ($n \leq 3$), where $\sigma_{ii} = 0.3$, $\sigma_{ij} = 0$ $(i \neq j)$, $r = 0.1$, $T = 1.0$, and $E = 100$, and again the computational



a. Single asset. *b. Two assets.*

c. Three assets.

FIG. 3. *European call option on the minimum of n assets.*

grid was centered about $\ln(E)$ per coordinate axis. Note that the European put on the geometric average of a single asset is provided in Figure 2(a). We remark that the presence of the free boundary appears to upset the stability of the method (ii) implementation in the single asset American case for "small" computational domain sizes. (Again, for "larger" computational domain sizes, this was not a problem.) This occurrence was not present for the multiple asset ($n > 1$) tests.

(IV) *Option on a foreign currency.* Here, we denote the (stochastic) instantaneous risk-free rate of return at time t within the domestic economy by $a_0 = X_1$, the foreign economy by X_2, and the logarithm of the cross-currency exchange rate by X_3. It follows then that the payoff of a put on a foreign currency is given by

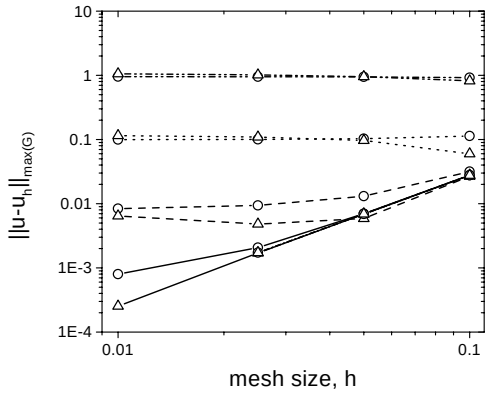
$$\psi(X_T, T) = \max\{E - e^{X_3}, 0\},$$

such that in the European case the time varying constraint is specified as

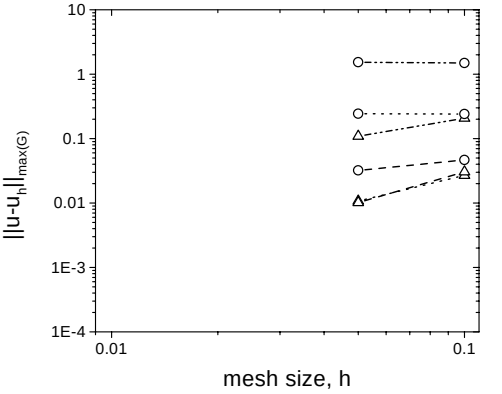
$$\psi(X_t, t) = \begin{cases} \max\{E - e^{X_3}, 0\} & \text{if } X_1 < 0, \\ \max\{E \cdot e^{-X_1 (T-t)} - e^{X_3}, 0\} & \text{if } X_1 \geq 0, \end{cases}$$

and by

$$\psi(X_t, t) = \max\{E - e^{X_3}, 0\},$$

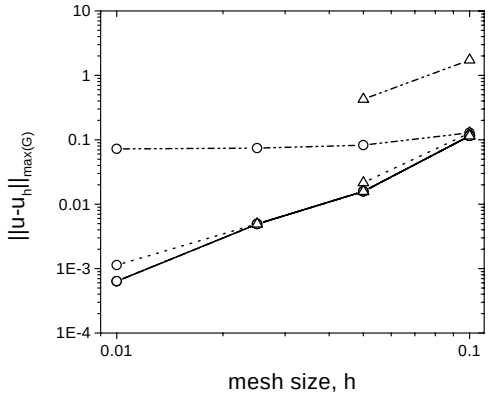


a. Two assets.

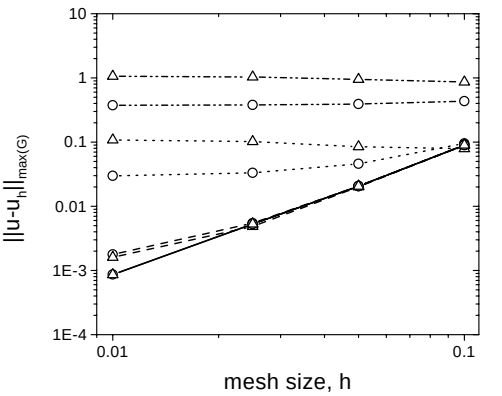


b. Three assets.

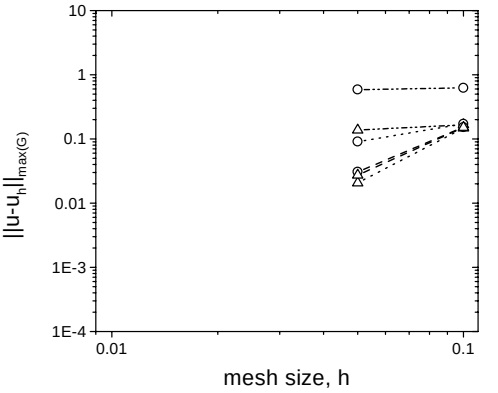
FIG. 4. *European put option on the geometric average of n assets.*



a. Single asset.



b. Two assets.



c. Three assets.

FIG. 5. *American put option on the geometric average of n assets.*

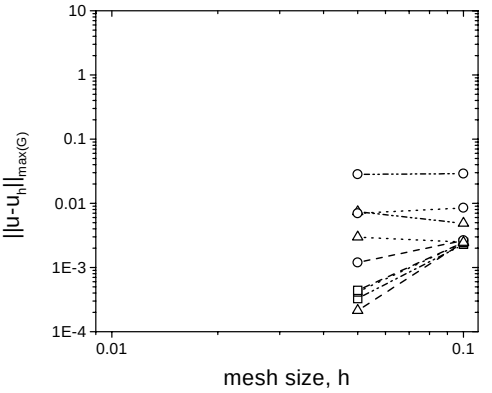


FIG. 6. *European put option on a foreign currency.*

in the American case, where E is the exercise price of the option. The specific model utilized is introduced in Choi and Marcozzi [13]. The general model for American options on a foreign currency under stochastic interests rates was first developed in Amin and Jarrow [2], which includes an analytic solution applicable to the European case specified here. The convergence results for the approximation schemes of section 3 are presented in Figure 6 for the European option. In particular, we suppose

$$\sigma = \begin{bmatrix} 0.05 & 0.005 & 0.005 \\ 0.005 & 0.05 & 0.005 \\ 0.03 & 0.03 & 0.3 \end{bmatrix}$$

in addition to taking $b_1 = 0.005$, $b_2 = 0.005 - \sum_{i=1}^{3} \sigma_{2i}\sigma_{3i}$, $b_3 = X_1 - X_2 - \sum_{i=1}^{3} \sigma_{3i}^2/2$, an exercise price of $E = 1$, and a duration of $T = 1$.

We note that for a computational mesh radius of $R_c = 0.8$ and a mesh size of $h = 0.05$, we obtained a computed solution utilizing method (i) of $u_k = 0.114450$ at $(0.05, 0.05, 0)$, the center of the computational domain. This compares to the analytic value of $u = 0.114477$, for a difference of 0.02%. We remark by way of comparison that the two-step discrete formulation in Amin and Bordurtha [1] was never demonstrated to have attained asymptotic convergence nor an accuracy greater than possibly 2–10% in spite of having employed a mesh of up to 2.6 million nodes (versus about 36,000 nodes for the present computations) and a graded time step (which significantly reduced the number of nodes). For the corresponding American option, we obtained a value of 0.117160, or only approximately 2% greater than for the European case.

We summarize the following points.

1. Method (ii) demonstrates the feasibility (and desirability) of solving evolutionary partial differential equations and variational inequalities on unbounded regions by implicit time discretization without the use of explicit boundary information. For European options written on a single asset ($n = 1$), we did, however, find it preferable to actually employ a variational inequality format and projected SOR when using method (ii).
2. The estimate (3.9) appears valid for method (ii) as well as method (i). In all tests, both methods converged at the anticipated asymptotic rate and realized comparable accuracies on the fixed approximation domain for suitably large computational domains. Conversely, for both methods (i) and (ii), and in

all cases considered, mesh refinement without suitably large computational domains resulted in no improvement of accuracy.

3. It appears that method (ii) becomes increasingly more efficient relative to method (i) as the number of state variables (the dimension n) increases; that is, method (ii) reaches its asymptotic limit with smaller computational domains. Indeed, in almost all cases tested involving three sources of uncertainty, method (ii) was able to obtain comparable accuracies to method (i) while simultaneously employing significantly fewer nodes in the mesh, the generality of this conclusion being predicated, however, on the particular choice of the constraint employed.
4. In the case of early stopping involving calculations employing a uniform mesh (or without an adaptive mesh strategy to locate the free boundary), the use of a smaller computational domain coupled with finer mesh sizing (relative to the nonearly stopping case) seems optimal. This may prove to be a necessary approach as adaptive mesh strategies for complicated free surfaces in dimensions greater than two may not be practical to implement. Conversely, when early stopping occurs, some form of adaptive meshing or local mesh refinement appears necessary in order to achieve asymptotic convergence rates and accuracies. We remark that this has consequences relative to the efficiency of projected solvers.

With the exception of dynamic programming (e.g., lattice) methods, which can be developed as a special case of variational techniques (cf. [23], [31]), there exist *no* alternative methodologies for determining the value function when continuous early exercise is allowed. Additionally, because lattice techniques rely on explicit time discretization and do not employ explicit boundary conditions, they are inherently less efficient than variational methods, particularly as greater accuracy is sought. With the exception of example (IV) above, we note that no existing benchmark computations exist at this time with which to compare the performance of the proposed methods. Ongoing research indicates that so-called *hp*-methods, featuring relatively large mesh sizing and adaptive time stepping (cf. [4], [45], [22]), and high performance computing appear to hold particular potential for multiple state variables applications. The present framework may also be extended to include path dependent processes (cf. [41]). Finally, this paper validates the asymptotic efficiency of methods (i) and (ii) for computing the value functions associated with general optimal stopping problems through their field equation representations.

REFERENCES

[1] K. I. Amin and J. N. Bordurtha, *Discrete-time valuation of American options with stochastic interest rates*, Review of Financial Studies, 8 (1995), pp. 193–234.

[2] K. I. Amin and R. A. Jarrow, *Pricing foreign currency options under stochastic interest rates*, J. International Money and Finance, 10 (1991), pp. 310–329.

[3] K. I. Amin and R. A. Jarrow, *Pricing options on risky assets in a stochastic interest rate economy*, in Vasicek and Beyond, Approaches to Building and Applying Interest Rate Models, L. Hughston, ed., RISK Publications, London, 1996, pp. 235–252.

[4] I. Babuška and M. Suri, *The p and hp versions of the finite element method, basic principles and properties*, SIAM Rev., 36 (1994), pp. 578–632.

[5] G. Barles, C. Daher, and M. Romano, *Convergence of numerical schemes for parabolic equations arising in finance theory*, Math. Models Methods Appl. Sci., 5 (1995), pp. 125–143.

[6] M. Baxter and A. Rennie, *Financial Calculus: An Introduction to Derivative Pricing*, Cambridge University Press, Cambridge, UK, 1996.

[7] A. Bensoussan, *On the theory of option pricing*, Acta Appl. Math., 2 (1984), pp. 139–158.
[8] A. Bensoussan and J. L. Lions, *Applications of Variational Inequalities in Stochastic Control*, North-Holland, Amsterdam, The Netherlands, 1982.
[9] F. Black and M. Scholes, *The pricing of options and corporate liabilities*, J. Political Economy, 81 (1973), pp. 637–654.
[10] P. Boyle, *Options: A Monte Carlo Approach*, J. Financial Economics, 4 (1977), pp. 323–338.
[11] A. Brandt and C. W. Cryer, *Multigrid algorithms for the solution of linear complementarity problems arising from free boundary problems*, SIAM J. Sci. Statist. Comput., 4 (1983), pp. 655–684.
[12] M. Broadie and P. Glasserman, *A Stochastic Mesh Method for Pricing High-Dimensional American options*, manuscript, 1997.
[13] S. Choi and M. D. Marcozzi, *On the Valuation of Foreign Currency Options under Stochastic Interest Rates*, manuscript, 1999.
[14] P. G. Ciarlet, *The Finite Element Method for Elliptic Problems*, North-Holland, Amsterdam, The Netherlands, 1978.
[15] N. Clarke and K. Parrot, *The multigrid solution of two factor American put options*, Research report 96-12, Oxford Computing Laboratory, Oxford, UK, 1996.
[16] R. Cottle, J.-S. Pang, and R. E. Stone, *The Linear Complementary Problem*, Academic Press, New York, 1992.
[17] J. Crank, *Free and Moving Boundary Problems*, Oxford University Press, Oxford, UK, 1984.
[18] C. W. Cryer, *The solution of a quadratic programming problem using systematic overrelaxation*, SIAM J. Control, 9 (1971), pp. 385–392.
[19] S. R. Das, *Discrete-time bond and option pricing for jump-diffusion processes*, Review of Derivative Research, 1 (1997), pp. 211–243.
[20] F. Delbaen and W. Schachermayer, *The fundamental theorem of asset pricing*, Math. Ann., 300 (1994), pp. 463–520.
[21] J. Dewynne and P. Wilmott, *Asian options as linear complementary problems: Analysis and finite-difference solutions*, Advances in Futures and Operations Research, 8 (1995), pp. 145–173.
[22] K. Eriksson and C. Johnson, *Adaptive finite element methods for parabolic problems* I*: A linear model problem*, SIAM J. Numer. Anal., 28 (1991), pp. 43–77.
[23] W. H. Fleming and H. M. Soner, *Controlled Markov Processes and Viscosity Solutions*, Springer-Verlag, New York, 1992.
[24] A. Friedman, *Variational Principles and Free-Boundary Problems*, Kreiger, Malabar, FL, 1988.
[25] R. Geske and K. Shastri, *Valuation by approximation: A comparison of alternative option valuation techniques*, J. Financial and Qualitative Analysis, 20 (1985), pp. 45–71.
[26] R. Glowinski, J. L. Lions, and R. Tremolieres, *Numerical Analysis of Variational Inequalities*, North-Holland, Amsterdam, The Netherlands, 1981.
[27] R. Glowinski, *Numerical Methods for Nonlinear Variational Problems*, Springer-Verlag, New York, Berlin, 1984.
[28] R. J. Haber, P. J. Schönbucher, and P. Wilmott, *Pricing Parisian options*, J. Derivatives, Spring (1999), pp. 71–79.
[29] M. Harrison and S. Pliska, *Martingales and stochastic integrals in the theory of continuous trading*, Stochastic Process. Appl., 11 (1981), pp. 215–260.
[30] R. H. W. Hoppe and R. Kornhuber, *Adaptive multilevel methods for obstacle problems*, SIAM J. Numer. Anal., 31 (1994), pp. 301–323.
[31] J. Hull, *Options, Futures, and Other Derivative Securities*, 2nd ed., Prentice-Hall, Englewood Cliffs, NJ, 1993.
[32] P. Jaillet, D. Lamberton, and B. Lapeyre, *Variational inequalities and the pricing of American options*, Acta Appl. Math., 21 (1990), pp. 263–289.
[33] H. Johnson, *Options on the maximum or the minimum of several assets*, J. Fin. Quan. Anal., 22 (1987), pp. 277–283.
[34] I. Karatzas, *On the pricing of American options*, Appl. Math. Optim., 17 (1988), pp. 37–60.
[35] I. Karatzas and S. E. Shreve, *Brownian Motion and Stochastic Calculus*, 2nd ed., Springer-Verlag, New York, Berlin, 1988.
[36] I. Karatzas and S. E. Shreve, *Methods of Mathematical Finance*, Springer-Verlag, New York, 1998.
[37] R. Kornhuber, *Monotone multigrid methods for elliptic variational inequalities* II, Numer. Math., 72 (1996), pp. 481–499.
[38] D. Lamberton and B. Lapeyre, *Introduction to Stochastic Calculus Applied to Finance*, Chapman and Hall, London, UK, 1996.

[39] C. E. Lemke, *Bimatrix equilibrium points and mathematical programming*, Management Sci., 11 (1965), pp. 681–689.
[40] J. Lemke, *A multilevel iterative method for symmetric, positive definite linear complementary problems*, Appl. Math. Optim., 11 (1984), pp. 77–95.
[41] M. D. Marcozzi, *On the Valuation of Asian Option by Variational Methods*, manuscript, 2000.
[42] R. C. Merton, *Theory of rational option pricing*, Bell J. Econom. and Management Sci., 4 (1973), pp. 141–183.
[43] R. Myneni, *The pricing of the American option*, Ann. Appl. Probab., 2 (1992), pp. 1–23.
[44] H. Niederreiter, *Random Number Generation and Quasi-Monte Carlo Methods*, CBMS-NSF Regional Conf. Ser. in Appl. Math. 63, SIAM, Philadelphia, PA, 1992.
[45] J. T. Oden, I. Babŭska, and C. E. Baumann, *A discontinuous hp finite element method for diffusion problems*, J. Comput. Phys., 146 (1998), pp. 495–519.
[46] M. J. Schervish, *Multivariate normal probabilities with error bound*, Appl. Statist., 33 (1984), pp. 81–94.
[47] R. M. Stultz, *Options on the minimum or the maximum of two risky assets: Analysis and applications*, J. Fin. Econ., 10 (1982), pp. 161–185.
[48] P. Van Moerbeke, *On optimal stopping and free boundary problems*, Arch. Ration. Mech. Anal., 60 (1976), pp. 101–148.
[49] P. Wilmott, J. Dewynne, and S. Howison, *Option Pricing: Mathematical Models and Computation*, Oxford Financial Press, Oxford, UK, 1993.
[50] X. L. Zhang, *Numerical analysis of American option pricing in a jump-diffusion model*, Math. Oper. Res., 22 (1997), pp. 668–690.
[51] R. Zvan, P. A. Forsyth, and K. R. Vetzal, *Penalty methods for American options with stochastic volatility*, J. Comput. Appl. Math., 91 (1998), pp. 199–218.

SIAM J. SCI. COMPUT.
Vol. 22, No. 5, pp. 1885–1904
© 2001 Society for Industrial and Applied Mathematics

SYMPLECTIC BALANCING OF HAMILTONIAN MATRICES*

PETER BENNER†

Abstract. We discuss the balancing of Hamiltonian matrices by structure preserving similarity transformations. The method is closely related to balancing nonsymmetric matrices for eigenvalue computations as proposed by Osborne [*J. ACM*, 7 (1960), pp. 338–345] and Parlett and Reinsch [*Numer. Math.*, 13 (1969), pp. 296–304] and implemented in most linear algebra software packages. It is shown that isolated eigenvalues can be deflated using similarity transformations with symplectic permutation matrices. Balancing is then based on equilibrating row and column norms of the Hamiltonian matrix using symplectic scaling matrices. Due to the given structure, it is sufficient to deal with the leading half rows and columns of the matrix. Numerical examples show that the method improves eigenvalue calculations of Hamiltonian matrices as well as numerical methods for solving continuous-time algebraic Riccati equations.

Key words. balancing, eigenvalues, Hamiltonian matrix, symplectic method

AMS subject classifications. 15A18, 65F15, 65F35

PII. S1064827500367993

1. Introduction.

The eigenvalue problem for *Hamiltonian matrices*

$$H = \begin{bmatrix} A & G \\ Q & -A^T \end{bmatrix}, \tag{1.1}$$

where $A, G, Q \in \mathbb{R}^{n\times n}$ and G, Q are symmetric, plays a fundamental role in many algorithms of control theory and other areas of applied mathematics as well as computational physics and chemistry. Computing the eigenvalues of Hamiltonian matrices is required, e.g., when computing the $\mathcal{H}_\infty$–norm of transfer matrices (see, e.g., [9, 10]), when calculating the stability radius of a matrix [14, 30], when computing response functions [23], and when performing many other calculations. Hamiltonian matrices are also closely related to *continuous-time algebraic Riccati equations* (CARE) of the form

$$0 = Q + A^T X + XA - XGX \tag{1.2}$$

with A, G, Q as in (1.1): $X \in \mathbb{R}^{n\times n}$ is a symmetric solution matrix. Many numerical methods for solving (1.2) are based on computing certain invariant subspaces of the related Hamiltonian matrices; see, e.g., [20, 22, 27, 29]. For a detailed discussion of the relations of Hamiltonian matrices and CARE (1.2) we refer to [19].

In eigenvalue computations, matrices and matrix pencils are often preprocessed using a *balancing* procedure as described in [24, 26] for a general matrix $A \in \mathbb{R}^{n\times n}$. First, A is permuted via similarity transformations in order to isolate eigenvalues, i.e., a permutation matrix $P \in \mathbb{R}^{n\times n}$ is computed such that

$$P^T A P = \begin{bmatrix} T_1 & X & Y \\ 0 & Z & W \\ 0 & 0 & T_2 \end{bmatrix}, \tag{1.3}$$

*Received by the editors February 23, 2000; accepted for publication (in revised form) October 6, 2000; published electronically February 8, 2001.
http://www.siam.org/journals/sisc/22-5/36799.html
†Universität Bremen, Fachbereich 3—Mathematik und Informatik, Zentrum für Technomathematik, 28334 Bremen, Germany (benner@math.uni-bremen.de).

where $T_1 \in \mathbb{R}^{p\times p}$ and $T_2 \in \mathbb{R}^{q\times q}$ are upper triangular matrices. Then a diagonal matrix

$$D := \begin{bmatrix} I_p & 0 & 0 \\ 0 & D_Z & 0 \\ 0 & 0 & I_q \end{bmatrix}, \qquad D_Z := \operatorname{diag}(d_{p+1}, \ldots, d_{n-q}),$$

is computed such that rows and columns of $D_Z^{-1} Z D_Z$ are as close in norm as possible. That is, balancing consists of a permutation step and a scaling step. In the scaling step, the rows and columns of a matrix are scaled, which usually leads to a decrease of the matrix norm. This preprocessing step often improves the accuracy of computed eigenvalues significantly; *isolated eigenvalues* (i.e., those contained in T_1 and T_2) are even computed without roundoff error.

Unfortunately, applying this balancing strategy to a Hamiltonian matrix H as given in (1.1) will in general destroy the Hamiltonian structure. This is no problem if the subsequent eigenvalue algorithm does not preserve or use the Hamiltonian structure. But during the past fifteen years, several structure preserving methods for the Hamiltonian eigenproblem have been suggested. In particular, the square-reduced method [32], the Hamiltonian QR algorithm (if in (1.1), $\operatorname{rank} G = 1$ or $\operatorname{rank} Q = 1$) [13], the recently proposed algorithm based on a symplectic URV-like decomposition [7], or the implicitly restarted symplectic Lanczos method of [5] for large sparse Hamiltonian eigenproblems are appropriate choices for developing subroutines for library usage and raise the need for a symplectic balancing routine. Similarity transformations by symplectic matrices preserve the Hamiltonian structure. Thus, in order to balance a Hamiltonian matrix and to preserve its structure, the required permutation matrix and the diagonal scaling matrix should in general be symplectic—only in very special cases can these goals be achieved by nonsymplectic similarity transformations, e.g., if $A = 0$ in (1.1).

In section 2 we will give some necessary background. Isolating eigenvalues of Hamiltonian matrices without destroying the structure can be achieved using symplectic permutation matrices. This will be the topic of section 3. How to equilibrate rows and norms of Hamiltonian matrices in a similar way as proposed in [26] using symplectic diagonal scaling matrices will be presented in section 4. When invariant subspaces, eigenvectors, or solutions of algebraic Riccati equations are the target of the computations, some postprocessing steps are required. These and some other applications of the proposed symplectic balancing method are discussed in section 5. Some numerical examples on the use of the proposed balancing strategy for eigenvalue computation and the numerical solution of algebraic Riccati equations are given in section 6.

2. Preliminaries. The following classes of matrices will be employed in what follows.

DEFINITION 2.1. *Let*

$$J := J_n := \begin{bmatrix} 0 & I_n \\ -I_n & 0 \end{bmatrix}, \tag{2.1}$$

where I_n is the $n\times n$ identity matrix. If the size of the matrix is clear from the context, we leave off the subscript.

(a) *A matrix $H \in \mathbb{R}^{2n\times 2n}$ is* Hamiltonian *if $(HJ)^T = HJ$. The Lie algebra of Hamiltonian matrices in $\mathbb{R}^{2n\times 2n}$ is denoted by $\mathcal{H}_{2n}$.*

(b) *A matrix* $\mathcal{H} \in \mathbb{R}^{2n\times 2n}$ *is* skew-Hamiltonian *if* $(HJ)^T = -HJ$. *The Jordan algebra of skew-Hamiltonian matrices in* $\mathbb{R}^{2n\times 2n}$ *is denoted by* $\mathcal{SH}_{2n}$.

(c) *A matrix* $S \in \mathbb{R}^{2n\times 2n}$ *is* symplectic *if* $SJS^T = J$ *or, equivalently,* $S^TJS = J$. *The Lie group of symplectic matrices in* $\mathbb{R}^{2n\times 2n}$ *is denoted by* $\mathcal{S}_{2n}$.

(d) *A matrix* $U \in \mathbb{R}^{2n\times 2n}$ *is* unitary symplectic *if* $U \in \mathcal{S}_{2n}$ *and* $UU^T = I_{2n}$. *The compact Lie group of unitary symplectic matrices in* $\mathbb{R}^{2n\times 2n}$ *is denoted by* $\mathcal{US}_{2n}$.

Observe that every $H \in \mathcal{H}_{2n}$ has the block representation given in (1.1).

In [12], an important relation between symplectic and Hamiltonian matrices is proved.

PROPOSITION 2.2. *Let* $S \in \mathbb{R}^{2n\times 2n}$ *be nonsingular. Then* $S^{-1}HS$ *is Hamiltonian for all* $H \in \mathcal{H}_{2n}$ *if and only if* $S^TJS = \alpha J$ *for some* $\alpha \in \mathbb{R} \setminus \{0\}$.

This result shows that, in general, similarity transformations that preserve the Hamiltonian structure have to be symplectic up to scaling with a real scalar.

The following proposition shows that the structure of $2n \times 2n$ orthogonal symplectic matrices permits them to be represented as a pair of $n \times n$ matrices. Hence, the arithmetic cost and storage for accumulating orthogonal symplectic matrices can be halved.

PROPOSITION 2.3 (see [25]). *An orthogonal matrix* $U \in \mathbb{R}^{2n\times 2n}$ *is symplectic if and only if it takes the form* $U = \left[\begin{smallmatrix} U_1 & U_2 \\ -U_2 & U_1 \end{smallmatrix}\right]$, *where* $U_i \in \mathbb{R}^{n\times n}$, $i = 1, 2$.

We have the following well-known property of the spectra of Hamiltonian matrices (see, e.g., [19, 22] and the references given therein).

PROPOSITION 2.4. *The spectrum of a real Hamiltonian matrix* H, *denoted by* $\sigma(H)$ *is symmetric with respect to the imaginary axis, i.e., if* $\lambda \in \sigma(H)$, *then* $-\lambda \in \sigma(H)$. *The spectrum of* $H \in \mathcal{H}_{2n}$ *can therefore be partitioned as*

$$\sigma(H) = \{\lambda_1, \ldots, \lambda_k\} \cup \{-\lambda_1, \ldots, -\lambda_k\} \cup \{\pm\lambda_{k+1}, \ldots, \pm\lambda_n\} =: \Delta \cup (-\Delta) \cup \Delta_0,$$

where $Re(\lambda_j) > 0$, $j = 1, \ldots, k$, *and* $Re(\lambda_j) = 0$, $j = k+1, \ldots, n$.

In many applications, $\Delta_0 = \emptyset$. Note that the multiplicities of the eigenvalues collected in Δ_0 play an essential role for the existence of structured Schur forms for Hamiltonian matrices as outlined below.

Real Schur forms play an important role when solving eigenvalue problems for nonsymmetric matrices. For Hamiltonian matrices, Schur forms should be computed in a structure-preserving way.

DEFINITION 2.5. (a) *Let* $\hat{H} \in \mathcal{H}_{2n}$. *If* $\hat{H}$ *has the form*

$$\hat{H} = \begin{bmatrix} \hat{A} & \hat{G} \\ 0 & -\hat{A}^T \end{bmatrix}, \tag{2.2}$$

where $\hat{A} \in \mathbb{R}^{n\times n}$ *is in real Schur form (quasi-upper triangular) and* $\hat{G} = \hat{G}^T$, *then* $\hat{H}$ *is* real Hamiltonian quasi-triangular.

(b) *If* $H \in \mathcal{H}_{2n}$ *and there exists* $U \in \mathcal{US}_{2n}$ *such that* $\hat{H} = U^THU$ *is real Hamiltonian quasi-triangular, then* $\hat{H}$ *is in* real Hamiltonian Schur form *and* U^THU *is called a* Hamiltonian Schur decomposition.

DEFINITION 2.6. *A subspace* $\mathcal{V} \subset \mathbb{R}^{2n}$ *is called* isotropic *if* $x^TJy = 0$ *for all* $x, y \in \mathcal{V}$. *It is called* Lagrangian *if it is isotropic and maximal, i.e., if it is not contained in a larger isotropic subspace* $\mathcal{W} \subset \mathbb{R}^{2n}$.

If a Hamiltonian Schur decomposition exists such that $\hat{H}$ is as is (2.2), then U can be chosen such that $\sigma(\hat{A})$ is contained in the closed right half plane. Moreover, the first n columns of the orthogonal symplectic matrix U in the Hamiltonian Schur

decomposition span a Lagrangian H-invariant subspace. Using the notation from Proposition 2.4, if $\Delta_0 = \emptyset$ and $\sigma(\hat{A}) = -\Delta$, then we say that this H-invariant subspace is *c-stable*. That is, if it exists, the c-stable H-invariant subspace corresponds to the n eigenvalues of H in the open left half plane.

Most of the structure-preserving methods for the Hamiltonian eigenproblem, i.e., those using symplectic (similarity) transformations, rely on the following result. For Hamiltonian matrices with no purely imaginary eigenvalues this result was first stated in [25], while in its full generality as given below it has been proved in [21].

THEOREM 2.7. *Let $H \in \mathcal{H}_{2n}$, and let $i\alpha_1, \ldots, i\alpha_\ell$ be its pairwise distinct nonzero purely imaginary eigenvalues. Furthermore, let the associated H-invariant subspaces be spanned by the columns of $U_k \in \mathbb{R}^{2n\times p_k}$, $k = 1, \ldots, \ell$. Then the following are equivalent.*

(i) *There exists $S \in \mathcal{S}_{2n}$ such that $S^{-1}HS$ is real Hamiltonian quasi-triangular.*
(ii) *There exists $U \in \mathcal{US}_{2n}$ such that U^THU is in real Hamiltonian Schur form.*
(iii) *$U_k^H J_n U_k$ is congruent to J_{p_k} for all $k = 1, \ldots, \ell$.*

Note that from Theorem 2.7 it follows that purely imaginary eigenvalues of $H \in \mathcal{H}_{2n}$ (if any) must have even algebraic multiplicity in order for the Hamiltonian Schur form of H to exist.

3. Isolating eigenvalues by symplectic permutations. Let P denote any $n \times n$ permutation matrix. It is easy to see that symplectic permutation matrices have the form

$$P_s = \begin{bmatrix} P & 0 \\ 0 & P \end{bmatrix}. \tag{3.1}$$

With matrices of type (3.1) it is possible to transform a Hamiltonian matrix to the form

$$\tilde{H} = P_s H P_s = \left[\begin{array}{ccc|ccc} A_{11} & A_{12} & A_{13} & G_{11} & G_{12} & G_{13} \\ 0 & A_{22} & A_{23} & G_{12}^T & G_{22} & 0 \\ 0 & 0 & A_{33} & G_{13}^T & 0 & 0 \\ \hline 0 & 0 & Q_{13} & -A_{11}^T & 0 & 0 \\ 0 & Q_{22} & Q_{23} & -A_{12}^T & -A_{22}^T & 0 \\ Q_{13}^T & Q_{23}^T & Q_{33} & -A_{13}^T & -A_{23}^T & -A_{33}^T \end{array}\right] \begin{array}{l} \}p \\ \}n-p-q =: r \\ \}q \\ \}p \\ \}n-p-q = r \\ \}q \end{array}, \tag{3.2}$$
$$\underbrace{}_{p}\ \underbrace{}_{r}\ \underbrace{}_{q}\ \underbrace{}_{p}\ \underbrace{}_{r}\ \underbrace{}_{q}$$

where A_{11}, A_{33} are upper triangular and either $Q_{13} = 0$ or $G_{13} = 0$. The existence of such a P_s will be proved in a constructive way later by Algorithm 3.4, which transforms a Hamiltonian matrix to the form given in (3.2). From a Hamiltonian matrix having the form (3.2) a total of $2(p+q)$ eigenvalues of H can be read off directly as seen by the following result.

LEMMA 3.1. *Let $H \in \mathcal{H}_{2n}$ and $p, q, r \in \mathbb{N}_0$ with $r = n - p - q$, where H is of the form* (3.2) *and either $G_{13} = 0$ or $Q_{13} = 0$. Then there exists a permutation matrix $P \in \mathbb{R}^{2n\times 2n}$ such that*

$$T = P^THP = \begin{bmatrix} T_1 & Y & Z \\ 0 & H_{22} & W \\ 0 & 0 & T_2 \end{bmatrix}, \tag{3.3}$$

where $T_1, T_2 \in \mathbb{R}^{(p+q)\times(p+q)}$ are upper triangular with

$$\sigma(T_1) = \sigma(-T_2) \tag{3.4}$$

and

$$H_{22} = \begin{bmatrix} A_{22} & G_{22} \\ Q_{22} & -A_{22}^T \end{bmatrix} \tag{3.5}$$

is a $2r \times 2r$ *Hamiltonian submatrix of* H.

Proof. Let H be as in (3.2) and

$$P_1 := \left[\begin{array}{ccc|ccc} 0 & I_p & 0 & 0 & 0 & 0 \\ 0 & 0 & I_r & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & I_q \\ \hline 0 & 0 & 0 & 0 & I_p & 0 \\ 0 & 0 & 0 & I_r & 0 & 0 \\ I_q & 0 & 0 & 0 & 0 & 0 \end{array}\right]. \tag{3.6}$$

Then

$$P_1^T H P_1 = \left[\begin{array}{cc|cc|cc} -A_{33}^T & Q_{13}^T & Q_{23}^T & -A_{23}^T & -A_{13}^T & Q_{33} \\ G_{13}^T & A_{11} & A_{12} & G_{12} & G_{11} & A_{13} \\ \hline 0 & 0 & A_{22} & G_{22} & G_{12}^T & A_{23} \\ 0 & 0 & Q_{22} & -A_{22}^T & -A_{12}^T & Q_{23} \\ \hline 0 & 0 & 0 & 0 & -A_{11}^T & Q_{13} \\ 0 & 0 & 0 & 0 & G_{13} & A_{33} \end{array}\right]. \tag{3.7}$$

Define

$$\Pi_m := \begin{bmatrix} & & 1 \\ & \diagup & \\ 1 & & \end{bmatrix} \in \mathbb{R}^{m \times m}. \tag{3.8}$$

If $G_{13} = 0$, then set $P_2 := \operatorname{diag}(\Pi_q, I_p, I_r, I_r, \Pi_p, I_q)$. Otherwise ($Q_{13} = 0$), set

$$P_2 := \left[\begin{array}{cc|cc|cc} 0 & \Pi_q & & & & \\ I_p & 0 & & & & \\ \hline & & I_r & 0 & & \\ & & 0 & I_r & & \\ \hline & & & & 0 & \Pi_p \\ & & & & I_q & 0 \end{array}\right].$$

Thus, $T := P_2^T P_1^T H P_1 P_2$ has the desired form. The relation (3.4) follows from

$$\begin{bmatrix} 0 & I_p \\ -I_q & 0 \end{bmatrix}^{-1} \begin{bmatrix} -A_{11}^T & Q_{13} \\ G_{13} & A_{33} \end{bmatrix} \begin{bmatrix} 0 & I_p \\ -I_q & 0 \end{bmatrix} = - \begin{bmatrix} -A_{33}^T & Q_{13}^T \\ G_{13}^T & A_{11} \end{bmatrix}^T. \qquad \square$$

Lemma 3.1 is merely of theoretical interest and demonstrates that in order to solve the Hamiltonian eigenvalue problem, we can proceed by working only with H_{22}. But the transformations we have used in the proof are, in general, not symplectic. If we want to compute invariant subspaces, eigenvectors, and/or the Hamiltonian Schur form given in Theorem 2.7, we can transform the Hamiltonian matrix in (3.2) such that it has Hamiltonian Schur form in rows and columns $1, \ldots, p+q$ and $n+1, \ldots, n+p+q$. But this cannot, in general, be accomplished using only symplectic permutation matrices of the form (3.1). Therefore, we need another class of transformation matrices.

DEFINITION 3.2. *A matrix $P_J \in \mathbb{R}^{2n\times 2n}$ is called a* J-permutation matrix *if*
(a) *it is symplectic, i.e., $P_J^T J P_J = J$,*
(b) *$(P_J)_{j,k} \in \{-1, 0, 1\}$ for $j,k = 1,\ldots,2n$, and*
(c) *each row and column have exactly one nonzero entry.*

As $P_J \in \mathcal{US}_{2n}$, it is clear that a similarity transformation by a J-permutation matrix preserves the Hamiltonian structure. In analogy to standard permutations, similarity transformations with P_J can be performed without floating point operations. Moreover, they can be represented by a signed integer vector IP of length n, where $\mathtt{IP}(k) = \pm j$, $k = 1,\ldots,n$, $j = 1,\ldots,2n$, if rows and columns k, j are to be interchanged while the sign indicates if the corresponding entry in P_J is $+1$ or -1. The entries of P_J in rows $n+1$ to $2n$ can be deduced from IP and Proposition 2.3. Furthermore, symplectic permutation matrices as given in (3.1) are J-permutation matrices.

LEMMA 3.3. *For any $H \in \mathcal{H}_{2n}$ having the form* (3.2)*, there exists a J-permutation matrix P_J such that*

$$\hat{H} \;=\; P_J^T H P_J \;=\; \left[\begin{array}{cc|cc} \hat{A}_{11} & \hat{A}_{12} & \hat{G}_{11} & \hat{G}_{12} \\ 0 & \hat{A}_{22} & \hat{G}_{12}^T & \hat{G}_{22} \\ \hline 0 & 0 & -\hat{A}_{11}^T & 0 \\ 0 & \hat{Q}_{22} & -\hat{A}_{12}^T & -\hat{A}_{22}^T \end{array}\right] \begin{array}{l} \}p+q \\ \}n-p-q=r \\ \}p+q \\ \}n-p-q=r \end{array}, \tag{3.9}$$

where $\hat{A}_{11}$ is upper triangular and, with the notation in (3.2)*,*

$$\hat{A}_{22} \;=\; A_{22}, \qquad \hat{G}_{22} \;=\; G_{22}, \qquad \hat{Q}_{22} \;=\; Q_{22}.$$

Proof. Let a Hamiltonian matrix H be given as in (3.2). We need a J-permutation matrix only in the first step. Let

$$P_1 \;:=\; \left[\begin{array}{ccc|ccc} I_p & 0 & 0 & 0 & 0 & 0 \\ 0 & I_r & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & I_q \\ \hline 0 & 0 & 0 & I_p & 0 & 0 \\ 0 & 0 & 0 & 0 & I_r & 0 \\ 0 & 0 & -I_q & 0 & 0 & 0 \end{array}\right].$$

Obviously, P_1 is a J-permutation matrix and

$$H_1 \;:=\; P_1^T H P_1 \;=\; \left[\begin{array}{ccc|ccc} A_{11} & A_{12} & -G_{13} & G_{11} & G_{12} & A_{13} \\ 0 & A_{22} & 0 & G_{12}^T & G_{22} & A_{23} \\ -Q_{13}^T & -Q_{23}^T & -A_{33}^T & A_{13}^T & A_{23}^T & -Q_{33} \\ \hline 0 & 0 & 0 & -A_{11}^T & 0 & Q_{13} \\ 0 & Q_{22} & 0 & -A_{12}^T & -A_{22}^T & Q_{23} \\ 0 & 0 & 0 & G_{13}^T & 0 & A_{33} \end{array}\right].$$

Now assume $G_{13} = 0$. Set $P_2 := \operatorname{diag}(\tilde{P}_2, \tilde{P}_2)$, where

$$\tilde{P}_2 \;:=\; \left[\begin{array}{ccc} 0 & I_p & 0 \\ 0 & 0 & I_r \\ I_q & 0 & 0 \end{array}\right].$$

Then

$$H_2 \;:=\; P_2^T H_1 P_2 \;=\; \left[\begin{array}{ccc|ccc} -A_{33}^T & -Q_{13}^T & -Q_{23}^T & -Q_{33} & A_{13}^T & A_{23}^T \\ 0 & A_{11} & A_{12} & A_{13} & G_{11} & G_{12} \\ 0 & 0 & A_{22} & A_{23} & G_{12}^T & G_{22} \\ \hline 0 & 0 & 0 & A_{33} & 0 & 0 \\ 0 & 0 & 0 & Q_{13} & -A_{11}^T & 0 \\ 0 & 0 & Q_{22} & Q_{23} & -A_{12}^T & -A_{22}^T \end{array}\right].$$

We thus obtain the form (3.9) by another similarity transformation with

$$P_3 := \operatorname{diag}(\Pi_q,\, I_p,\, I_r,\, \Pi_q,\, I_p,\, I_r),$$

where Π_q is defined in (3.8). For the other case, i.e., $Q_{13} = 0$, set

$$\tilde{P}_2 \;=\; \left[\begin{array}{ccc} I_p & 0 & 0 \\ 0 & 0 & I_r \\ 0 & I_q & 0 \end{array}\right],$$

and $P_2 := \operatorname{diag}(\tilde{P}_2,\, \tilde{P}_2)$, $P_3 := \operatorname{diag}(I_p,\, \Pi_q,\, I_r,\, I_p,\, \Pi_q,\, I_r)$.

In both cases, $\hat{P} := P_1 P_2 P_3$ is a J-permutation matrix and $\hat{H} := \hat{P}^T H \hat{P}$ is a Hamiltonian matrix having the desired form (3.9). □

In order to isolate eigenvalues, it is sufficient to restrict ourselves to symplectic permutations. But having computed the form (3.2), it is possible that there are still isolated eigenvalues in H_{22}. Applying the same procedure used to isolate eigenvalues in H to H_{22}, we can transform H_{22} to the form (3.2). This process can then be repeated until no more isolated eigenvalues are found. Accumulating all permutations in a symplectic permutation matrix P_s of the form (3.1) results in a similarity transformation

$$\tilde{H} = P_s^T H P_s \tag{3.10}$$

$$= \left[\begin{array}{ccccc|ccccc} A_{11} & \dots & A_{1,t} & \dots & A_{1,s} & G_{11} & \dots & G_{1,t} & \dots & G_{1,s} \\ 0 & \ddots & \vdots & & \vdots & \vdots & & \vdots & \diagup & 0 \\ \vdots & \ddots & A_{t,t} & \dots & A_{t,s} & G_{1,t}^T & \dots & G_{t,t} & 0 & \vdots \\ \vdots & & \ddots & \ddots & \vdots & \vdots & \diagup & 0 & & \vdots \\ 0 & \dots & \dots & 0 & A_{s,s} & G_{1,s}^T & 0 & \dots & \dots & 0 \\ \hline 0 & \dots & \dots & 0 & Q_{1,s} & -A_{11}^T & 0 & \dots & \dots & 0 \\ \vdots & & 0 & \diagup & \vdots & \vdots & \ddots & \ddots & & \vdots \\ \vdots & 0 & Q_{t,t} & \dots & Q_{t,s} & -A_{t,s}^T & \dots & -A_{t,t}^T & \ddots & \vdots \\ 0 & \diagup & \vdots & & \vdots & \vdots & & \vdots & \ddots & 0 \\ Q_{1,s}^T & \dots & Q_{t,s}^T & \dots & Q_{s,s} & -A_{1,s}^T & \dots & -A_{t,s}^T & \dots & -A_{s,s}^T \end{array}\right] \begin{array}{l} \}p_1 \\ \}p_2 + \ldots + p_{t-1} \\ \}p_t \\ \}p_{t+1} + \ldots + p_{s-1} \\ \}p_s \\ \}p_1 \\ \}p_2 + \ldots + p_{t-1} \\ \}p_t \\ \}p_{t+1} + \ldots + p_{s-1} \\ \}p_s \end{array}.$$

Here, $A_{j,j} \in \mathbb{R}^{p_j \times p_j}$, $j = 1, \ldots, s$, $j \neq t$, are upper triangular and for $j = 1, \ldots, t-1$, either $Q_{j,s-j+1} = 0$ or $G_{j,s-j+1} = 0$ and the Hamiltonian submatrix

$$H_{t,t} = \left[\begin{array}{cc} A_{t,t} & G_{t,t} \\ Q_{t,t} & -A_{t,t}^T \end{array}\right] \tag{3.11}$$

has no isolated eigenvalues. If we now define $p := \sum_{j=1}^{t-1} p_j$, $q := \sum_{j=t+1}^{s} p_j$, then we can partition $\tilde{H}$ in (3.10) as in (3.2). Then the first step in the proof of Lemma 3.1 can be performed to obtain the form (3.7). Just the block-structure of the upper left and lower right diagonal blocks in (3.7) is more complicated. But it is still possible to bring them to upper triangular form using repeatedly the same sequence of permutations used in the proof of Lemma 3.1. This shows that $2(p+q)$ eigenvalues of the Hamiltonian matrix can be read off directly from (3.10) and that $\tilde{H}$ is permutationally similar to

$$\begin{bmatrix} \tilde{T}_1 & \tilde{Y} & \tilde{Z} \\ 0 & H_{t,t} & \tilde{W} \\ 0 & 0 & \tilde{T}_2 \end{bmatrix} = \begin{bmatrix} \blacksquare & \square & \square \\ & \square & \square \\ & & \blacksquare \end{bmatrix}.$$

Further, we have $H_{t,t} \in \mathbb{R}^{2r \times 2r}$, where now $r := p_t = n - p - q$, and

$$\sigma(H) = \sigma(H_{t,t}) \cup \bigcup_{\substack{j=1 \\ j \neq t}}^{s} \sigma(A_{j,j}) \cup \bigcup_{\substack{j=1 \\ j \neq t}}^{s} \sigma(-A_{j,j}^T). \tag{3.12}$$

If only eigenvalues are required, we can continue working only with $H_{t,t}$. If also eigenvectors and/or invariant subspaces are required, the similarity transformations used to solve the reduced-order eigenproblem for $H_{t,t}$ have to be applied to the whole matrix $\tilde{H}$. In that case, $\tilde{H}$ should be transformed to the form given in (3.9). Partitioning $\tilde{H}$ from (3.10) as in (3.2), we can perform the first step of the proof of Lemma 3.3 with the J-permutation matrix P_1. The subsequent steps to achieve upper triangular form in the first $p+q$ rows and columns then have to be performed for each of the first $t-1$ block columns, distinguishing the cases $Q_{j,s-j+1} = 0$ or $G_{j,s-j+1} = 0$ for $j = 1, \ldots, t-1$.

A procedure to transform a Hamiltonian matrix H to the form in (3.10) is given in the following algorithm. Note that in the given algorithm, $p_j = 1$ for all $j = 1, \ldots, s$, $j \neq t$.

ALGORITHM 3.4.

Input: *Matrices $A, G, Q \in \mathbb{R}^{n \times n}$, where $Q = Q^T$, $G = G^T$, defining a Hamiltonian matrix $H \in \mathbb{R}^{2n \times 2n}$.*

Output: *A symplectic permutation matrix P_s; matrices A, G, Q with $G = G^T$, $Q = Q^T$ defining a Hamiltonian matrix $H = P_s^T H P_s$ having the form (3.10); integers $i_l = p_1 + \cdots + p_{t-1} + 1 = p + 1$ and $i_h = i_l + p_t = n - q$.*

$P_s = I_{2n}$
$i_l = 1, \quad i_h = n$
$\tilde{i}_l = 0, \quad \tilde{i}_h = n$
`WHILE` $(i_l \neq \tilde{i}_l)$ `AND` $(i_h \neq \tilde{i}_h)$
  $\tilde{i}_l = i_l, \quad \tilde{i}_h = i_h$
  $i = i_h$
  `WHILE` $(i > i_l)$ `AND` $(i_h = \tilde{i}_h)$
    $\tilde{i}_h = i_h$
    $r = \sum_{\substack{j=i_l \\ j \neq i}}^{i_h} |a_{ij}| + \sum_{j=i_l}^{i_h} |g_{ij}|$

```
        IF (r = 0) THEN
            P̃ = I_n + e_i e_{i_h}^T + e_{i_h} e_i^T − e_i e_i^T − e_{i_h} e_{i_h}^T
            A = P̃^T A P̃,      G = P̃^T G P̃,      Q = P̃^T Q P̃
            P = P · diag(P̃, P̃)
            i_h = i_h − 1
        END IF
        i = i − 1
    END WHILE
    i = i_l
    WHILE (i < i_h) AND (i_l = ĩ_l)
        ĩ_l = i_l
        c = Σ_{j=i_l, j≠i}^{i_h} |a_ji| + Σ_{j=i_l}^{i_h} |q_ji|
        IF (c = 0) THEN
            P̃ = I_n + e_i e_{i_l}^T + e_{i_l} e_i^T − e_i e_i^T − e_{i_l} e_{i_l}^T
            A = P̃^T A P̃,      G = P̃^T G P̃,      Q = P̃^T Q P̃
            P = P · diag(P̃, P̃)
            i_l = i_l + 1
        END IF
        i = i + 1
    END WHILE
  END WHILE
END
```

In each execution of the outer `WHILE`-loop, we first search a row isolating an eigenvalue. If such a row is found, we look for a column isolating an eigenvalue. In this fashion it can be guaranteed that at the end, there are no more isolated eigenvalues, although we always only touch the first n rows and columns of the Hamiltonian matrix.

In an actual implementation one would of course never form the permutation matrices explicitly but store the relevant information in an integer vector. Multiplications by permutation matrices are realized by swapping the data contained in the rows or columns to be permuted; for details, see, e.g., [3].

It is rather difficult to give a complete account of the cost of Algorithm 3.4. If there are no isolated eigenvalues, the algorithm requires $4n^2 - 2n$ floating point additions and $2n$ comparisons as opposed to $8n^2 - 4n$ additions and $4n$ comparisons for the unstructured permutation procedure from [26] as implemented in the LAPACK subroutine xGEBAL [3] when applied to $H \in \mathbb{R}^{2n\times 2n}$. The worst case for Algorithm 3.4 would be that in each execution of the outer `WHILE`-loop, an isolated eigenvalue is found in the last execution of the second inner `WHILE`-loop. In that case, the cost consists of $4n^3/3 + n^2 + O(n)$ floating point additions, $n^2 + n$ comparisons, and moving $2n^2 + 2n$ floating point numbers. But in this worst-case analysis, all eigenvalues are isolated such that after permuting, there is nothing left to do, and the Hamiltonian matrix is in Hamiltonian Schur form. A worst-case study for xGEBAL shows that the permutation part requires $8n^3/3 - 2n^2 + O(n)$ additions, $2n^2 + n$ comparisons, and moving $4n^2 + 2n$ floating point numbers. We can therefore conclude that Algorithm 3.4 is about half as expensive as the procedure proposed in [26] applied to a Hamiltonian matrix.

4. Symplectic scaling. Suppose now that we have transformed the Hamiltonian matrix to the form (3.10). Since all subsequent transformations are determined from $H_{t,t}$, the scaling parameters to balance $H_{t,t}$ now have to be chosen such that the rows and columns of $H_{t,t}$ (instead of $\tilde{H}$) are as close in norm as possible.

In order to simplify notation, in what follows we will call the Hamiltonian matrix again H. Let H_{off} be the off-diagonal part of H, i.e.,

$$H_{\text{off}} = H - \operatorname{diag}(H).$$

We may without loss of generality assume that none of the rows and columns of H_{off} vanishes identically. Otherwise, we could isolate another pair of eigenvalues.

Now we want to scale H such that the norms of its rows and columns are close in norm. As noted before, employing the technique of Parlett and Reinsch [26] destroys the Hamiltonian structure. Diagonal scaling has thus to be performed using a symplectic diagonal matrix D_s. Such a matrix must have the form,

$$D_s = \begin{bmatrix} D & 0 \\ 0 & D^{-1} \end{bmatrix}, \tag{4.1}$$

where $D \in \mathbb{R}^{n\times n}$ is a nonsingular diagonal matrix.

Let us at first note an obvious result for Hamiltonian matrices. Here and in what follows we will use the *colon notation* (see, e.g., [16]) $H(:,k)$, $H(j,:)$ to indicate the kth column and jth row, respectively, of a matrix H.

LEMMA 4.1. *Let $H \in \mathbb{R}^{2n\times 2n}$ be a Hamiltonian matrix. Then for all $p \geq 1$ and for all $i = 1,\ldots,n$,*

$$\|H(:,i)\|_p = \|H(n+i,:)^T\|_p, \tag{4.2}$$

$$\|H(i,:)^T\|_p = \|H(:,n+i)\|_p, \tag{4.3}$$

i.e., the p-norm of the ith column equals the norm of the $(n+i)$th row and the norm of the ith row equal the norm of the $(n+i)$th column.

Proof. The result is obvious by noting that $\|x\|_p = (\sum_{k=1}^{2n} |x_k|^p)^{\frac{1}{p}}$ for $x \in \mathbb{R}^{2n}$ and by observing that from the structure of Hamiltonian matrices, we have

$$\begin{aligned} H(:,i) &= [a_{1,i}, a_{2,i}, \ldots, a_{n,i}, q_{1,i}, q_{2,i}, \ldots, q_{n,i}]^T, \\ H(n+i,:) &= [q_{i,1}, q_{i,2}, \ldots, q_{i,n}, -a_{1,i}, -a_{2,i}, \ldots, -a_{n,i}], \end{aligned}$$

and, furthermore, $q_{ij} = q_{ji}$ for all $1 \leq i,j \leq n$. Equation (4.3) follows analogously by noting that $g_{ij} = g_{ji}$ for all $1 \leq i,j \leq n$. ∎

We can now conclude that it is sufficient to equilibrate the norms of the first n rows and columns of a $2n \times 2n$ Hamiltonian matrix by using a consequence of Lemma 4.1.

COROLLARY 4.2. *Let $H \in \mathbb{R}^{2n\times 2n}$ be a Hamiltonian matrix. Then for all $p \geq 1$ and for all $i = 1,\ldots,n$,*

$$\|H(:,i)\|_p = \|H(i,:)\|_p \iff \|H(:,n+i)\|_p = \|H(n+i,:)\|_p.$$

Since a similarity transformation with any diagonal matrix does not affect the diagonal elements of the transformed matrix, it is in the following sufficient to consider H_{off}. We will employ the notation

$$\begin{aligned} h_i &:= H_{\text{off}}(:,i) \;=\; i\text{th column of } H_{\text{off}}, \\ h^i &:= H_{\text{off}}(i,:)^T \;=\; \text{transpose of the } i\text{th row of } H_{\text{off}}. \end{aligned}$$

In what follows, we will for convenience use $p = 1$. The results also hold for any other p-norm. From a computational point of view it is also reasonable to use the 1–norm since its computation does not involve any floating point multiplications, and, furthermore, reducing the norm of H in one norm usually implies also a reduction in the other norms.

Equilibrating $\|h_i\|_1$ and $\|h^i\|_1$ can now be achieved in a similar way as in the Parlett–Reinsch method. If β denotes the base of the floating point arithmetic and σ_i is any signed integer, then they compute β^{σ_i} closest to the real scalar

$$\delta_i = \sqrt{\|h^i\|_1/\|h_i\|_1}.$$

Thus, with $D^{(i)} = I_{2n} + (\delta_i - 1)e_i e_i^T$ and $\hat{H} = (D^{(i)})^{-1} H D^{(i)}$, it follows that $\|\hat{h}^i\|_1 = \|\hat{h}_i\|_1$. But if $\delta_i \neq 1$, $\hat{H}$ is in general no longer Hamiltonian. Unfortunately, using the symplectic diagonal matrix

$$D_s^{(i)} = \begin{bmatrix} D_i & 0 \\ 0 & D_i^{-1} \end{bmatrix},$$

where $D_i = I_n + (\delta_i - 1)e_i e_i^T$, and computing

$$\tilde{H} = (D_s^{(i)})^{-1} H D_s^{(i)}, \tag{4.4}$$

we obtain

$$\|\tilde{h}_i\|_1 \;=\; \|\hat{h}_i\|_1 - \delta_i|q_{ii}| + \delta_i^2|q_{ii}|, \qquad \|\tilde{h}^i\|_1 \;=\; \|\hat{h}^i\|_1 - \frac{1}{\delta_i}|g_{ii}| + \frac{1}{\delta_i^2}|g_{ii}| \tag{4.5}$$

and thus in general, $\|\tilde{h}_i\|_1 \neq \|\tilde{h}^i\|_1$.

Nevertheless, equilibrating the 1-norms of h_i and h^i can be achieved by requiring $\|\tilde{h}_i\|_1 = \|\tilde{h}^i\|_1$ and solving the resulting quartic equation

$$\delta_i(\|h_i\|_1 - |q_{ii}|) + \delta_i^2|q_{ii}| \;=\; \frac{1}{\delta_i}(\|h^i\|_1 - |g_{ii}|) + \frac{1}{\delta_i^2}|g_{ii}|. \tag{4.6}$$

It remains to show that (4.6) has a positive solution.

THEOREM 4.3. *Let $H \in \mathbb{R}^{2n\times 2n}$ be a Hamiltonian matrix, and denote its off-diagonal part by H_{off}. Assume that none of the rows and columns of H_{off} vanishes identically. Then there exists a unique real number $\delta_i > 0$ such that for $\tilde{H}$ as in* (4.4) *we have*

$$\|\tilde{h}_i\|_1 = \|\tilde{h}^i\|_1 = \|\tilde{h}_{n+i}\|_1 = \|\tilde{h}^{n+i}\|_1.$$

Proof. Solutions of (4.6) are nonzero solutions of

$$0 = |q_{ii}|\delta_i^4 + (\|h_i\|_1 - |q_{ii}|)\delta_i^3 + (|g_{ii} - \|h^i\|_1)\delta_i - |g_{ii}| =: p(\delta_i), \tag{4.7}$$

where $p(t) = \sum_{k=0}^{4} a_k t^k$. Recalling that $g_{ii} = (h^i)_{n+i}$, $q_{ii} = (h_i)_{n+i}$, the coefficients of the polynomial p satisfy

$$\begin{aligned} a_0 &= -|g_{ii}| \;\le\; 0, \qquad a_1 \;=\; |g_{ii}| - \|h^i\|_1 \;\le\; 0, \qquad a_2 \;=\; 0, \\ a_3 &= \|h_i\|_1 - |q_{ii}| \;\ge\; 0, \qquad a_4 \;=\; |q_{ii}| \ge 0. \end{aligned}$$

Since there is at most one change of sign in the coefficients of the polynomial p, Descartes' rule of signs shows that there is at most one positive zero of p. So if

there exists a positive solution of (4.6), it is unique. By assumption, $\|h_i\|_1 \neq 0$ and $\|h^i\|_1 \neq 0$. Therefore, either $a_0 < 0$ or $a_1 < 0$ (as g_{ii} is part of h^i), and either $a_3 > 0$ or $a_4 > 0$ (as q_{ii} is part of h_i). Thus, we know that p is a polynomial of degree at least 3 with positive leading coefficient, and hence $\lim_{t\to\infty} p(t) = +\infty$.

On the other hand, if $g_{ii} \neq 0$, then $p(0) < 0$, and, using the mean value theorem, it follows that there exists a positive zero of p. If $g_{ii} = 0$, then $t = 0$ is a zero of p and $a_1 < 0$. The third order polynomial $q(t) = a_4t^3 + a_3t^2 + a_1$ has a positive zero because of the mean value theorem and $q(0) = a_1 < 0$ as well as $\lim_{t\to\infty} q(t) = +\infty$. Thus, by $p(t) = tq(t)$, p has again at least one positive zero, and hence (4.6) has at least one positive real solution regardless of the value of g_{ii}. On the other hand, it was already observed that there is at most one such solution, and we can conclude that there exists a unique $\delta_i > 0$ solving (4.6) whence $\|\tilde{h}_i\|_1 = \|\tilde{h}^i\|_1$.

The other equalities follow immediately from Corollary 4.2. □

Computing the exact value of δ_i that is needed to equilibrate the ith and $(n+i)$th rows and columns would require the solution of the fourth-order equation (4.6). Since the diagonal similarity transformations are to be chosen from the set of machine numbers, it is sufficient to find the machine number β^{σ_i} closest to δ_i. This can be done similarly to the computation in the general case as proposed in [26] and as implemented in the Fortran 77 subroutines BALANC from EISPACK [15] or in its successor xGEBAL from LAPACK [3]. That is, starting with $\delta_i = 1$, the quantities in (4.5) are evaluated and compared. If $\|\tilde{h}_i\|_1 < \|\tilde{h}^i\|_1$, then this is repeated for $\delta_i = \beta^k$, $k = 1, 2, \ldots$ until $\|\tilde{h}_i\|_1 > \|\tilde{h}^i\|_1$. Otherwise, if $\|\tilde{h}_i\|_1 > \|\tilde{h}^i\|_1$, then we use $\delta_i = \beta^{-k}$, $k = 1, 2, \ldots$ until $\|\tilde{h}_i\|_1 < \|\tilde{h}^i\|_1$. This is achieved by the following algorithm.

ALGORITHM 4.4.

Input: *Hamiltonian matrix $H \in \mathcal{H}_{2n}$ having no isolated eigenvalues, β = base of floating point arithmetic.*

Output: *Diagonal matrix $D_s \in \mathcal{S}_{2n}$, H is overwritten by $D_s^{-1}HD_s \in \mathcal{H}_{2n}$ with row and column norms equilibrated as far as possible.*

$D_s = I_{2n}$
FOR $k = 1, 2, \ldots$ until convergence,
  FOR $i = 1, \ldots, n$,
    $\delta = 1$
    $c = \|h_i\|_1$, $q_a = |h_{n+i,i}|$, $c_a = c - q_a$,
    $r = \|h^i\|_1$, $g_a = |h_{i,n+i}|$, $r_a = r - g_a$,
    WHILE $c < r$
      $c_a = \beta c_a$, $q_a = \beta^2 q_a$, $c = c_a + q_a$,
      $r_a = r_a/\beta$, $g_a = g_a/\beta^2$, $r = r_a + g_a$,
      $\delta = \beta\delta$,
    END WHILE
    IF $\delta = 1$ THEN
      WHILE $r < c$
        $c_a = c_a/\beta$, $q_a = q_a/\beta^2$, $c = c_a + q_a$,
        $r_a = \beta(r_a - g_a)$, $g_a = \beta^2 g_a$, $r = r_a + g_a$,
        $\delta = \delta/\beta$,
      END WHILE
    END IF
    $D_i = I + (\delta - 1)e_ie_i^T$, $D_s = D_s\mathrm{diag}\left(D_i,\, D_i^{-1}\right)$,
    $H = \mathrm{diag}\left(D_i^{-1},\, D_i\right) \cdot H \cdot \mathrm{diag}\left(D_i,\, D_i^{-1}\right)$,

```
    END FOR i
  END FOR k
END
```

One execution of the outer FOR-loop of Algorithm 4.4 can be considered as a sweep. The algorithm is terminated if, for a whole sweep, all $D_i = I_n$. Usually, the row and column norms are approximately equal after very few sweeps. Afterwards, the iteration makes only very limited progress. Therefore, Parlett and Reinsch propose a modification in [26] that, translated to our problem, becomes the following.

Let δ_i be determined by the two inner WHILE*-loops of Algorithm* 4.4*, and compute*

$$c_i = \delta_i(\|h_i\|_1 - |q_{ii}|) + \delta_i^2|q_{ii}|, \qquad r_i = \frac{1}{\delta_i}(\|h^i\|_1 - |g_{ii}|) + \frac{1}{\delta_i^2}|g_{ii}|. \tag{4.8}$$

If $(c_i + r_i) < \gamma(\|h_i\|_1 + \|h^i\|_1)$ (where γ is a given positive constant), then compute D_i as in Algorithm 4.4*. Otherwise, set $D_i = I_n$.*

For $\gamma = 1$, the behavior is essentially the same as for Algorithm 4.4. (In a few cases, Algorithm 4.4 increases $\|h_i\|_1 + \|h^i\|_1$, which cannot happen if $\gamma = 1$.) For γ slightly smaller than one, a step is skipped if it would produce an insubstantial reduction of $\|h_i\|_1 + \|h^i\|_1$.

In an actual computation, the similarity transformations with the D_i's can be applied directly to the blocks A, G, and Q of the Hamiltonian matrix without forming the Hamiltonian matrix itself. Thus, each similarity transformation can be performed using only $4n - 4$ multiplications. When the standard (not structure-preserving) scaling procedure from [26] is applied to H, each similarity transformation requires $4n - 2$ multiplications. (Recall that in Algorithm 4.4, two rows and columns are equilibrated at a time, while only one row and column are treated in each step of the inner FOR-loop of the standard procedure.)

The number of sweeps required to converge is similar to that for the general case since the theory derived in [26] requires only the assumption of similarity transformations with diagonal matrices and that in step i of each sweep, the ith rows and columns are equilibrated as far as possible with $\delta_i = \beta^{\sigma_i}$. But this is accomplished by Algorithm 4.4. Moreover, if δ_i is taken as the exact solution of (4.6), the convergence of the sequence of similarity transformations to a stationary point can be proved as in [24, 17]. That is, if $\delta_i^{(k)}$ is the solution of (4.6) in sweep k, then $\lim_{k\to\infty} \delta_i^{(k)} = 1$ for all $i = 1, \ldots, n$, and hence, in the limit, H is a balanced Hamiltonian matrix.

Note that here each sweep has length n, while in the standard balancing algorithm one has to go through each row/column pair of the matrix, and thus each sweep has length $2n$. Thus, the computational cost for scaling a $2n \times 2n$ Hamiltonian matrix by Algorithm 4.4, assuming k_1 sweeps are required, is $4n^2k_1 + O(k_1n)$ as opposed to $8n^2k_2 + O(k_2n)$ for the standard scaling procedure as given in [26] with assumed k_2 sweeps required for convergence. In general, $k_1 \approx k_2$ such that the structure-preserving scaling strategy is about half as expensive as the standard procedure. These flop counts are based on the assumption that the cost for determining the δ_i can be considered as small ($O(1)$) compared to the similarity transformations.

Remark 4.5. In [18] it is proposed to solve the matrix balancing problem using a convex programming approach. To compare the complexity of this approach to that of Algorithm 4.4, suppose that Algorithm 4.4 terminates after k_1 sweeps with

$|\,\|h_i\|_1 - \|h^i\|_1\,| < \mu_i$. For the matrix H to be balanced, let

$$\mathcal{E} := \{(i,j) \in \{1,\ldots,n\} \times \{1,\ldots,n\} \mid i \neq j,\ \ h_{ij} \neq 0\},$$

$\tau = \sum_{(i,j)\in\mathcal{E}} |h_{ij}|$, and $h_{\min} = \min\{|h_{ij}|\,|\,(i,j) \in \mathcal{E}\}$. Assume that $\sum_{i=1}^{n} \mu_i^2 =: \mu^2$ and $\mu < e\tau$. (Here, $e = \exp(1)$.) Then Theorem 5 in [18] states that the complexity of computing a diagonal matrix Y with positive diagonal entries such that the rows and columns of $Y^{-1}HY$ are balanced with the same accuracy as achieved by Algorithm 4.4 is $O(n^4 \ln(\frac{ne\tau}{\mu} \ln \frac{\tau}{h_{\min}}))$. From numerical experience, it can be assumed that $k_1 = O(1)$ with respect to n. Hence, Algorithm 4.4 can be considered to be of complexity $O(n^2)$. This complexity is clearly superior to that of the convex programming approach which is still the case if $k_1 = O(n)$.

Algorithm 4.4 requires a careful implementation to guard against over- and underflow due to a very large/small δ_i. Here, we can use the bounds discussed in [26] and implemented in LAPACK subroutine xGEBAL [3]; we just have to take into account that in each step we scale by $\beta^{\pm 2}$ rather than β as in xGEBAL.

5. Backtransformation, ordering of eigenvalues, and applications. So far we have considered only the problem of computing the eigenvalues of a Hamiltonian matrix. In order to compute eigenvectors, invariant subspaces, and the solutions of algebraic Riccati equations, we have to transform the Hamiltonian matrix to real Schur form. As we are considering structure-preserving methods, the goal is to transform the Hamiltonian matrix to real Hamiltonian Schur form as given in Theorem 2.7(a)—if it exists.

Assume that we have applied Algorithm 3.4 to the Hamiltonian matrix and obtained a symplectic permutation matrix P_s such that $P_s^T H P_s$ has the form given in (3.10). Then, we have applied a J-permutation P_J to the permuted Hamiltonian matrix such that its rows and columns numbered $1,\ldots,p+q,\ n+1,\ldots,n+p+q$ are in Hamiltonian Schur form, i.e., $P_J^T P_s^T H P_s P_J$ has the form given in (3.9). (From Lemma 3.3 we know that such a P_J exists.) Next, we have applied Algorithm 4.4 to the Hamiltonian submatrix $H_{t,t} \in \mathcal{H}_{2r}$ from (3.11) and obtained a diagonal matrix $D = \operatorname{diag}\left(D_t, D_t^{-1}\right) \in \mathcal{S}_{2r}$. Let

$$D_s \;=\; \begin{bmatrix} I_{p+q} & 0 & 0 & 0 \\ 0 & D_r & 0 & 0 \\ 0 & 0 & I_{p+q} & 0 \\ 0 & 0 & 0 & D_r^{-1} \end{bmatrix}.$$

Then

$$\hat{H} \;:=\; D_s^{-1} P_J^T P_s^T H P_s P_j D_s \;=\; \begin{bmatrix} \hat{A}_{11} & \hat{A}_{12} & \hat{G}_{11} & \hat{G}_{12} \\ 0 & \hat{A}_{22} & \hat{G}_{12}^T & \hat{G}_{22} \\ 0 & 0 & -\hat{A}_{11}^T & 0 \\ 0 & \hat{Q}_{22} & -\hat{A}_{12}^T & -\hat{A}_{22}^T \end{bmatrix},$$

where $\hat{A}_{11} \in \mathbb{R}^{(p+q)\times(p+q)}$ is upper triangular and the Hamiltonian submatrix $\hat{H}_{22} := \left[\begin{smallmatrix} \hat{A}_{22} & \hat{G}_{22} \\ \hat{Q}_{22} & -\hat{A}_{22}^T \end{smallmatrix}\right]$ has no isolated eigenvalues and its rows and columns are equilibrated by Algorithm 4.4. Now assume that the Hamiltonian Schur form of $\hat{H}_{22}$ exists and that we have computed $U_{22} = \left[\begin{smallmatrix} \hat{U}_{22} & -\hat{V}_{22} \\ \hat{V}_{22} & \hat{U}_{22} \end{smallmatrix}\right] \in \mathcal{US}_{2r}$, transforming $\hat{H}_{22}$ into real Hamiltonian

Schur form. Set

$$U := \begin{bmatrix} I_{p+q} & 0 & 0 & 0 \\ 0 & \hat{U}_{22} & 0 & -\hat{V}_{22} \\ 0 & 0 & I_{p+q} & 0 \\ 0 & \hat{V}_{22} & 0 & \hat{U}_{22} \end{bmatrix}$$

and $S := P_s P_J D_s U$. Then $H_1 := S^{-1} H S$ is real Hamiltonian quasi-triangular and $S \in \mathcal{S}_{2n}$. The first n columns of S span a Lagrangian H-invariant subspace. In most applications, the c-stable H-invariant subspace is desired. Let us assume the method used to transform $\hat{H}_{22}$ to Hamiltonian Schur form chooses U_{22} such that the first r columns of U_{22}, i.e., the columns of $\left[\begin{smallmatrix} \hat{U}_{22} \\ \hat{V}_{22} \end{smallmatrix}\right]$, span the $\hat{H}_{22}$-invariant subspace of choice. But there is no guarantee that the isolated eigenvalues in $\hat{A}_{11}$ are the desired ones. In that case, we have to reorder the Hamiltonian Schur form in order to move the undesired eigenvalues to the lower right block of $\hat{H}$ and the desired ones to the upper left block. Assume that we want to compute the Lagrangian H-invariant subspace corresponding to a set $\Lambda = \{\lambda_1 \ldots, \lambda_n\} \subset \sigma(H)$, which is closed under complex conjugation. (Note that this is a necessary condition in order to obtain a Lagrangian invariant subspace [2].) Using the standard reordering algorithm for the real Schur form of an $n \times n$ nonsymmetric matrix as given in [16, 31], we can find an orthogonal matrix $\tilde{U}_\Lambda$ such that with the orthogonal symplectic matrix $U_\Lambda = \operatorname{diag}(\tilde{U}_\Lambda, \tilde{U}_\Lambda)$ we have that

$$H_2 := U_\Lambda^T H_1 U_\Lambda = \begin{bmatrix} \tilde{A}_{11} & \tilde{A}_{12} & \tilde{G}_{11} & \tilde{G}_{12} \\ 0 & \tilde{A}_{22} & \tilde{G}_{12}^T & \tilde{G}_{22} \\ 0 & 0 & -\tilde{A}_{11}^T & 0 \\ 0 & 0 & -\tilde{A}_{12}^T & -\tilde{A}_{22}^T \end{bmatrix} \begin{matrix} \}n-k \\ \}k \\ \}n-k \\ \}k \end{matrix}, \tag{5.1}$$

where A_{11}, A_{22} are quasi-upper triangular, and that

$$\Lambda = \sigma(\tilde{A}_{11}) \cup \sigma(-\tilde{A}_{22}^T), \qquad -\Lambda = \sigma(\tilde{A}_{22}) \cup \sigma(-\tilde{A}_{11}^T). \tag{5.2}$$

Therefore, we have to swap the eigenvalues in $\tilde{A}_{22}$ and $-\tilde{A}_{22}^T$. Note that the eigenvalues to be reordered are among the isolated eigenvalues and hence are real. This implies that $\tilde{A}_{22}$ is upper triangular. The reordering can be achieved analogously to the reordering of eigenvalues in the real Schur form as given in [16, 31]. The following procedure uses this standard reordering in order to swap eigenvalues within $\tilde{A}_{22}$ (and $-\tilde{A}_{22}^T$) and requires rotations working exclusively in rows and columns n and $2n$ in order to exchange eigenvalues from $\tilde{A}_{22}$ with eigenvalues from $-\tilde{A}_{22}^T$. Assume $(H_2)_{n,n} = (\tilde{A}_{22})_{kk} = -\lambda_n$. Then $(H_2)_{2n,2n} = (-\tilde{A}_{22}^T)_{kk} = \lambda_n$. Let $\left[\begin{smallmatrix} c_{\lambda_n} & -s_{\lambda_n} \\ s_{\lambda_n} & c_{\lambda_n} \end{smallmatrix}\right]$ be a 2×2 Givens rotation matrix that annihilates the second component of $\left[\begin{smallmatrix} \tilde{g}_{nn} \\ -2\lambda_n \end{smallmatrix}\right]$, where $\tilde{g}_{nn} = (H_2)_{n,2n} = (\tilde{G}_{22})_{kk}$. Define

$$U_{\lambda_n} := I_{2n} + (c-1)(e_n e_n^T + e_{2n} e_{2n}^T) + s(e_{2n} e_n^T - e_n e_{2n}^T).$$

Then U_{λ_n} is a symplectic Givens rotation matrix acting in planes n and $2n$, and

$$U_{\lambda_n}^T H_2 U_{\lambda_n} = \left[\begin{array}{ccc|ccc} \tilde{A}_{11} & \tilde{A}_{12} & \bar{a}_{1,n} & \tilde{G}_{11} & \tilde{G}_{12} & \bar{g}_{1n} \\ 0 & \tilde{A}_{22} & \vdots & \tilde{G}_{12}^T & \tilde{G}_{22} & \vdots \\ 0 & 0 & \lambda_n & \bar{g}_{1n} & \cdots & \bar{g}_{nn} \\ \hline 0 & 0 & 0 & -\tilde{A}_{11}^T & 0 & 0 \\ 0 & 0 & 0 & -\tilde{A}_{12}^T & -\tilde{A}_{22}^T & 0 \\ 0 & 0 & 0 & -\bar{a}_{1,n} & \cdots & -\lambda_n \end{array}\right] \begin{matrix} \}n-k \\ \}k-1 \\ \}1 \\ \}n-k \\ \}k-1 \\ \}1 \end{matrix}.$$

Here, the bar indicates elements changed by the similarity transformation.

The next step is now to move λ_n up in the upper diagonal block using again the standard ordering subroutine such that we obtain again the form given in (5.1), just $\tilde{A}_{11} \in \mathbb{R}^{(n-k+1)\times(n-k+1)}$, and again the relations (5.2) hold. This procedure has now to be repeated until $k = 0$ and $\Lambda = \sigma(\hat{A}_{11})$.

Remark 5.1. If the Hamiltonian matrix has the form $H = \left[\begin{smallmatrix} A & BB^T \\ C^TC & -A^T \end{smallmatrix}\right]$, which corresponds to a linear system $\dot{x} = Ax + Bu$, $y = Cx$, with (A, B) stabilizable and (C, A) detectable (see, e.g., [19, 29]), then each isolated eigenvalue in (5.1) given by the diagonal elements of $\tilde{A}_{11}$ has negative real part. Otherwise, these eigenvalues are unstable or undetectable and cannot be stabilized/detected. Therefore, if we have not mixed up blocks by the J-permutation matrix P_J (i.e., in Algorithm 3.4, $i_h = n$) and the c-stable H-invariant subspace is required, no reordering is necessary.

Remark 5.2. When solving algebraic Riccati equations using any approach based on the Hamiltonian eigenproblem, the symplectic balancing strategy proposed here is often not enough to minimize errors caused by ill-scaling. This is due to the effect that for a balanced Hamiltonian matrix $H = \left[\begin{smallmatrix} A & G \\ Q & -A^T \end{smallmatrix}\right]$ we may still have $\|Q\| \gg \|G\|$, which can cause large errors when computing invariant subspaces [28]. Therefore, another symplectic scaling using a similarity transformation with $\operatorname{diag}(\sqrt{\rho}I_n, \frac{1}{\sqrt{\rho}}I_n) \in \mathcal{S}_{2n}$, $\rho \in \mathbb{R}$, should be applied to H in order to achieve $\|A\| \approx \|G\| \approx \|Q\|$ as far as possible; see [4] for details and a discussion of several heuristic strategies to achieve this.

Remark 5.3. Everything derived so far for Hamiltonian matrices can be applied in the same way to skew-Hamiltonian matrices. If $N \in \mathcal{SH}_{2n}$, then $N = \left[\begin{smallmatrix} A & G \\ Q & A^T \end{smallmatrix}\right]$ with $G = -G^T$, $Q = -Q^T$. The skew-Hamiltonian structure is again preserved under symplectic similarity transformations. Hence, isolating eigenvalues, reordering, etc., can be achieved in the same way as for Hamiltonian matrices as all considered transformations do not depend on the signs in the matrix blocks A, G, Q but only on the distinction zero/nonzero when isolating eigenvalues and on the absolute values of the entries when equilibrating rows and norms. Note that Algorithm 4.4 even simplifies quite a lot for *real* skew-Hamiltonian matrices. As $q_{ii} = g_{ii} =$ for all $i = 1, \ldots, n$, the scaling factor δ_i can be computed as in the general balancing algorithm for nonsymmetric matrices because in (4.5) we obtain $\|\tilde{h}_i\|_1 = \|\tilde{h}^i\|_1$.

Eigenvalues of skew-Hamiltonian matrices as well as a skew-Hamiltonian Schur form can be computed in a numerically strong backward stable way by Van Loan's method [32]. It is advisable to balance skew-Hamiltonian matrices using the proposed strategies prior to applying this algorithm.

Remark 5.4. We have considered so far only real Hamiltonian and skew-Hamiltonian matrices. Isolating eigenvalues and equilibrating rows and columns for complex (skew-)Hamiltonian matrices can be achieved in exactly the same way. A structure-preserving, numerically backward stable (and hence numerically strong backward stable [11]) method for solving the complex (skew-)Hamiltonian eigenproblem has recently been proposed [8]. The proposed symplectic balancing method can (and should) also be used prior to applying this algorithm.

6. Numerical examples. We have tested the symplectic balancing strategy for eigenvalue computations. The computations were done in MATLAB[1] version 5.2 with machine precision $\varepsilon \approx 2.2204 \times 10^{-16}$. Algorithms 3.4 and 4.4 were implemented as

[1] MATLAB is a trademark of The MathWorks, Inc.

MATLAB functions. We used the modified algorithm as suggested by (4.8), where we set $\gamma = 0.95$ as suggested in [26] and as implemented in the LAPACK subroutine xGEBAL [3]. The eigenvalues of the balanced and the unbalanced Hamiltonian matrix were computed by the square-reduced method using a MATLAB function `sqred`, which implements the explicit version of the square-reduced method (see [32]).

We also tested the effects of symplectic balancing for the numerically backward stable, structure-preserving method for the Hamiltonian eigenvalue problem presented in [7]. Like the square-reduced method, this algorithm uses the square of the Hamiltonian matrix. But it avoids forming the square, explicitly using a symplectic URV-type decomposition of the Hamiltonian matrix.

As reference values we used the eigenvalues computed by the unsymmetric QR algorithm with Parlett–Reinsch balancing as implemented in the LAPACK expert driver routine DGEEVX [3], applied to the Hamiltonian matrix and using quadruple precision.

Moreover, we tested the effects of balancing when solving algebraic Riccati equations with the structure-preserving multishift method presented in [1] for the examples from the benchmark collection [6]. We only present some of the most intriguing results.

Example 6.1 (see [6, Example 6]). The system data come from an optimal control problem for a J-100 jet engine as a special case of a multivariable servomechanism problem. The resulting Hamiltonian matrix $H \in \mathbb{R}^{60\times 60}$ has eight isolated eigenvalues: triple eigenvalues at ± 20.0 and simple eigenvalues at ± 33.3.

Algorithm 3.4 returns $i_l = 5$ and $i_h = 30$, and for the permuted Hamiltonian matrix we have

$$\begin{aligned} H_{1:i_l-1,1:i_l-1} &= \operatorname{diag}(-33.3,\ -20.0,\ -20.0,\ -20.0\,), \\ H_{n+1:n+i_l-1,n+1:n+i_l-1} &= \operatorname{diag}(\,33.3,\ 20.0,\ 20.0,\ 20.0\,). \end{aligned}$$

Next, the Hamiltonian submatrix

$$H_{22} \;=\; \begin{bmatrix} H_{5:30,5:30} & H_{5:30,35:60} \\ H_{35:60,5:30} & -H_{5:30,5:30}^T \end{bmatrix}$$

is scaled using Algorithm 4.4. After six sweeps, we obtain the balanced Hamiltonian matrix H_b. We have $\|H\|_2 = 1.44 \times 10^8$ and $\|H_b\|_2 = 6.54 \times 10^2$; that is, we have decreased the 2-norm of the matrix used in the subsequent eigenvalue computation by more than five orders of magnitude. If the eigenvalues are computed by the square-reduced method applied to the unbalanced Hamiltonian matrix, the triplet of isolated eigenvalues is returned as a pair of conjugate complex eigenvalues with relative errors $\approx 4.06 \times 10^{-11}$ and a simple eigenvalue with relative error $\approx 3.96 \times 10^{-11}$. For the simple eigenvalue at 33.3, the relative error is $\approx 7.7 \times 10^{-15}$. For the balanced version, these eigenvalues are returned with full accuracy since they are not affected by roundoff errors. The relative errors for the other (not isolated) eigenvalues are given in Figure 6.1, where we use the relative distance of the computed eigenvalues to those computed by DGEEVX as an estimate of the real relative error.

Figure 6.1 only contains the relative errors for the eigenvalues with positive real parts as `sqred` returns the eigenvalues as exact plus-minus pairs. The "+" for the 26th eigenvalue is missing as the computed relative error for the balanced version is zero with respect to machine precision. The eigenvalues are ordered by increasing absolute values. From Figure 6.1, the increasing accuracy for decreasing ratio $\|H\|_2/|\lambda|$ is

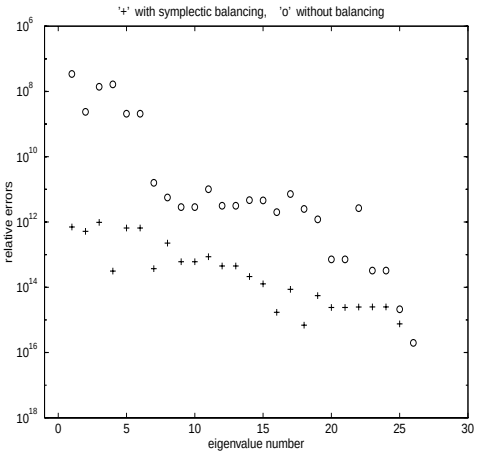


FIG. 6.1. *Square-reduced method.*

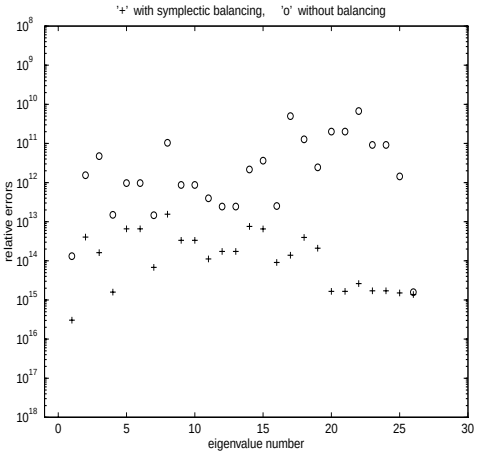


FIG. 6.2. *Symplectic URV + periodic QR.*

obvious—with or without balancing. All computed eigenvalues of the balanced matrix are more accurate than for the unbalanced one. The increase in accuracy is more significant for the eigenvalues of smaller magnitude. This reflects the decrease of the ratios $\|H\|_2/|\lambda|$, which more or less determines the accuracy of the computed eigenvalues; see [32]. The decrease factor for $\|H\|_2$ is about 5×10^{-6}. The accuracy for the eigenvalues of smaller magnitude increases by almost the same factor.

From Figure 6.2 we see that symplectic balancing also improves the eigenvalues computed by the method proposed in [7]. As the method does not suffer from the $\|H\|_2/|\lambda|$ perturbation, the accuracy for all computed eigenvalues is similar. Also note that in the unbalanced version, the isolated eigenvalues are computed with a relative accuracy ranging from 7.0×10^{-14} to 1.2×10^{-15}.

Using the balanced matrix in order to solve algebraic Riccati equations by the multishift method as described in [1], we obtain the following results. If the multishift method is applied to the unbalanced data, the computed solution yields a residual

$$r_F := \|Q + A^T X + XA - XGX\|_F \tag{6.1}$$

of size 1.5×10^{-6}, while using the balanced Hamiltonian matrix we get $r_F = 8.1 \times 10^{-10}$. This shows that numerical methods for solving algebraic Riccati equations can be substantially improved by employing balancing.

Example 6.2 (see [6, Example 13]). The Hamiltonian matrix is defined as in (1.1) with

$$A = \begin{bmatrix} 0 & 0.4 & 0 & 0 \\ 0 & 0 & 0.345 & 0 \\ 0 & -0.524 \cdot \tau & -0.465 \cdot \tau & 0.262 \cdot \tau \\ 0 & 0 & 0 & -\tau \end{bmatrix}, \qquad \begin{aligned} G &= \tau^2 e_4 e_4^T, \\ Q &= \operatorname{diag}(1,\ 0,\ 1,\ 0), \end{aligned}$$

where $\tau = 10^6$ and e_4 denotes the fourth unit vector. After four sweeps of Algorithm 4.4, $\|H\|_2$ is reduced from 10^{12} to 1.5×10^6. The accuracy of the computed eigenvalues did not improve significantly, but for the stabilizing solution of the algebraic Riccati equation, the Frobenius norm of the residual as defined in (6.1) dropped from $r_F = 5.4 \times 10^{-11}$ to $r_F = 1.8 \times 10^{-15}$.

7. Concluding remarks. We have seen that isolated eigenvalues of a real Hamiltonian matrix can be deflated using similarity transformations with symplectic permutation matrices, the deflated problem can be scaled in order to reduce the norm of the deflated Hamiltonian matrix and to equilibrate its row and column norms, and the remaining (not isolated) eigenvalues then can be determined by computing the eigenvalues of the deflated, balanced Hamiltonian submatrix. If invariant subspaces are required, then we can use J-permutation matrices and a symplectic reordering strategy in order to obtain the desired invariant subspaces. The same method can be applied in order to balance skew-Hamiltonian and complex (skew-)Hamiltonian matrices.

Numerical examples demonstrate that symplectic balancing can significantly improve the accuracy of eigenvalues of Hamiltonian matrices as well as the accuracy of solutions of the associated algebraic Riccati equations computed by structure-preserving methods.

Final remark and acknowledgments. The work presented in this article continues preliminary results derived in [4]. The author would like to thank Ralph Byers, Heike Faßbender, and Volker Mehrmann for helpful suggestions.

REFERENCES

[1] G. Ammar, P. Benner, and V. Mehrmann, *A multishift algorithm for the numerical solution of algebraic Riccati equations*, Electron. Trans. Numer. Anal., 1 (1993), pp. 33–48.

[2] G. Ammar and V. Mehrmann, *On Hamiltonian and symplectic Hessenberg forms*, Linear Algebra Appl., 149 (1991), pp. 55–72.

[3] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, and D. Sorensen, *LAPACK Users' Guide*, 2nd ed., SIAM, Philadelphia, 1995.

[4] P. Benner, *Contributions to the Numerical Solution of Algebraic Riccati Equations and Related Eigenvalue Problems*, Dissertation, Fakultät für Mathematik, TU Chemnitz–Zwickau, 09107 Chemnitz, Germany, 1997.

[5] P. Benner and H. Fassbender, *An implicitly restarted symplectic Lanczos method for the Hamiltonian eigenvalue problem*, Linear Algebra Appl., 263 (1997), pp. 75–111.

[6] P. Benner, A. Laub, and V. Mehrmann, *A Collection of Benchmark Examples for the Numerical Solution of Algebraic Riccati Equations* I: *Continuous-Time Case*, Tech. report SPC 95_22, Fakultät für Mathematik, TU Chemnitz–Zwickau, 09107 Chemnitz, Germany, FRG, 1995. Available from http://www.tu-chemnitz.de/sfb393/spc95pr.html.

[7] P. Benner, V. Mehrmann, and H. Xu, *A numerically stable, structure preserving method for computing the eigenvalues of real Hamiltonian or symplectic pencils*, Numer. Math., 78 (1998), pp. 329–358.

[8] P. Benner, V. Mehrmann, and H. Xu, *A note on the numerical solution of complex Hamiltonian and skew-Hamiltonian eigenvalue problems*, Electron. Trans. Numer. Anal., 8 (1999), pp. 115–126.

[9] S. Boyd, V. Balakrishnan, and P. Kabamba, *A bisection method for computing the $\mathcal{H}_\infty$ norm of a transfer matrix and related problems*, Math. Control Signals Systems, 2 (1989), pp. 207–219.

[10] N. Bruinsma and M. Steinbuch, *A fast algorithm to compute the H_∞-norm of a transfer function matrix*, Systems Control Lett., 14 (1990), pp. 287–293.

[11] J. Bunch, *The weak and strong stability of algorithms in numerical algebra*, Linear Algebra Appl., 88 (1987), pp. 49–66.

[12] A. Bunse-Gerstner, *Matrix factorization for symplectic QR-like methods*, Linear Algebra Appl., 83 (1986), pp. 49–77.

[13] R. Byers, *A Hamiltonian QR-algorithm*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 212–229.

[14] R. Byers, *A bisection method for measuring the distance of a stable matrix to unstable matrices*, SIAM J. Sci. Statist. Comput., 9 (1988), pp. 875–881.

[15] B. Garbow, J. Boyle, J. Dongarra, and C. Moler, *Matrix Eigensystem Routines—EISPACK Guide Extension*, Lecture Notes in Comput. Sci. 51, Springer-Verlag, New York, 1977.
[16] G. Golub and C. Van Loan, *Matrix Computations*, 3rd ed., Johns Hopkins University Press, Baltimore, 1996.
[17] J. Grad, *Matrix balancing*, Comput. J., 14 (1971), pp. 280–284.
[18] B. Kalantari, L. Khachiyan, and A. Shokoufandeh, *On the complexity of matrix balancing*, SIAM J. Matrix Anal. Appl., 18 (1997), pp. 450–463.
[19] P. Lancaster and L. Rodman, *The Algebraic Riccati Equation*, Oxford University Press, Oxford, UK, 1995.
[20] A. Laub, *Invariant subspace methods for the numerical solution of Riccati equations*, in The Riccati Equation, S. Bittanti, A. Laub, and J. Willems, eds., Springer-Verlag, Berlin, 1991, pp. 163–196.
[21] W.-W. Lin, V. Mehrmann, and H. Xu, *Canonical forms for Hamiltonian and symplectic matrices and pencils*, Linear Algebra Appl., 301–303 (1999), pp. 469–533.
[22] V. Mehrmann, *The Autonomous Linear Quadratic Control Problem, Theory and Numerical Solution*, Lecture Notes in Control and Inform. Sci. 163, Springer-Verlag, Heidelberg, 1991.
[23] J. Olson, H. Jensen, and P. Jørgensen, *Solution of large matrix equations which occur in response theory*, J. Comput. Phys., 74 (1988), pp. 265–282.
[24] E. E. Osborne, *On pre-conditioning of matrices*, J. ACM, 7 (1960), pp. 338–345.
[25] C. Paige and C. Van Loan, *A Schur decomposition for Hamiltonian matrices*, Linear Algebra Appl., 14 (1981), pp. 11–32.
[26] B. Parlett and C. Reinsch, *Balancing a matrix for calculation of eigenvalues and eigenvectors*, Numer. Math., 13 (1969), pp. 296–304.
[27] P. Petkov, N. Christov, and M. Konstantinov, *Computational Methods for Linear Control Systems*, Prentice-Hall, Hertfordshire, UK, 1991.
[28] P. Petkov, M. Konstantinov, D. Gu, and I. Postlethwaite, *Solving Continuous-Time Matrix Algebraic Riccati Equations with Condition and Accuracy Estimates*, Tech. report 94–64, Control Systems Research, Department of Engineering, Leicester University, Leicester LE1 7RH, UK, 1994.
[29] V. Sima, *Algorithms for Linear-Quadratic Optimization*, Monogr. Textbooks, Pure Appl. Math. 200, Marcel Dekker, New York, 1996.
[30] J. Sreedhar, P. Van Dooren, and A. Tits, *A fast algorithm to compute the real structured stability radius*, in Stability Theory (Ascona, 1995), Internat. Ser. Numer. Math. 121, Birkhäuser, Basel, 1996, pp. 219–230.
[31] G. W. Stewart, *Algorithm* 506*—HQR3 and EXCHNG: Fortran subroutines for calculating and ordering the eigenvalues of a real upper Hessenberg matrix*, ACM Trans. Math. Software, 2 (1976), pp. 275–280.
[32] C. Van Loan, *A symplectic method for approximating all the eigenvalues of a Hamiltonian matrix*, Linear Algebra Appl., 61 (1984), pp. 233–251.

SIAM J. SCI. COMPUT.
Vol. 22, No. 6, pp. 1905–1925
© 2001 Society for Industrial and Applied Mathematics

STRUCTURE-PRESERVING METHODS FOR COMPUTING EIGENPAIRS OF LARGE SPARSE SKEW-HAMILTONIAN/HAMILTONIAN PENCILS*

VOLKER MEHRMANN† AND DAVID WATKINS‡

Abstract. We study large, sparse generalized eigenvalue problems for matrix pencils, where one of the matrices is Hamiltonian and the other is skew-Hamiltonian. Problems of this form arise in the numerical simulation of elastic deformation of anisotropic materials, in structural mechanics, and in the linear quadratic control problem for partial differential equations. We develop a structure-preserving skew-Hamiltonian, isotropic, implicitly restarted shift-and-invert Arnoldi algorithm (SHIRA). Several numerical examples demonstrate the superiority of SHIRA over a competing unstructured method.

Key words. skew-Hamiltonian/Hamiltonian pencil, generalized eigenvalue problem, quadratic eigenvalue problem, implicitly restarted Arnoldi method, Lamé equations, classical mechanics, linear quadratic control, algebraic Riccati equation

AMS subject classifications. 65F15, 65N25, 65N30, 65Y05

PII. S1064827500366434

1. Introduction.

In this paper we study the numerical computation of a small number of eigenvalues (and the associated eigenvectors) of large-scale generalized eigenvalue problems having a certain structure that arises frequently in applications. A $2n \times 2n$ real matrix pencil

$$\alpha\mathcal{N} - \beta\mathcal{H} = \alpha \begin{bmatrix} F_1 & G_1 \\ H_1 & F_1^T \end{bmatrix} - \beta \begin{bmatrix} F_2 & G_2 \\ H_2 & -F_2^T \end{bmatrix}, \tag{1.1}$$

where $G_1 = -G_1^T$, $H_1 = -H_1^T$, $G_2 = G_2^T$, and $H_2 = H_2^T$, is called a *skew-Hamiltonian/Hamiltonian* pencil or, more briefly, an *SHH* pencil. Throughout this paper we will assume that $\mathcal{N}$ and $\mathcal{H}$ are large and sparse. Several examples of applications that give rise to large, sparse generalized eigenvalue problems with SHH pencils are given in section 2.

The reason for the terminology is simply this: matrices the form of $\mathcal{N}$ and $\mathcal{H}$ in (1.1) are called *skew-Hamiltonian* and *Hamiltonian*, respectively. If

$$J = \begin{bmatrix} 0 & I_n \\ -I_n & 0 \end{bmatrix}, \tag{1.2}$$

where I_n is the $n \times n$ identity matrix, then skew-Hamiltonian matrices satisfy $(\mathcal{N}J)^T = -(\mathcal{N}J)$ and Hamiltonian matrices satisfy $(\mathcal{H}J)^T = \mathcal{H}J$.

*Received by the editors February 3, 2000; accepted for publication (in revised form) August 28, 2000; published electronically February 21, 2001.
http://www.siam.org/journals/sisc/22-6/36643.html

†Fachbereichß Mathematik, Ma 4-5, Technische Universität Berlin, Str. des 17 Juni 136, D–10623 Berlin, Fed. Rep. Germany (mehrmann@math.tu-berlin.de). The work of this author was supported by Deutsche Forschungsgemeinschaft within project A8 of SFB393 Numerische Simulation auf massiv parallelen Rechnern.

‡Department of Mathematics, Washington State University, Pullman, WA, 99164-3113 (watkins@wsu.edu). The work of this author was supported by Deutsche Forschungsgemeinschaft within project A8 of SFB393 Numerische Simulation auf massiv parallelen Rechnern and by Deutscher Akademischer Austauschdienst within the program HSP III, Förderung ausländischer Gastdozenten.

An SHH pencil and, in particular, its spectrum have considerable structure. The eigenvalues occur in quadruplets $(\lambda, \bar{\lambda}, -\lambda, -\bar{\lambda})$ or in real or purely imaginary pairs $(\lambda, -\lambda)$ [24, 25, 26]. The objective of this paper is to develop algorithms that preserve and exploit this structure. The payoffs are more efficient and more accurate algorithms. In some cases, preservation of the structure is crucial to the stable solution of a problem.

Typical applications require the few eigenvalues that are smallest in magnitude or closest to the imaginary axis. To achieve this we must apply transformations that have the effect of shifting the desired eigenvalues to the periphery of the spectrum, so that they can be computed efficiently by Krylov subspace methods, e.g., Arnoldi or Lanczos, possibly with implicit restarts. This is a standard procedure in methods for large sparse eigenvalue problems, [33, 39]. What is special to this paper is that our transformations and our Krylov subspace methods respect the structure of the problem.

Our approach requires that the skew-Hamiltonian matrix $\mathcal{N}$ be presented as a product in the special form

$$\mathcal{N} = \mathcal{Z}_1\mathcal{Z}_2, \quad \text{where} \quad \mathcal{Z}_2^T J = \pm J\mathcal{Z}_1, \tag{1.3}$$

with $\mathcal{Z}_1$ and $\mathcal{Z}_2$ sparse. This allows us to transform the pencil $\alpha\mathcal{N} - \beta\mathcal{H}$ to a standard eigenvalue problem $\alpha I - \beta\mathcal{W} = \alpha I - \beta\mathcal{Z}_1^{-1}\mathcal{H}\mathcal{Z}_2^{-1}$, in which the matrix $\mathcal{W} = \mathcal{Z}_1^{-1}\mathcal{H}\mathcal{Z}_2^{-1}$ turns out to be Hamiltonian.

This procedure is analogous to the technique by which a symmetric pencil $A - \lambda B$, with B positive definite, is transformed to a standard eigenvalue problem using a Cholesky decomposition $B = LL^T$. Because of the symmetry of the decomposition, the matrix $L^{-1}AL^{-T}$ inherits the symmetry of A. In the current context, if we introduce the skew-symmetric "inner product" $\langle x, y\rangle_J = y^T Jx$, we find that a matrix is Hamiltonian if and only if it is skew-symmetric with respect to this J-inner product, i.e., $\langle Hx, y\rangle_J = -\langle x, Hy\rangle_J$ for all $x, y \in \mathbf{R}^{2n}$. The relationship between $\mathcal{Z}_1$ and $\mathcal{Z}_2$ given in (1.3) implies that $\mathcal{Z}_2$ is (plus or minus) the adjoint of $\mathcal{Z}_1$ with respect to the J-inner product, i.e., $\langle \mathcal{Z}_1 x, y\rangle_J = \pm\langle x, \mathcal{Z}_2 y\rangle_J$ for all $x, y \in \mathbf{R}^{2n}$. Thus the decomposition $\mathcal{N} = \mathcal{Z}_1\mathcal{Z}_2$ is a symmetric, Cholesky-like decomposition of $\mathcal{N}$. Consequently, $\mathcal{W} = \mathcal{Z}_1^{-1}\mathcal{H}\mathcal{Z}_2^{-1}$ inherits the J-skew symmetry of $\mathcal{H}$; that is, $\mathcal{W}$ is Hamiltonian.

We discuss two approaches that make different transformations of the Hamiltonian operator $\mathcal{W}$. The first maps $\mathcal{W}$ to a skew-Hamiltonian operator, from which the eigenvalues can be extracted by an implicitly restarted Arnoldi (IRA) method that has been modified to preserve the structure. For this approach we provide numerical results demonstrating its effectiveness. The second approach maps $\mathcal{W}$ to a symplectic operator by a generalized Cayley transform. The desired eigenvalues can then be extracted by an implicitly restarted symplectic Lanczos method. We only outline this approach and discuss its advantages and disadvantages.

2. Applications. The need to solve SHH generalized eigenvalue problems arises in many applications. The best-known example is the linear quadratic optimal control problem for descriptor systems, where the pencil typically has the particular form

$$\alpha \begin{bmatrix} E & 0 \\ 0 & E^T \end{bmatrix} - \beta \begin{bmatrix} A & -BB^T \\ C^TC & -A^T \end{bmatrix}, \tag{2.1}$$

with B of size $n \times m$, C of size $p \times n$, and $m << n, p << n$; see [3, 4, 26]. Large sparse problems of this type arise, for example, in the control of semidiscretized parabolic partial differential equations [18, 31, 32].

Here the skew-Hamiltonian matrix can be written in factored form as

$$\mathcal{N} = \mathcal{Z}_1\mathcal{Z}_2 = \begin{bmatrix} E & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} I & 0 \\ 0 & E^T \end{bmatrix}, \tag{2.2}$$

with $\mathcal{Z}_2^T J = J\mathcal{Z}_1$.

The application that we will discuss in detail in this paper arises from quadratic eigenvalue problems of the form

$$\lambda^2 Mx + \lambda Gx + Kx = 0, \tag{2.3}$$

where $M = M^T$ is positive definite, $K = K^T$, and $G = -G^T$.

Large sparse eigenvalue problems of this form arise, for example, in finite element discretization in structural analysis [35] and in the elastic deformation of anisotropic materials [19, 21, 34]. In these applications M is a mass matrix, and $-K$ is a stiffness matrix. Depending on the applications, different parts of the spectrum are of interest; typically, one is interested in the eigenvalues with smallest real part or the eigenvalues smallest or largest in modulus.

At first glance the quadratic eigenvalue problem (2.3) and the SHH generalized eigenvalue problem (1.1) seem not to have much in common. However, it is well known that the eigenvalues of (2.3) occur in quadruplets $(\lambda, \bar{\lambda}, -\lambda, -\bar{\lambda})$ or real or purely imaginary pairs $(\lambda, -\lambda)$ [20], just as they do for (1.1). The reason for this similarity is that (2.3) can be "linearized" to have the form (1.1). This can be done in several different ways. For example, if we make the substitution

$$y = \lambda Mx \tag{2.4}$$

in (2.3) and rewrite the substitution as $\lambda x - M^{-1}y = 0$, we obtain the SHH generalized eigenvalue problem

$$\lambda\mathcal{N}z - \mathcal{H}z = \lambda \begin{bmatrix} I & G \\ 0 & I \end{bmatrix} \begin{bmatrix} y \\ x \end{bmatrix} - \begin{bmatrix} 0 & -K \\ M^{-1} & 0 \end{bmatrix} \begin{bmatrix} y \\ x \end{bmatrix} = 0. \tag{2.5}$$

Since $\mathcal{N}$ is invertible, the pencil $\lambda\mathcal{N} - \mathcal{H}$ is regular, i.e., $\det(\lambda\mathcal{N} - \mathcal{H})$ does not vanish identically.

The skew-Hamiltonian matrix $\mathcal{N}$ can be written in factored form as

$$\mathcal{N} = \mathcal{Z}_1\mathcal{Z}_2 = \begin{bmatrix} I & \frac{1}{2}G \\ 0 & I \end{bmatrix} \begin{bmatrix} I & \frac{1}{2}G \\ 0 & I \end{bmatrix}, \tag{2.6}$$

with $\mathcal{Z}_2^T J = J\mathcal{Z}_1$. We note that $\mathcal{Z}_1 = \mathcal{Z}_2$ is a skew-Hamiltonian square root of $\mathcal{N}$.

A second approach is to use the substitution

$$y = \lambda x \tag{2.7}$$

in (2.3) and rewrite the substitution as $\lambda Mx - My = 0$. This yields the SHH generalized eigenvalue problem

$$\lambda\mathcal{N}z - \mathcal{H}z = \lambda \begin{bmatrix} M & G \\ 0 & M \end{bmatrix} \begin{bmatrix} y \\ x \end{bmatrix} - \begin{bmatrix} 0 & -K \\ M & 0 \end{bmatrix} \begin{bmatrix} y \\ x \end{bmatrix} = 0. \tag{2.8}$$

In this case the skew Hamiltonian matrix $\mathcal{N}$ can be written in factored form as

$$\mathcal{N} = \mathcal{Z}_1\mathcal{Z}_2 = \begin{bmatrix} I & \frac{1}{2}G \\ 0 & M \end{bmatrix} \begin{bmatrix} M & \frac{1}{2}G \\ 0 & I \end{bmatrix}, \tag{2.9}$$

with $\mathcal{Z}_2^T J = J\mathcal{Z}_1$. This linearization has also been discussed in the context of Lanczos-like algorithms for quadratic eigenvalue problems in [14].

An approach that is intermediate to the previous two uses the Cholesky factorization $M = LL^T$ and the substitution

$$y = \lambda L^T x \tag{2.10}$$

in (2.3) to obtain the SHH generalized eigenvalue problem

$$\lambda\mathcal{N}z - \mathcal{H}z = \lambda \begin{bmatrix} L & G \\ 0 & L^T \end{bmatrix} \begin{bmatrix} y \\ x \end{bmatrix} - \begin{bmatrix} 0 & -K \\ I & 0 \end{bmatrix} \begin{bmatrix} y \\ x \end{bmatrix} = 0. \tag{2.11}$$

The skew-Hamiltonian matrix $\mathcal{N}$ can be written in the factored form

$$\mathcal{N} = \mathcal{Z}_1\mathcal{Z}_2 = \begin{bmatrix} I & \frac{1}{2}G \\ 0 & L^T \end{bmatrix} \begin{bmatrix} L & \frac{1}{2}G \\ 0 & I \end{bmatrix}, \tag{2.12}$$

with $\mathcal{Z}_2^T J = J\mathcal{Z}_1$.

It turns out that all three of these linearizations give rise to the same Hamiltonian operator $\mathcal{W} = \mathcal{Z}_1^{-1}\mathcal{H}\mathcal{Z}_2^{-1}$, so which one we use is just a matter of convenience.

If the matrix K is nonsingular, the substitution (2.7) together with the equivalent equation $Ky = \lambda Kx$ leads to the Hamiltonian/skew-Hamiltonian generalized eigenvalue problem

$$\lambda \begin{bmatrix} 0 & -M \\ K & 0 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} - \begin{bmatrix} K & G \\ 0 & K \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = 0. \tag{2.13}$$

We prefer not to use this linearization. If K is nearly singular, as is typically the case in practice, then the matrix pencil in (2.13) is close to a singular pencil [11].

Other linearizations that have been used are

$$\lambda \begin{bmatrix} I & 0 \\ G & M \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} - \begin{bmatrix} 0 & I \\ -K & 0 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = 0 \tag{2.14}$$

or, equivalently,

$$\lambda \begin{bmatrix} I & 0 \\ 0 & M \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} - \begin{bmatrix} 0 & I \\ -K & -G \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = 0, \tag{2.15}$$

which are both obtained with the substitution (2.7). Neither (2.14) nor (2.15) is SHH. Although, in particular, (2.15) seems attractive, since the matrix $\left[\begin{smallmatrix} I & 0 \\ 0 & M \end{smallmatrix}\right]$ is Hermitian positive definite, it is still preferable to use an SHH linearization, which reflects the structure of the spectrum. Since we construct methods that respect this structure also in finite arithmetic, the computed eigenvalues will be the exact eigenvalues of a perturbed SHH pencil. Thus the computed eigenvalues will occur in quadruplets, as they should. See the perturbation analysis in [3, 7, 8, 9]. Furthermore, the comparison of the perturbation theory for quadratic eigenvalue problems and the related linearization, as it has been analyzed in [41], show that even numerically backward stable methods that do not respect the special structure of the eigenvectors of the quadratic eigenvalue problem in its linearization may be unstable.

Surveys of numerical methods for the quadratic eigenvalue problem (in various forms) have been given in [23, 37, 38]. In particular, the Lanzcos and Arnoldi methods

[17, 29, 33] as well as the Jacobi–Davidson iterations [36] are compared. Simultaneous iteration methods for quadratic eigenvalue problems have been studied in [13, 28, 38].

Some of the methods can be directly applied to the quadratic problem, while others only work for the linearizations. The major difference between the direct solution of the quadratic problem and its linearizations is that in order to solve the projected problems and to use implicit restarts, one needs some kind of generalized Schur form for the problem. Such a form is not known for quadratic problems, while for matrix pencils this is the generalized Schur form [17], and for SHH pencils there is a structured generalized form that was analyzed in [24, 25]. Numerical methods for its computation have been derived in [3, 4].

3. Rational transformations of Hamiltonian matrices. If the skew-Hamiltonian matrix $\mathcal{N}$ in the SHH pencil $\alpha\mathcal{N} - \beta\mathcal{H}$ is invertible and given in the factored form $\mathcal{N} = \mathcal{Z}_1\mathcal{Z}_2$ with $\mathcal{Z}_2^T J = \pm J\mathcal{Z}_1$, then the pencil is equivalent to the pencil

$$\alpha I - \beta\mathcal{W} := \alpha I - \beta\mathcal{Z}_1^{-1}\mathcal{H}\mathcal{Z}_2^{-1}. \tag{3.1}$$

As we explained in the introduction, $\mathcal{W}$ is again Hamiltonian. Hence we can consider first transformations of Hamiltonian matrices, and then apply these to SHH pencils by rewriting them in terms of the original data.

Suppose we want to find the eigenvalues of $\mathcal{W}$ that lie nearest to some target value λ_0. The standard transformation for this purpose is $(\mathcal{W} - \lambda_0 I)^{-1}$, which, however, fails to preserve the structure. Each eigenvalue near λ_0 is related to eigenvalues near $\bar{\lambda}_0$, $-\lambda_0$, and $-\bar{\lambda}_0$. If we are to hope to preserve the structure, we must extract all four of these eigenvalues together. Thus we should also bring $(\mathcal{W} - \bar{\lambda}_0 I)^{-1}$, $(\mathcal{W} + \lambda_0 I)^{-1}$, and $(\mathcal{W} + \bar{\lambda}_0 I)^{-1}$ into play.

3.1. Transformation to a skew-Hamiltonian operator. For finding the eigenvalues nearest the quadruplet $(\lambda_0, \bar{\lambda}_0, -\lambda_0, -\bar{\lambda}_0)$ the obvious rational transformation is

$$R_1(\lambda_0, \mathcal{W}) = (\mathcal{W} - \lambda_0 I)^{-1}(\mathcal{W} + \lambda_0 I)^{-1}(\mathcal{W} - \bar{\lambda}_0 I)^{-1}(\mathcal{W} + \bar{\lambda}_0 I)^{-1}. \tag{3.2}$$

If the target λ_0 is either real or purely imaginary, one may choose to use the simpler transformation

$$R_2(\lambda_0, \mathcal{W}) = (\mathcal{W} - \lambda_0 I)^{-1}(\mathcal{W} + \lambda_0 I)^{-1}. \tag{3.3}$$

Both R_1 and R_2 turn out to be skew-Hamiltonian. In what follows we will focus on the operator R_1. In every case we can make similar assertions about R_2.

The transformation $R_1(\lambda_0, \mathcal{W})$ maps all eigenvalues near to $\lambda_0, -\lambda_0, \bar{\lambda}_0, -\bar{\lambda}_0$ simultaneously to values of large modulus. Hence fast convergence of the eigenvalues near these points can be expected if we apply any of the standard iterative methods, [33, 39].

In order to apply an iterative method such as subspace iteration or the Arnoldi process, we need to be able to multiply the matrix R_1 by an arbitrary vector at reasonable cost, since this operation is performed repeatedly by these algorithms. Thus we need to be able to apply operators of the form $(\mathcal{W} - \sigma I)^{-1}$ inexpensively. We can either work with $\mathcal{W}$ directly or we can refer back to the original data $\mathcal{H}$ and $\mathcal{N}$. Substituting $\mathcal{Z}_1^{-1}\mathcal{H}\mathcal{Z}_2^{-1}$ for $\mathcal{W}$ in (3.2) and simplifying, we find that

$$\begin{aligned} R_1 &= R_1(\lambda_0, \mathcal{H}, \mathcal{N}, \mathcal{Z}_1, \mathcal{Z}_2) \\ &= \mathcal{Z}_2(\mathcal{H} - \lambda_0\mathcal{N})^{-1}\mathcal{N}(\mathcal{H} + \lambda_0\mathcal{N})^{-1}\mathcal{N}(\mathcal{H} - \bar{\lambda}_0\mathcal{N})^{-1}\mathcal{N}(\mathcal{H} + \bar{\lambda}_0\mathcal{N})^{-1}\mathcal{Z}_1. \end{aligned} \tag{3.4}$$

Since $\mathcal{Z}_1$, $\mathcal{Z}_2$, and $\mathcal{N}$ are sparse matrices, they can be applied easily. The only question, then, is how to apply operators of the form $(\mathcal{H}-\sigma\mathcal{N})^{-1}$ inexpensively. This question will be discussed in connection with specific applications in section 4.

We now consider the structural properties of $R_1(\lambda_0,\mathcal{W})$ and $R_2(\lambda_0,\mathcal{W})$, beginning with a well-known lemma.

LEMMA 3.1.

(i) *If $\mathcal{W}$ is Hamiltonian, then $\mathcal{W}^2$ is skew-Hamiltonian.*

(ii) *If $\mathcal{W}$ is skew-Hamiltonian, then $\mathcal{W}^2$ is skew-Hamiltonian.*

(iii) *If $\mathcal{W}$ is skew-Hamiltonian and invertible, then $\mathcal{W}^{-1}$ is skew-Hamiltonian.*

We omit the proof, which is straightforward, see, e.g., [1].

PROPOSITION 3.2. *If $\mathcal{W}\in\mathbf{R}^{2n\times 2n}$ is a Hamiltonian matrix, then $R_1(\lambda_0,\mathcal{W})$ in* (3.2) *is real and skew-Hamiltonian. If, in addition, λ_0 is either real or purely imaginary, then $R_2(\lambda_0,\mathcal{W})$ in* (3.3) *is also real and skew-Hamiltonian.*

Proof. Direct calculation shows that

$$R_1(\lambda_0,\mathcal{W}) = (|\lambda_0|^4 I + [2|\lambda_0|^2 - 4(\Re(\lambda_0))^2]\mathcal{W}^2 + \mathcal{W}^4)^{-1}.$$

Hence $R_1(\lambda_0,\mathcal{W})$ is a real matrix, and by Lemma 3.1 it is skew-Hamiltonian, since the skew-Hamiltonian matrices form a real vector space that contains the identity matrix.

A similar but simpler argument shows that $R_2(\lambda_0,\mathcal{W})$ is skew-Hamiltonian. ∎

Notice that the eigenvalues λ and $-\lambda$ of $\mathcal{W}$ both get mapped to $\mu=(\lambda^4+c_2\lambda^2+c_0)^{-1}$ of R_1 and to $\nu=(\lambda^2-\lambda_0^2)^{-1}$ of R_2, where $c_0=|\lambda_0|^4$ and $c_2=2|\lambda_0|^2-4(\Re(\lambda_0))^2$. Thus all eigenvalues of R_1 and R_2 have even multiplicity. This fact, which is true of skew-Hamiltonian matrices in general, must be taken into account by our numerical methods.

Recall that the *Krylov subspace* $\mathcal{K}_j(A,v)$ is defined by

$$\mathcal{K}_j(A,v) = \operatorname{span}\{v, Av, A^2v, \ldots, A^{j-1}v\}.$$

The Arnoldi method and other Krylov subspace methods generate Krylov subspaces, where v is the starting vector and A is the operator being applied, e.g., R_1 in our case; see [33]. Recall, furthermore, that a subspace $\mathcal{S}$ of $\mathbf{R}^{2n}$ is called *isotropic* if and only if $y^TJx=0$ for all $x,\ y\in\mathcal{S}$. In other words, $\mathcal{S}$ is isotropic if and only if $J\mathcal{S}$ is orthogonal to $\mathcal{S}$ with respect to the standard inner product on $\mathbf{R}^{2n}$; see [2]. The beauty of skew-Hamiltonian operators is that the Krylov subspaces that they generate are isotropic.

PROPOSITION 3.3. *Let $A\in\mathbf{R}^{2n\times 2n}$ be a skew-Hamiltonian matrix, let $v\in\mathbf{R}^{2n}$, and let j be a positive integer. Then the Krylov subspace $\mathcal{K}_j(A,v)$ is isotropic.*

Proof. Since A is real skew-Hamiltonian, one easily shows that all of its powers A^i, $i=0,1,2,3,\ldots$, are also skew-Hamiltonian. This means that JA^i is skew-symmetric. To establish isotropy of $\mathcal{K}_j(A,v)$, it suffices, by linearity, to prove that $(A^iv)^TJA^kv=0$ for all $i\ge 0$ and $k\ge 0$. Since $A^TJ=JA$, we have $(A^iv)^TJA^kv=v^TJA^{i+k}v$, which equals zero because JA^{i+k} is skew-symmetric. ∎

We now introduce an *isotropic* Arnoldi process, which will form the basis for our numerical method. At first we allow A to be an arbitrary $2n\times 2n$ real matrix. Recall that the Arnoldi process starts with an arbitrary unit vector q_1 and produces orthonormal vectors as follows. Given orthonormal vectors $q_1,\ldots,q_j$, the next vector q_{j+1} is generated by forming Aq_j and then orthogonalizing it against $q_1,\ldots,q_j$. Thus

$$q_{j+1}h_{j+1,j} = Aq_j - \sum_{i=1}^{j} q_ih_{ij},$$

where $h_{ij} = q_i^T A q_j$, $i = 1, \ldots, j$, and $h_{j+1,j}$ is a positive constant chosen so that $\|q_{j+1}\|_2 = 1$.

If we wish to build isotropic subspaces, we should orthogonalize against the vectors $Jq_1, \ldots, Jq_j$ as well. Thus the jth step of the *isotropic Arnoldi process* is

$$q_{j+1} h_{j+1,j} = A q_j - \sum_{i=1}^{j} q_i h_{ij} - \sum_{i=1}^{j} J q_i t_{ij}, \tag{3.5}$$

where

$$h_{ij} = q_i^T A q_j \quad \text{and} \quad t_{ij} = (Jq_i)^T A q_j, \tag{3.6}$$

and $h_{j+1,j}$ is a positive constant chosen so that $\|q_{j+1}\|_2 = 1$. This generates orthonormal vectors that span isotropic subspaces.

If A is real skew-Hamiltonian, then the coefficients t_{ij} in (3.5, 3.6) will all be zero, by Proposition 3.3. Thus the isotropic Arnoldi process reduces to the ordinary Arnoldi process in this case, at least in theory.

If the quantity on the right-hand side of (3.5) is zero, then the process breaks down. It can be continued by setting $h_{j+1,j} = 0$ and taking q_{j+1} to be any unit vector that is orthogonal to $q_1, \ldots, q_j$ and $Jq_1, \ldots, Jq_j$. Equation (3.5) holds in this case, too. In exact arithmetic, the process terminates after $n-1$ steps, as $q_1, \ldots, q_n$ together with $Jq_1, \ldots, Jq_n$ form an orthonormal basis of $\mathbf{R}^{2n}$.

Let $Q \in \mathbf{R}^{2n \times n}$ be the matrix whose columns are $q_1, \ldots, q_n$. Then (3.5) can be rewritten as

$$AQ = \begin{bmatrix} Q & -JQ \end{bmatrix} \begin{bmatrix} H \\ -T \end{bmatrix}, \tag{3.7}$$

where H is an upper Hessenberg matrix built from the coefficients h_{ij} and T is an upper triangular matrix built from the coefficients t_{ij}.

Recall that a matrix $S \in \mathbf{R}^{2n \times 2n}$ is *symplectic* if $S^T J S = J$. The next lemma recalls a few simple properties that we need; see [2].

LEMMA 3.4.

(i) *If the columns of $Q \in \mathbf{R}^{2n \times n}$ are orthonormal and span an isotropic subspace, then the matrix $\begin{bmatrix} Q & -JQ \end{bmatrix}$ is both orthogonal and symplectic.*

(ii) *If $\mathcal{N}$ is skew-Hamiltonian and S is symplectic, then $S^{-1}\mathcal{N}S$ is skew-Hamiltonian.*

Equation (3.7) implies that

$$A \begin{bmatrix} Q & -JQ \end{bmatrix} = \begin{bmatrix} Q & -JQ \end{bmatrix} \begin{bmatrix} H & N \\ -T & P \end{bmatrix} \tag{3.8}$$

for some N and P. The matrix $\mathcal{U} = \begin{bmatrix} Q & -JQ \end{bmatrix}$ is orthogonal and symplectic by Lemma 3.4, and we have

$$\mathcal{U}^{-1} A \mathcal{U} = \begin{bmatrix} H & N \\ -T & P \end{bmatrix}.$$

If A is skew-Hamiltonian, then, since the skew-Hamiltonian property is preserved under symplectic similarity by Lemma 3.4, we must have $P = H^T$, $T^T = -T$, and $N^T = -N$. Since T is upper triangular, the additional equation $T^T = -T$ implies

$T = 0$, which we had already noted. This gives the following result, which is not new [42].

PROPOSITION 3.5. *Let $\mathcal{N}$ be a real skew-Hamiltonian matrix. Then for every unit vector $q_1 \in \mathcal{R}^{2n}$ there exists an orthogonal symplectic matrix $\mathcal{U}$ that has q_1 as its first column such that*

$$\mathcal{U}^T \mathcal{N} \mathcal{U} = \begin{bmatrix} H & N \\ 0 & H^T \end{bmatrix}, \tag{3.9}$$

and H is in Hessenberg form.

The multiplicity of the eigenvalues of $\mathcal{N}$ is reflected in the structure of the matrix in (3.9). For each double eigenvalue of $\mathcal{N}$, one copy resides in H, and the other copy is in H^T.

When we apply the Arnoldi process to a large, sparse matrix in practice, we have neither the time nor the storage space to carry the process to completion. Instead we stop after k steps with $k \ll n$. The coefficients h_{ij} computed to this point form a $k \times k$ submatrix of H whose eigenvalues (*Ritz values*) we can compute and use as estimates of eigenvalues of H, and hence of $\mathcal{N}$. Since the eigenvalues do not appear in duplicate in H, we get each eigenvalue once, not twice. This is important to the success of the numerical method. If we make the effort to compute, say, six eigenvalues, we would like to get six distinct eigenvalues, not three eigenvalues in duplicate.

The developments outlined here are valid only for real matrices. However, it has been shown in [10] that for complex problems an embedding of the problem into a double sized real problem can be used to get a real skew-Hamiltonian problem from which all the spectral information can be obtained.

3.2. Transformation to symplectic form. Another natural rational transformation that is sometimes applied to Hamiltonian matrices is the generalized Cayley transform. See [27] for a detailed analysis. Given a target λ_0, the generalized Cayley transform of the Hamiltonian matrix $\mathcal{W}$ is

$$S_1(\lambda_0, \mathcal{W}) = (\mathcal{W} - \lambda_0 I)^{-1}(\mathcal{W} + \bar{\lambda}_0 I)(\mathcal{W} - \bar{\lambda}_0 I)^{-1}(\mathcal{W} + \lambda_0 I). \tag{3.10}$$

By using four factors we obtain a real matrix, even when the target λ_0 is complex. If λ_0 is real, we can use the simpler two-factor Cayley transform

$$S_2(\lambda_0, \mathcal{W}) = (\mathcal{W} - \lambda_0 I)^{-1}(\mathcal{W} + \lambda_0 I). \tag{3.11}$$

Notice that for $i = 1,\ 2$, $S_i^{-1}(\lambda) = S_i(-\lambda)$. Thus it is no more expensive to apply S_i^{-1} than S_i.

In terms of the original data we have

$$\begin{aligned} S_1 &= S_1(\lambda_0, \mathcal{H}, \mathcal{N}, \mathcal{Z}_1, \mathcal{Z}_2) \\ &= \mathcal{Z}_2(\mathcal{H} - \lambda_0 \mathcal{N})^{-1}(\mathcal{H} + \bar{\lambda}_0 \mathcal{N})(\mathcal{H} - \bar{\lambda}_0 \mathcal{N})^{-1}(\mathcal{H} + \lambda_0 \mathcal{N})\mathcal{Z}_2^{-1}. \end{aligned} \tag{3.12}$$

A similar expression holds for S_2.

We now discuss the structural properties of S_1 and S_2. The following proposition is well known [27].

PROPOSITION 3.6.

(i) *If $\mathcal{W} \in \mathbf{R}^{2n \times 2n}$ is Hamiltonian, then $S_1(\lambda_0, \mathcal{W})$ in* (3.10) *is real symplectic.*

(ii) *If $\mathcal{W} \in \mathbf{R}^{2n \times 2n}$ is Hamiltonian and λ_0 is real, then $S_2(\lambda_0, \mathcal{W})$ in* (3.11) *is real symplectic.*

The use of symplectic matrices has several advantages and disadvantages compared with skew-Hamiltonian matrices. First of all, in contrast to the skew-Hamiltonian case, not every symplectic matrix has a symplectic Schur form [22]. What is worse in the symplectic case is that in order to achieve a symplectic Hessenberg-like form with an orthogonal symplectic transformation matrix Q, we must choose a first column of Q that has a very special form [2]. Thus Arnoldi methods, with or without restarts, cannot be used with symplectic matrices, which is definitely a disadvantage of this rational transformation.

The transformation S_1 maps all eigenvalues near $\lambda_0, \bar{\lambda}_0$ simultaneously to values of large modulus. At the same time the eigenvalues near $-\lambda_0, -\bar{\lambda}_0$ are mapped close to 0. These correspond to large eigenvalues of the inverse transformation S_1^{-1}. The coexistence of extremely large and small eigenvalues is an inevitable consequence of the fact that eigenvalues of a symplectic matrix occur in quadruplets $\mu, \bar{\mu}, \mu^{-1}, \bar{\mu}^{-1}$. Any method that preserves the symplectic structure must extract quadruplets intact. This can be achieved by structure-preserving Lanczos-like methods that employ both S and S^{-1} in a symmetric manner. See [5, 6, 15] for structure-preserving implicitly restarted Lanczos-like methods applicable to symplectic problems. These are the sorts of methods we must apply to S_1 and S_2.

Another disadvantage of the Cayley transform is that it is effective only if the target λ_0 is not too close to the imaginary axis. Notice that if λ_0 is purely imaginary, then $S_1 = I$, which is not useful for extracting spectral information about $\mathcal{W}$.

An advantage of the Cayley transform approach is that, unlike the skew-Hamiltonian approach, it can be extended to complex matrices and matrix pencils in a straightforward way.

4. Applying the operators. In our two applications the matrices $\mathcal{N}$, $\mathcal{H}$, $\mathcal{Z}_1$, and $\mathcal{Z}_2$ have further structure that can be used to simplify the formulas.

4.1. Quadratic eigenproblems. Let us first study the quadratic eigenvalue problem in the linearization (2.8) with $\mathcal{N}$ in factored form (2.9). First,

$$\begin{aligned}\mathcal{H} - \sigma\mathcal{N} &= -\begin{bmatrix} \sigma M & \sigma G + K \\ -M & \sigma M \end{bmatrix} \\ &= \begin{bmatrix} I & -\sigma I \\ 0 & I \end{bmatrix} \begin{bmatrix} 0 & -Q(\sigma) \\ M & 0 \end{bmatrix} \begin{bmatrix} I & -\sigma I \\ 0 & I \end{bmatrix}, \end{aligned} \tag{4.1}$$

where

$$Q(\sigma) := \sigma^2 M + \sigma G + K. \tag{4.2}$$

Thus

$$\begin{aligned}(\mathcal{W} - \sigma I)^{-1} &= \mathcal{Z}_2(\mathcal{H} - \sigma\mathcal{N})^{-1}\mathcal{Z}_1 \\ &= \begin{bmatrix} M & \frac{1}{2}G \\ 0 & I \end{bmatrix} \begin{bmatrix} I & \sigma I \\ 0 & I \end{bmatrix} \begin{bmatrix} 0 & M^{-1} \\ -Q(\sigma)^{-1} & 0 \end{bmatrix} \begin{bmatrix} I & \sigma I \\ 0 & I \end{bmatrix} \begin{bmatrix} I & \frac{1}{2}G \\ 0 & M \end{bmatrix}. \end{aligned} \tag{4.3}$$

We can obtain an expression that does not involve M^{-1} by using one or the other of the factorizations

$$\begin{bmatrix} 0 & M^{-1} \\ -Q(\sigma)^{-1} & 0 \end{bmatrix} = \begin{bmatrix} 0 & I \\ -Q(\sigma)^{-1} & 0 \end{bmatrix} \begin{bmatrix} I & 0 \\ 0 & M^{-1} \end{bmatrix} \tag{4.4}$$

and

$$\begin{bmatrix} 0 & M^{-1} \\ -Q(\sigma)^{-1} & 0 \end{bmatrix} = \begin{bmatrix} M^{-1} & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} 0 & I \\ -Q(\sigma)^{-1} & 0 \end{bmatrix}. \tag{4.5}$$

Substituting (4.4) into (4.3) and combining the last three matrices, we obtain

$$(\mathcal{W} - \sigma I)^{-1} = \begin{bmatrix} M & \frac{1}{2}G \\ 0 & I \end{bmatrix} \begin{bmatrix} I & \sigma I \\ 0 & I \end{bmatrix} \begin{bmatrix} 0 & I \\ -Q(\sigma)^{-1} & 0 \end{bmatrix} \begin{bmatrix} I & \frac{1}{2}G + \sigma M \\ 0 & I \end{bmatrix}. \tag{4.6}$$

If we use (4.5) instead of (4.4), we obtain instead

$$(\mathcal{W} - \sigma I)^{-1} = \begin{bmatrix} I & \frac{1}{2}G + \sigma M \\ 0 & I \end{bmatrix} \begin{bmatrix} 0 & I \\ -Q(\sigma)^{-1} & 0 \end{bmatrix} \begin{bmatrix} I & \sigma I \\ 0 & I \end{bmatrix} \begin{bmatrix} I & \frac{1}{2}G \\ 0 & M \end{bmatrix}. \tag{4.7}$$

Both of these expressions are useful.

To apply this operator in either form (4.6) or (4.7) we will need a sparse LU decomposition of the sparse, nonsymmetric matrix $Q(\sigma)$ [17]. In $R_1(\lambda_0, \mathcal{W})$ there are four factors of the form $(\mathcal{W} - \sigma I)^{-1}$, corresponding to $\sigma = \pm\lambda_0$ and $\sigma = \pm\bar{\lambda}_0$. Thus it might appear that we need four sparse LU factorizations in order to apply R_1. However, one immediately observes that

$$\begin{aligned} Q(\bar{\sigma}) &= \bar{\sigma}^2 M + \bar{\sigma} G + K = \overline{Q(\sigma)}, \\ Q(-\sigma) &= \sigma^2 M - \sigma G + K = Q(\sigma)^T, \\ Q(-\bar{\sigma}) &= \bar{\sigma}^2 M - \bar{\sigma} G + K = \overline{Q(\sigma)}^T. \end{aligned} \tag{4.8}$$

An LU factorization of $Q(\sigma)$ immediately yields factorizations of $Q(\bar{\sigma})$, $Q(-\sigma)$, and $Q(-\bar{\sigma})$. Thus one sparse LU factorization is all we need.

There are several other ways to apply $(W - \sigma I)^{-1}$ efficiently to a real vector for real W and complex σ. One possibility is described in [30]. Another possibility is to embed the matrix in a double sized real matrix and then use a real factorization. Which of these different approaches is best will depend on the structure of the matrices G, K, amd M and also on the choice of shifts.

To derive the formulas for applying the operator R_1, we first discuss the application of R_2. If we combine expressions (4.6) and (4.7), replacing σ by λ_0 in (4.6) and by $-\lambda_0$ in (4.7), we obtain

$$\begin{aligned} R_2(\lambda_0, \mathcal{W}) &= (\mathcal{W} - \lambda_0 I)^{-1} (\mathcal{W} + \lambda_0 I)^{-1} \\ &= \begin{bmatrix} M & \frac{1}{2}G \\ 0 & I \end{bmatrix} \begin{bmatrix} I & \lambda_0 I \\ 0 & I \end{bmatrix} \begin{bmatrix} 0 & I \\ -Q(\lambda_0)^{-1} & 0 \end{bmatrix} \begin{bmatrix} I & G \\ 0 & I \end{bmatrix} \\ &\quad \times \begin{bmatrix} 0 & I \\ -Q(-\lambda_0)^{-1} & 0 \end{bmatrix} \begin{bmatrix} I & -\lambda_0 I \\ 0 & I \end{bmatrix} \begin{bmatrix} I & \frac{1}{2}G \\ 0 & M \end{bmatrix}, \end{aligned} \tag{4.9}$$

which can be readily translated into an algorithm for applying the operator R_2, i.e., for multiplying R_2 by a vector. For example, to multiply the factor $\left[\begin{smallmatrix} I & \frac{1}{2}G \\ 0 & M \end{smallmatrix}\right]$ by the vector $\left[\begin{smallmatrix} x \\ y \end{smallmatrix}\right]$, we only need to perform the operations $x \leftarrow x + \frac{1}{2}Gy$ and $y \leftarrow My$, which requires two sparse $n \times n$ matrix-vector products and one saxpy operation. To multiply $\left[\begin{smallmatrix} 0 & I \\ -Q(-\lambda_0)^{-1} & 0 \end{smallmatrix}\right]$ by $\left[\begin{smallmatrix} x \\ y \end{smallmatrix}\right]$, we perform the operations $\hat{x} \leftarrow -x$, $x \leftarrow y$, and $y \leftarrow Q(\lambda_0)^{-1}\hat{x}$. Given a sparse LU factorization of $Q(\lambda_0)$, we obtain $y \leftarrow Q(\lambda_0)^{-1}\hat{x}$ by

two sparse triangular solves. Applying these observations to (4.9), we easily determine the total work for multiplying $R_2(\lambda_0, \mathcal{W})$ by a vector.

The sparse LU factorization of $Q(\lambda_0)$ needs to be computed only once; then R_2 can be applied repeatedly. No further LU factorizations are needed unless the target λ_0 is changed. If λ_0 is real, then all operations are real and we may use R_2 instead of R_1.

Since $R_1(\lambda_0, \mathcal{W}) = R_2(\lambda_0, \mathcal{W})R_2(\bar{\lambda}_0, \mathcal{W})$, we obtain an expression for $R_1(\lambda_0, \mathcal{W})$ by combining two copies of (4.9), one with λ_0 replaced by $\bar{\lambda}_0$. Combining two matrices in the middle of the product, we eliminate one sparse matrix-vector product and one saxpy operation. Thus the cost of applying R_1 is just slightly less than twice that of applying R_2. We summarize the costs in Table 4.1.

TABLE 4.1
Operation count for applying the operator R_1, assuming a sparse LU factorization of $Q(\lambda_0)$ is available.

8	Sparse triangular solves
4	Symmetric sparse matrix-vector products Mz
5	Skew symmetric sparse matrix-vector products Gz
9	Saxpy operations

In general the arithmetic is complex; however, if λ_0 is real, then all operations are real.

If λ_0 is neither real nor purely imaginary, there is a trick that halves the cost of computing $R_1(\lambda_0, \mathcal{W})$ [30]. In this case one easily verifies that $R_1(\lambda_0, \mathcal{W})$ is proportional to the imaginary part of $R_2(\lambda_0, \mathcal{W})$, so we can obtain $R_1(\lambda_0, \mathcal{W})$ at the costs listed in Table 4.2 (with complex arithmetic) by simply computing $R_2(\lambda_0, \mathcal{W})$ and keeping the imaginary part.

TABLE 4.2
Operation count for applying the operator R_2, assuming a sparse LU factorization of $Q(\lambda_0)$ is available.

4	Sparse triangular solves
2	Symmetric sparse matrix-vector products Mz
3	Skew symmetric sparse matrix-vector products Gz
5	Saxpy operations

The symplectic operators S_1 and S_2 can be applied similarly. Inverting (4.7) and replacing σ by $-\sigma$, we obtain

$$(\mathcal{W} + \sigma I) = \begin{bmatrix} I & \frac{1}{2}G \\ 0 & M \end{bmatrix}^{-1} \begin{bmatrix} I & \sigma I \\ 0 & I \end{bmatrix} \begin{bmatrix} 0 & -Q(-\sigma) \\ I & 0 \end{bmatrix} \begin{bmatrix} I & \sigma M - \frac{1}{2}G \\ 0 & I \end{bmatrix}. \tag{4.10}$$

Now combining (4.7) with (4.10) and replacing σ by λ_0, we get the expression

$$\begin{aligned} S_2(\lambda_0, \mathcal{W}) &= (\mathcal{W} - \lambda_0 I)^{-1}(\mathcal{W} + \lambda_0 I) \\ &= \begin{bmatrix} I & \lambda_0 M + \frac{1}{2}G \\ 0 & I \end{bmatrix} \begin{bmatrix} 0 & I \\ -Q(\lambda_0)^{-1} & 0 \end{bmatrix} \begin{bmatrix} I & 2\lambda_0 I \\ 0 & I \end{bmatrix} \\ &\quad \times \begin{bmatrix} 0 & -Q(-\lambda_0) \\ I & 0 \end{bmatrix} \begin{bmatrix} I & \lambda_0 M - \frac{1}{2}G \\ 0 & I \end{bmatrix}, \end{aligned} \tag{4.11}$$

which can be readily translated into an algorithm for applying the operator S_2. The total work for multiplying $S_2(\lambda_0, \mathcal{W})$ by a vector is summarized in Table 4.3.

TABLE 4.3
Operation count for applying the operator S_2, assuming a sparse LU factorization of $Q(\lambda_0)$ is available.

2	Sparse triangular solves
3	Symmetric sparse matrix-vector products Mz
3	Skew symmetric sparse matrix-vector products Gz
1	Symmetric sparse matrix-vector product Kz
7	Saxpy operations

We have counted the operation $Q(-\lambda_0)v$ as three matrix-vector products, one each by M, G, and K, and two saxpy operations. The work can be decreased by computing $Q(-\lambda_0)$ in advance and storing it if there is sufficient storage space. If λ_0 is real, then all operations are real.

Juxtaposing two copies of (4.11), we obtain the following expression for $S_1(\lambda_0, \mathcal{W})$:

$$
\begin{aligned}
S_1(\lambda_0,\mathcal{W}) &= (\mathcal{W}-\lambda_0 I)^{-1}(\mathcal{W}+\lambda_0 I)(\mathcal{W}-\bar\lambda_0 I)^{-1}(\mathcal{W}+\bar\lambda_0 I)\\
&= \begin{bmatrix} I & \lambda_0 M+\frac{1}{2}G \\ 0 & I \end{bmatrix}\begin{bmatrix} 0 & I \\ -Q(\lambda_0)^{-1} & 0 \end{bmatrix}\begin{bmatrix} I & 2\lambda_0 I \\ 0 & I \end{bmatrix}\\
&\quad\times\begin{bmatrix} 0 & -Q(-\lambda_0) \\ I & 0 \end{bmatrix}\begin{bmatrix} I & 2\Re(\lambda_0)M \\ 0 & I \end{bmatrix}\begin{bmatrix} 0 & I \\ -Q(\bar\lambda_0)^{-1} & 0 \end{bmatrix}\\
&\quad\times\begin{bmatrix} I & 2\bar\lambda_0 I \\ 0 & I \end{bmatrix}\begin{bmatrix} 0 & -Q(-\bar\lambda_0) \\ I & 0 \end{bmatrix}\begin{bmatrix} I & \bar\lambda_0 M-\frac{1}{2}G \\ 0 & I \end{bmatrix},
\end{aligned}
\tag{4.12}
$$

which can be translated into an algorithm for applying the operator S_1. The total work for multiplying $S_1(\lambda_0, \mathcal{W})$ by a vector is summarized in Table 4.4.

TABLE 4.4
Operation count for applying the operator S_1, assuming a sparse LU factorization of $Q(\lambda_0)$ is available.

4	Sparse triangular solves
5	Symmetric sparse matrix-vector products Mz
4	Skew symmetric sparse matrix-vector products Gz
2	Symmetric sparse matrix-vector products Kz
11	Saxpy operations

In general the arithmetic is complex; however, if λ_0 is real, then all operations are real.

Typically, the triangular LU factors are sparse, but not nearly as sparse as the original matrices. Thus the most expensive operations listed in Tables 4.2–4.4 are the sparse triangular solves. The matrix-vector products are cheaper, but still significant. The saxpy operations are relatively insignificant. Bearing these facts in mind and comparing Tables 4.1 and 4.4, we conclude that the symplectic operator S_1 can be applied significantly less expensively than R_1 can.

4.2. Optimal control problems. Consider the SHH pencil

$$
\alpha\begin{bmatrix} E & 0 \\ 0 & E^T \end{bmatrix} - \beta\begin{bmatrix} A & -BB^T \\ C^TC & -A^T \end{bmatrix}
\tag{4.13}
$$

from the linear quadratic optimal control problems for descriptor systems, and consider the decomposition of $\mathcal{N}$ given by (2.2). We have

$$
\begin{aligned}
R_1 &= R_1(\lambda_0,\mathcal{H},\mathcal{N},\mathcal{Z}_1,\mathcal{Z}_2)\\
&= \mathcal{Z}_2(\mathcal{H}-\lambda_0\mathcal{N})^{-1}\mathcal{N}(\mathcal{H}+\lambda_0\mathcal{N})^{-1}\mathcal{N}(\mathcal{H}-\bar\lambda_0\mathcal{N})^{-1}\mathcal{N}(\mathcal{H}+\bar\lambda_0\mathcal{N})^{-1}\mathcal{Z}_1
\end{aligned}
$$

from (3.4). In this case each of the terms $(\mathcal{H}-\sigma\mathcal{N})^{-1}$ can be factored as

$$(\mathcal{H}-\sigma\mathcal{N})^{-1} = \begin{bmatrix} A-\sigma E & -BB^T \\ C^TC & -\left(A^T+\sigma E^T\right) \end{bmatrix}^{-1}$$

$$(4.14) = \begin{bmatrix} F^{-1} & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} I & BB^T \\ 0 & I \end{bmatrix} \begin{bmatrix} F & 0 \\ 0 & D^{-1} \end{bmatrix} \begin{bmatrix} I & 0 \\ -C^TC & I \end{bmatrix} \begin{bmatrix} F^{-1} & 0 \\ 0 & I \end{bmatrix},$$

where $F = A-\sigma E$ and $D = -\left(A^T+\sigma E^T\right)+C^TCF^{-1}BB^T$. To apply this operator we will certainly need an LU factorization of $A-\sigma E$. We will also need an LU factorization of $A^T+\sigma E^T$, which we will use together with the Sherman–Morrison–Woodbury formula [17] to evaluate D^{-1}.

We have to apply (4.14) with $\sigma = \lambda_0$, $\bar{\lambda}_0$, $-\lambda_0$, and $-\bar{\lambda}_0$. If we have sparse LU factorizations associated with one choice of σ, then we automatically get the other factorizations. For example, if we have LU factorizations $A-\lambda_0 E = L_1U_1$ and $A^T+\lambda_0E^T = L_2U_2$, then we also have

$$(4.15) \qquad \begin{array}{llll} A-\bar{\lambda}_0E &= \bar{L}_1\bar{U}_1, & A^T+\bar{\lambda}_0E^T &= \bar{L}_2\bar{U}_2, \\ A+\lambda_0E &= U_2^TL_2^T, & A^T-\lambda_0E^T &= U_1^TL_1^T, \\ A+\bar{\lambda}_0E &= \bar{U}_2^T\bar{L}_2^T, & A^T-\bar{\lambda}_0E^T &= \bar{U}_1^T\bar{L}_1^T. \end{array}$$

To apply D^{-1} efficiently, we exploit the fact that the term $C^TCF^{-1}BB^T$ has low rank. Recalling that m and p are the number of columns in B and C^T, respectively, let us assume that $p \le m$. (Otherwise, exchange the roles of B and C.) Then, with $A-\sigma E = T_1V_1$, $-\left(A^T+\sigma E^T\right) = T_2V_2$ being any of the factorizations listed above, we can apply the Sherman–Morrison–Woodbury formula [17] to obtain

$$\begin{aligned} D^{-1} &= (T_2V_2 + C^TCV_1^{-1}T_1^{-1}BB^T)^{-1} \\ &= V_2^{-1}(I - T_2^{-1}C^TN^{-1}CV_1^{-1}T_1^{-1}BB^TV_2^{-1})T_2^{-1}, \end{aligned} \qquad (4.16)$$

where $N = I + CV_1^{-1}T_1^{-1}BB^TV_2^{-1}T_2^{-1}C^T$ is by assumption a very small matrix $(p\times p)$ and typically full. The cost to compute N consists of $4p$ triangular solves plus $4nmp + 2mp^2$ flops (much less if B is sparse). An additional $2p^3$ flops are needed to invert this small matrix. This part of the computation must be done four times, once for each choice of σ.

With N^{-1} available we see from (4.16) that the cost of multiplying D^{-1} by a vector is essentially that of six sparse triangular solves. The other operations, such as multiplication of B by a vector, are relatively insignificant if B and C are sparse. Thus the cost of applying $(\mathcal{H}-\sigma\mathcal{N})^{-1}$ in (4.14) is 10 sparse triangular solves plus small change, and the cost of applying R_1 in (3.4) is 40 sparse triangular solves plus change. Similarly, the cost of applying S_1 (3.12) is a bit more than 20 sparse triangular solves.

4.3. Skew-Hamiltonian versus symplectic operators. We have already discussed some of the advantages and disadvantages of the two rational transformations that lead to skew-Hamiltonian and symplectic operators, respectively.

The advantage of the skew-Hamiltonian operator is that we can apply the IRA method, while for the symplectic operator we have to use special symplectic Lanczos methods with all their possible stability problems due to breakdowns or near-breakdowns. Due to the possible breakdowns, these latter methods are also more difficult to implement in practice; see [16]. Furthermore, as we have discussed, the

symplectic operator cannot be used when the target shift is close to or on the imaginary axis, which is a frequent situation in optimal control problems.

But the symplectic operators also have advantages, since the cost of applying the operator is roughly half of that for the skew-Hamiltonian operator, and the method is also applicable for complex pencils. It should be noted, though, that in the symplectic Lanczos method we need to apply both the operator and its inverse at each step.

Taking this comparison into account, we have implemented the IRA method based on the skew-Hamiltonian operator for its simplicity and numerical robustness, and since we can also use it with shifts near or on the imaginary axis.

5. Implementation of the skew-Hamiltonian Arnoldi method. We implemented a modified version of the IRA method [39] that applies the skew-Hamiltonian operator R_1 (3.4). Since our method is a skew-Hamiltonian, isotropic, IRA method, we call it SHIRA.

In order to preserve the structure, instead of a standard Arnoldi step we use the isotropic Arnoldi step (3.5), (3.6) with $A = R_1$. In exact arithmetic the coefficients t_{ij} should all be zero, but in practice roundoff errors cause them to be nonzero. By subtracting out the tiny components Jq_it_{ij}, we ensure that the spaces $\text{span}\{q_1, \ldots, q_j\}$ are isotropic to working precision. For accuracy we use the modified Gram–Schmidt method [17] to compute the coefficients h_{ij} and t_{ij}, rather than applying the formulas (3.5) literally. Finally, since modified Gram–Schmidt does not always generate vectors that are orthogonal to working precision [12], we perform two orthogonalization sweeps on $q_1, \ldots, q_j$. After j steps of the isotropic Arnoldi process we have vectors $q_1, \ldots, q_{j+1}$ satisfying

$$AQ_j = Q_jH_j + JQ_jT_j + q_{j+1}h_{j+1,j}e_j^T, \tag{5.1}$$

where $Q_j = [q_1 \cdots q_j]$, and H_j and T_j are matrices of coefficients. Our method ignores T_j and so depends on the fact that the elements of T_j are tiny. Our implementation of the IRA method is standard [39]; the shifts are chosen to select for the eigenvalues of A of largest magnitude. Each implicit restart, i.e., each iteration of the IRA method, produces a new configuration of the form (5.1) with a different starting vector q_1 and, typically, a smaller $h_{j+1,j}$. After several restarts, $h_{j+1,j}$ becomes negligible, and we have, up to roundoff errors,

$$AQ_j = Q_jH_j.$$

The eigenvalues of H_j are j of the eigenvalues of $A = R_1(\lambda_0, \mathcal{W})$, and they are typically the j largest in magnitude. If we actually want fewer than j eigenvalues, we can monitor $h_{k+1,k}$ for $k \le j$. If we stop when $h_{k+1,k}$ is negligible, we get k eigenvalues.

5.1. Eigenvalue computation. The Arnoldi process yields eigenvalues of the matrix $R_1(\lambda_0, \mathcal{W})$, but we actually want eigenvalues of the Hamiltonian matrix $\mathcal{W}$. Each eigenvalue μ of R_1 corresponds to two eigenvalues $\pm\lambda$ of $\mathcal{W}$ satisfying $(\lambda^2 - \lambda_0^2)(\lambda^2 - \bar{\lambda}_0^2) = 1/\mu$. A seemingly straightforward approach is to solve the quadratic equation

$$(\nu - \lambda_0^2)(\nu - \bar{\lambda}_0^2) = 1/\mu \tag{5.2}$$

for ν, and then compute $\pm\sqrt{\nu}$ to get the eigenvalues. Unfortunately, (5.2) has two solutions, only one of which corresponds to eigenvalues of $\mathcal{W}$. Thus one is faced with deciding which ν is the correct one.

We have adopted a different approach which dodges this decision and also allows us to make a final test of the backward stability of the result. Once $h_{k+1,k}$ is negligible, the space $\operatorname{span}\{q_1, \ldots, q_k\}$ is, up to roundoff errors, an invariant subspace under R_1. Normally it is also invariant under $\mathcal{W}^2$ (but not under $\mathcal{W}$) since R_1 is a function of $\mathcal{W}^2$. The space $\operatorname{span}\{q_1, \ldots, q_k\}$ can fail to be invariant under $\mathcal{W}^2$ only if two distinct eigenvalues of $\mathcal{W}^2$ are mapped to the same eigenvalue of R_1. We calculate the Ritz values of $\mathcal{W}^2$ with respect to the space $\operatorname{span}\{q_1, \ldots, q_k\}$; that is, we calculate the eigenvalues μ_i of $B = Q_k^T \mathcal{W}^2 Q_k$. Then $\pm\sqrt{\mu_i}$ are the eigenvalues of $\mathcal{W}$ that we seek.

As a byproduct of the computation of B, we obtain the vectors $\mathcal{W}^2 Q_k$, which we use to compute the residual

$$\left\| \mathcal{W}^2 Q_k - Q_k B \right\|_F \tag{5.3}$$

and thereby check whether or not $\operatorname{span}\{q_1, \ldots, q_k\}$ really is invariant under $\mathcal{W}^2$. This test is necessary because, as we have already mentioned, it can happen that a space that is invariant under R_1 fails to be invariant under $\mathcal{W}^2$. For example, if λ_0 is chosen so unfortunately that two distinct eigenvalues of $\mathcal{W}^2$ are mapped to the same eigenvalue of R_1, then R_1 will have a four-dimensional eigenspace spanned by two two-dimensional eigenspaces of $\mathcal{W}^2$. If the Arnoldi process picks up only one vector from this space, it will typically not be an eigenvector of $\mathcal{W}^2$, and the space $\operatorname{span}\{q_1, \ldots, q_k\}$ will not be invariant under $\mathcal{W}^2$.

It is interesting to note that the practical need for such a stability test is related to our insistence upon enforcing isotropy. In principle a Krylov subspace can contain at most a one-dimensional subspace of a multidimensional eigenspace, so the Arnoldi process will find at most one copy of a geometrically multiple eigenvalue. However, as is well known [40], roundoff errors will turn multiple eigenvalues into simple ones, and hence we will detect different approximations of the multiple eigenvalues. This is mostly a nuisance for us, because the eigenvalues of $\mathcal{W}^2$ are all geometrically multiple, and we would rather not pay to calculate two copies of each eigenvalue. Therefore, we use the isotropic Arnoldi method which enforces isotropy. It has the effect that each eigenvalue is picked up only once in practice. It has the unfortunate side effect that when a four-dimensional invariant subspace of R_1 consists of two two-dimensional invariant subspaces of $\mathcal{W}^2$, only one vector from that space is found, from which it is impossible to deduce eigenvectors of $\mathcal{W}^2$. Fortunately the merging of eigenspaces is a rare event.

Finally, we should note that this approach to eigenvalue calculation requires application of the operator $\mathcal{W}^2$. In our quadratic eigenvalue applications, linearized as in (2.5) or (2.8), we have

$$\mathcal{W} = \begin{bmatrix} I & -\frac{1}{2}G \\ 0 & I \end{bmatrix} \begin{bmatrix} 0 & -K \\ M^{-1} & 0 \end{bmatrix} \begin{bmatrix} I & -\frac{1}{2}G \\ 0 & I \end{bmatrix}.$$

To apply M^{-1}, we need to compute the Cholesky decomposition of M. Typically the Cholesky factor is quite sparse, and this step does not add substantially to the overall computing time. At this point in the computation we no longer need the LU factors of $Q(\lambda_0)$ (used for applying R_1), so we can use that storage space for the Cholesky factor of M.

5.2. Eigenvector computation. In the course of the eigenvalue computation we can easily obtain the corresponding Ritz vectors, which are eigenvectors of $\mathcal{W}^2$. Each of these is a particular member of a two-dimensional eigenspace of $\mathcal{W}^2$ but

in general not an eigenvector of $\mathcal{W}$. Thus, if we want eigenvectors to go with our eigenvalues, we need to do more work. This is a shortcoming of the skew-Hamiltonian approach.

Given eigenvalues, the quickest way to obtain corresponding eigenvectors is to perform inverse iteration. In the case of the quadratic eigenvalue problem, in view of [41], we refer back to the original form of the problem:

$$Q(\lambda)v = \lambda^2 Mv + \lambda Gv + Kv = 0.$$

For each eigenvalue λ we form a sparse LU decomposition of $Q(\lambda)$ and use the decomposition to perform one step of inverse iteration

$$v = Q(\lambda)^{-1}w.$$

Due to the form of linearization that we have used in (2.8), the bottom half of the eigenvector of $\mathcal{W}$ corresponds to an eigenvector of the quadratic eigenvalue problem. Therefore we use as starting vector w the bottom half of the $2n$-dimensional Ritz vector of $\mathcal{W}^2$ associated with λ^2. We have found that the choice of starting vectors is not crucial. Even random starting vectors produce good results. Note further that the residual $\|Q(\lambda)v\|_2 / \|v\|_2$ gives another measure of backward stability of the computation.

To compute the eigenvector associated with the eigenvalue $-\lambda$, we exploit the relationship $Q(-\lambda) = Q(\lambda)^T$. Thus, as in (4.8), the LU decomposition of $Q(\lambda)$ can also be used for the computation $v = Q(-\lambda)^{-1}w$.

If the complex eigenvalue λ has eigenvector v, then $\bar{\lambda}$ has eigenvector $\bar{v}$.

In summary, the cost of computing the eigenvectors associated with a pair $\{\lambda, -\lambda\}$ or a quadruple $\{\lambda, -\lambda, \bar{\lambda}, -\bar{\lambda}\}$ is one sparse LU decomposition plus four sparse triangular solves. If several sets of eigenvectors are wanted, they can be computed in parallel, given available processors and memory, or they can be computed sequentially, reusing the memory space for the LU decompositions.

6. Numerical results. We applied the skew-Hamiltonian Arnoldi method to solve numerous quadratic eigenvalue problems

$$\lambda^2 Mx + \lambda Gx + Kx = 0. \tag{6.1}$$

All computations were done in Matlab version 5.2 on a Linux machine.

In one class of problems that we considered, we built matrices of order $n = m^2$ by a tensor product construction. Let B denote the $m \times m$ nilpotent Jordan block

$$B = \begin{bmatrix} 0 & & & 0 \\ 1 & & & \\ & \ddots & & \\ & & 1 & 0 \end{bmatrix},$$

and define $\tilde{M} = \frac{1}{6}(4I_m + B + B^T)$, $\tilde{G} = B - B^T$, and $\tilde{K} = -(2I_m - B - B^T)$. Then we set

$$\begin{aligned} M &= c_{11} I_m \otimes \tilde{M} + c_{12} \tilde{M} \otimes I_m, \\ G &= c_{21} I_m \otimes \tilde{G} + c_{22} \tilde{G} \otimes I_m, \\ K &= c_{31} I_m \otimes \tilde{K} + c_{32} \tilde{K} \otimes I_m, \end{aligned} \tag{6.2}$$

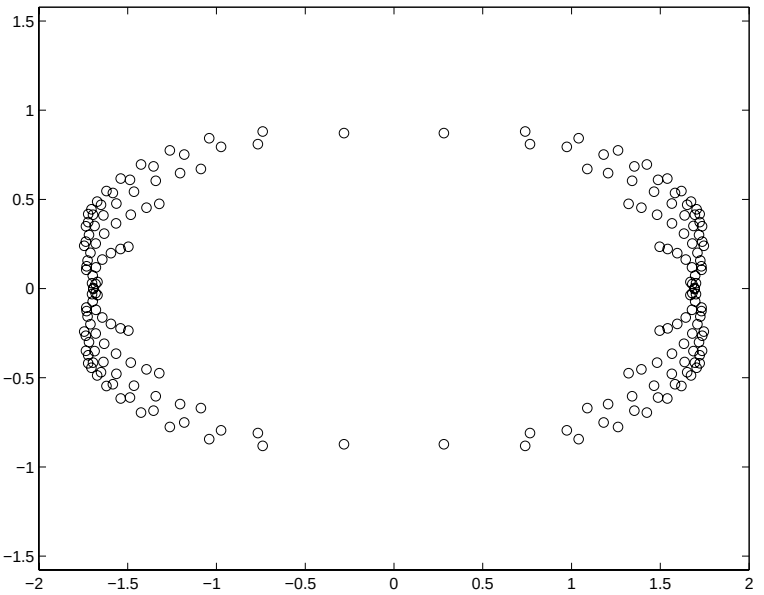


FIG. 6.1. *Eigenvalues of* 100×100 *quadratic pencil.*

where the coefficients c_{ij} are positive constants. We have $M = M^T > 0$, $G = -G^T$, and $K = K^T < 0$.

Example 6.1. If we take $m = 10$ and

$$\begin{array}{llllll} c_{11} & = & 1.00, & c_{12} & = & 1.30, \\ c_{21} & = & 1.35, & c_{22} & = & 1.10, \\ c_{31} & = & 1.00, & c_{32} & = & 1.20, \end{array} \tag{6.3}$$

then we obtain a 100×100 quadratic pencil, whose 200 eigenvalues are shown in Figure 6.1. These were computed by applying Matlab's `eig` command to the 200×200 matrix pencil (2.8), ignoring all structure, at a cost of 885×10^6 flops.

Suppose we want to use SHIRA to compute the 12 eigenvalues that are closest to the imaginary axis. Then, supposing that we know nothing about where the eigenvalues lie, our safest course of action is to choose a target shift λ_0 that lies on the imaginary axis. Table 6.1 shows the flop counts for computing these eigenvalues using three different choices of purely imaginary target. Results are given for our structured method SHIRA and a competing unstructured method, which applies the IRA method (in complex arithmetic) to the shifted inverted Hamiltonian operator $(\mathcal{W} - \lambda_0 I)^{-1}$; see (4.7). Because of the shift, this operator has no structure.

TABLE 6.1

Flop count for computing the 12 *smallest eigenvalues (and associated eigenvectors) of the pencil in Example* 6.1 *using structured and unstructured methods.*

	Flops (10^6)	
λ_0	SHIRA	Unstructured
$0.1i$	17.3	71.8
$1.0i$	11.4	26.3
$5.0i$	44.9	98.2

We see that both methods benefit from a good choice of shift, but regardless of shift, SHIRA outperforms the unstructured method by a factor of two or more. SHIRA applies IRA (in real arithmetic) to the skew-Hamiltonian operator R_1 (3.2) and finds the six largest eigenvalues (three complex-conjugate pairs), which correspond to three quadruplets of eigenvalues of (6.1). We used 10 Arnoldi steps per implicit restart. Using the shift $\lambda_0 = i$, for example, the competing method finds the six eigenvalues (in the upper half-plane) that are closest to the shift, not realizing that they constitute three pairs $(\lambda, -\bar{\lambda})$. We then deduce six other eigenvalues by taking complex conjugates. Again we used 10 Arnoldi steps per implicit restart. For both methods we used a deflation tolerance of 10^{-10}. In all cases the errors in the computed eigenvalues and eigenvectors were less than 10^{-9}.

Example 6.2. Consider another pencil built from matrices of the form (6.2), this time with $m = 5$. Use the same coefficients as in (6.3), except that $c_{21} = 0.1$. This results in a pencil whose smallest eigenvalues are real. The three smallest positive eigenvalues are

$$\begin{aligned} \lambda_1 &= 0.6726432397672, \\ \lambda_2 &= 0.9866442639296, \\ \lambda_3 &= 1.0689101679903. \end{aligned}$$

Using SHIRA with an appropriate tolerance and any reasonable shift (e.g., $\lambda_0 = 0$, i, or 0.5), we can compute these eigenvalues quickly, with any desired accuracy up to machine precision. The purpose of this example is to show that an unfortunate choice of shift can cause our eigenvalue computation scheme to fail. If we take $\lambda_0 = \hat{\lambda}_0 = \sqrt{(\lambda_1^2 + \lambda_2^2)/2}$, then the two eigenvalues λ_1^2 and λ_2^2 of $\mathcal{W}^2$ are mapped to the same eigenvalue of $R_1(\lambda_0, \mathcal{W})$, which then has a four-dimensional eigenspace associated with that eigenvalue. With this choice of λ_0, SHIRA is unable to calculate correct values for λ_1 and λ_2. Checking the residual (5.3), we find that $\left\|\mathcal{W}^2 Q_k - Q_k B\right\|_F \approx .37 \left\|\mathcal{W}^2 Q_k\right\|_F$, indicating that the space $\operatorname{span}\{q_1, \ldots, q_k\}$ is not invariant under $\mathcal{W}^2$. This is a red flag that tells us that our results cannot be trusted.

A simple remedy is to change λ_0. Even a tiny change suffices. For example, if we run SHIRA with $\lambda_0 = \hat{\lambda}_0 + 10^{-5}$ (and deflation tolerance 10^{-10}), we get the three smallest eigenvalues correct to twelve decimal places compared with the eigenvalues computed by the QZ-algorithm [17]. The residual (5.3) is $\left\|\mathcal{W}^2 Q_k - Q_k B\right\|_F \approx \left(4 \times 10^{-11}\right) \left\|\mathcal{W}^2 Q_k\right\|_F$, indicating that the subspace is invariant under $\mathcal{W}^2$ to within the desired tolerance.

Example 6.3. Finally, we consider a quadratic eigenvalue problem obtained by a finite element discretization of equations of elastic deformation of an anisotropic material [19, 21, 34]. The matrices have dimension 2223. Suppose we wish to find the twelve eigenvalues closest to the imaginary axis. It is known a priori that the eigenvalues lie near the real axis, so it makes sense to use a real target shift. In fact, the six smallest eigenvalues in the right half-plane are

$$\begin{aligned} \lambda_1 &= 0.96269644895, \\ \lambda_2 &= 0.98250961158 + 0.00066849814i, \\ \lambda_3 &= 0.98250961158 - 0.00066849814i, \\ \lambda_4 &= 1.35421843051, \\ \lambda_5 &= 1.39562564903, \\ \lambda_6 &= 1.49830518846. \end{aligned}$$

The flops needed to compute these eigenvalues by SHIRA using R_1, R_2, and the

TABLE 6.2
Flop count for computing smallest 12 *eigenvalues of the quadratic pencil in Example* 6.3 *using structured and unstructured methods.*

	Flops (10^7)		—
λ_0	SHIRA R_1	SHIRA R_2	Unstructured
0	32.6	28.4	140.1
0.3	32.6	28.5	79.6
0.6	28.4	25.7	69.8
0.9	28.4	23.0	50.7
1.2	19.8	17.5	31.5

unstructured method using various choices of real target shift are given in Table 6.2.

Again we used a deflation tolerance of 10^{-10}. We see that SHIRA can benefit from a good choice of λ_0 but also does well if a good shift is not known. In particular, it performs well even for the poor but safe choice $\lambda_0 = 0$. In each case SHIRA computes the six largest eigenvalues of $R_1(\lambda_0, \mathcal{W})$ or $R_2(\lambda_0, \mathcal{W})$, which turn out to be one complex pair and four real eigenvalues. These yield twelve eigenvalues of $\mathcal{W}$, one complex quadruplet, and four real $\{\lambda, -\lambda\}$ pairs. In all of these runs we used nine Arnoldi steps per implicit restart.

In contrast with SHIRA, the unstructured method is highly dependent on a good choice of shift. Given a good shift, such as $\lambda_0 = 1.2$, the unstructured method was nearly competitive with SHIRA. It computed the six positive eigenvalues closest to λ_0, which turned out to be the eigenvalues that we wanted. We then deduced the six negative eigenvalues $-\lambda_1, \ldots, -\lambda_6$ from the structure. We used nine Arnoldi steps per implicit restart.

The problem with the unstructured approach is that if we choose a target that is too large, we might miss the smallest eigenvalues. On the other hand, if we take a shift that is too close to the origin, then since an unstructured method does not reflect the relationship between the paired eigenvalues $\lambda, -\lambda, \bar{\lambda}, -\bar{\lambda}$, we end up computing some of the left half-plane eigenvalues explicitly. For example, in the case $\lambda_0 = 0.3$, λ_6 is not among the six smallest eigenvalues of the shifted operator. We had to compute nine eigenvalues in order to find $\lambda_1, \ldots, \lambda_6$; we also got $-\lambda_1$, $-\lambda_2$, and $-\lambda_3$. We took twelve Arnoldi steps per restart.

Using the shift $\lambda_0 = 0$, the safest choice, we have to compute all twelve eigenvalues explicitly. We used fifteen Arnoldi steps per restart. As Table 6.2 shows, many more flops were needed in this case than in all other cases.

Table 6.2 shows only the cost of the eigenvalue computation, which is SHIRA's strength. If one wants eigenvectors as well, the cost is an extra 15.7×10^7 flops for SHIRA and only 2.3×10^7 flops for the unstructured approach. If one adds these numbers to those in Table 6.2, one sees that SHIRA is still faster, indeed much faster in the cases where the best shift is not known in advance.

7. Conclusion. We have discussed structure preserving shift-and-invert Krylov subspace methods for the computation of a few eigenvalues and eigenvectors of large, sparse SHH pencils. We have demonstrated that a skew-Hamiltonian shift-and-invert IRA method can speed up the computation of desired eigenvalues in the interior of the spectrum significantly. Furthermore, by this approach it is guaranteed that the computed spectrum has the correct symmetry structure.

REFERENCES

[1] G. Ammar, C. Mehl, and V. Mehrmann, *Schur-like forms for matrix Lie groups, Lie algebras, and Jordan algebras*, Linear Algebra Appl., 287 (1999), pp. 11–39.

[2] G. Ammar and V. Mehrmann, *On Hamiltonian and symplectic Hessenberg forms*, Linear Algebra Appl., 149 (1991), pp. 55–72.

[3] P. Benner, R. Byers, V. Mehrmann, and H. Xu, *Numerical Computation of Deflating Subspaces of Embedded Hamiltonian Pencils*, Tech. report 99-15, Sonderforschungsbereich 393, Numerische Simulation auf massiv parallelen Rechnern, Fakultät für Mathematik, TU Chemnitz, 09107 Chemnitz, Germany, 1999. Available online from http://www.tu-chemnitz.de/sfb393/sfb99pr.html.

[4] P. Benner, R. Byers, V. Mehrmann, and H. Xu, *Numerical methods for linear quadratic and H_∞ control problems*, in Dynamical Systems, Control, Coding, Computer Vision, Progr. Systems Control Theory 25, G. P. and D. S. Gillian, eds., Birkhäuser, Basel, 1999, pp. 203–222.

[5] P. Benner and H. Fassbender, *An implicitly restarted symplectic Lanczos method for the Hamiltonian eigenvalue problem*, Linear Algebra Appl., 263 (1997), pp. 75–111.

[6] P. Benner and H. Fassbender, *The symplectic eigenvalue problem, the butterfly form, the SR algorithm, and the Lanczos method*, Linear Algebra Appl., 275/276 (1998), pp. 19–47.

[7] P. Benner, V. Mehrmann, and H. Xu, *A new method for computing the stable invariant subspace of a real Hamiltonian matrix*, J. Comput. Appl. Math., 86 (1997), pp. 17–43.

[8] P. Benner, V. Mehrmann, and H. Xu, *A numerically stable, structure preserving method for computing the eigenvalues of real Hamiltonian or symplectic pencils*, Numer. Math., 78 (1998), pp. 329–358.

[9] P. Benner, V. Mehrmann, and H. Xu, *A note on the numerical solution of complex Hamiltonian and skew-Hamiltonian eigenvalue problems*, Electron. Trans. Numer. Anal., 8 (1999), pp. 115–126.

[10] P. Benner, V. Mehrmann, and H. Xu, *Perturbation Analysis of the Eigenvalue Problem of a Formal Matrix Product*, Berichte aus der Technomathematik, Report 00–01, FB3–Mathematik und Informatik, Universität Bremen, 28334 Bremen, Germany, 2000. Available online from http://www.math.uni-bremen.de/zetem/berichte.html.

[11] R. Byers, C. He, and V. Mehrmann, *Where is the nearest non-regular pencil?*, Linear Algebra Appl., 285 (1998), pp. 81–105.

[12] J. Daniel, W. Gragg, L. Kaufman, and G. Stewart, *Reorthogonalization and stable algorithms for updating the Gram-Schmidt QR factorization*, Math. Comp., 30 (1976), pp. 772–795.

[13] E. D'yakunov and A. Knyazev, *The group iterative method for finding low-order eigenvalues*, Vestnik Moskov. Univ. Ser. XV Vychisl. Mat. Kibernet., 2 (1982), pp. 29–34.

[14] W. Ferng, W.-W. Lin, and C.-S. Wang, *Numerical algorithms for undamped gyroscopic systems*, Comput. Math. Appl., 37 (1999), pp. 49–66.

[15] R. Freund, *Transpose-free quasi-minimal residual methods for non-hermitian linear systems*, in Recent Advances in Iterative Methods, IMA Vol. Math. Appl. 60, G. Golub, A. Greenbaum, and M. Luskin, eds., Springer-Verlag, New York, 1994, pp. 69–94.

[16] R. W. Freund, M. H. Gutknecht, and N. M. Nachtigal, *An implementation of the look-ahead Lanczos algorithm for non-Hermitian matrices*, SIAM J. Sci. Comput., 14 (1993), pp. 137–158.

[17] G. Golub and C. Van Loan, *Matrix Computations*, 3rd ed., Johns Hopkins University Press, Baltimore, MD, 1996.

[18] A. S. Hodel and K. Poolla, *Heuristic approaches to the solution of very large sparse Lyapunov and algebraic Riccati equations*, in Proceedings of the 27th IEEE Conference on Decision and Control, Austin, TX, 1988, pp. 2217–2222.

[19] V. Kozlov, V. Maz'ya, and J. Rossmann, *Spectral properties of operator pencils generated by elliptic boundary value problems for the Lamé system*, Rostock. Math. Kolloq., 51 (1997), pp. 5–24.

[20] P. Lancaster, *Strongly stable gyroscopic systems*, Electron J. Linear Algebra, 5 (1999), pp. 53–66.

[21] D. Leguillon, *Computation of 3d-singularities in elasticity*, in Boundary Value Problems and Integral Equations in Nonsmooth Domains, Lecture Notes in Pure and Appl. Math. 167, Marcel Dekker, New York, 1995, pp. 161–170.

[22] W.-W. Lin, V. Mehrmann, and H. Xu, *Canonical forms for Hamiltonian and symplectic matrices and pencils*, Linear Algebra Appl., 301–303 (1999), pp. 469–533.

[23] K. Meerbergen, *Locking and Restarting Quadratic Eigenvalue Solvers*, Report RAL-TR-1999-

011, Rutherford Appleton Laboratory, Computational Science and Engineering Department, Atlas Centre, Oxon, OX11 0QX, UK, 1999.

[24] C. Mehl, *Compatible Lie and Jordan Algebras and Applications to Structured Matrices and Pencils*, Ph.D. thesis, Fakultät für Mathematik, Technische Universität Chemnitz, 09107 Chemnitz, Germany, 1998.

[25] C. Mehl, *Condensed forms for skew-Hamiltonian/Hamiltonian pencils*, SIAM J. Matrix Anal. Appl., 21 (1999), pp. 454–476.

[26] V. Mehrmann, *The Autonomous Linear Quadratic Control Problem, Theory and Numerical Solution*, in Lecture Notes in Control and Inform. Sci. 163, Springer-Verlag, Berlin, 1991.

[27] V. Mehrmann, *A step toward a unified treatment of continuous and discrete time control problems*, Linear Algebra Appl., 241–243 (1996), pp. 749–779.

[28] A. Meyer, *Die simultane Iteration zur Eigenwertberechnung für beliebige endlichdimensionale Operatoren und die Anwendung auf Schwingungsprobleme*, Report Wissenschaftliche Informationen 36, Fakultät für Mathematik, Technische Hochschule Karl-Marx-Stadt (now Technische Universität Chemnitz), Chemnitz, Germany, 1982.

[29] B. Parlett, *The Symmetric Eigenvalue Problem*, Prentice-Hall, Englewood Cliffs, NJ, 1980.

[30] B. Parlett and Y. Saad, *Complex shift and invert strategies for real matrices*, Linear Algebra Appl., 88/89 (1987), pp. 575–595.

[31] T. Penzl, *A cyclic low-rank Smith method for large sparse Lyapunov equations*, SIAM J. Sci. Comput., 21 (2000), pp. 1401–1418.

[32] I. G. Rosen and C. Wang, *A multilevel technique for the approximate solution of operator Lyapunov and algebraic Riccati equations*, SIAM J. Numer. Anal., 32 (1995), pp. 514–541.

[33] Y. Saad, *Numerical Methods for Large Eigenvalue Problems*, Manchester University Press, Manchester, UK, 1992.

[34] H. Schmitz, K. Volk, and W. L. Wendland, *On three-dimensional singularities of elastic fields near vertices*, Numer. Methods Partial Differential Equations, 9 (1993), pp. 323–337.

[35] H. Schwarz, *Methode der finiten Elemente*, Teubner, Stuttgart, Germany, 1984.

[36] G. Sleijpen and H. van der Vorst, *A Jacobi–Davidson iteration method for linear eigenvalue problems*, SIAM J. Matrix Anal. Appl., 17 (1996), pp. 401–425.

[37] S. Solov'ev, *The finite-element method for symmetric eigenvalue nonlinear problems*, Comput. Math. Math. Phys., 37 (1997), pp. 1269–1276. Translated from Zhurnal Vyschislitel'noi Matematiki i Matematicheskoi Fiziki.

[38] S. Solov'ev, *Convergence of the Modified Subspace Iteration Method for Nonlinear Eigenvalue Problems*, Preprint SFB393/99-35, Sonderforschungsbereich 393, Numerische Simulation auf massiv parallelen Rechnern, Fakultät für Mathematik, Technische Universität Chemnitz, D-09107 Chemnitz, Germany, 1999. Available from http://www.tu-chemnitz.de/sfb393/sfb99pr.html.

[39] D. C. Sorensen, *Implicit application of polynomial filters in a k-step Arnoldi method*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 357–385.

[40] G. Stewart and J.-G. Sun, *Matrix Perturbation Theory*, Academic Press, New York, 1990.

[41] F. Tisseur, *Backward error analysis of polynomial eigenvalue problems*, Linear Algebra Appl., 309 (2000), pp. 339–361.

[42] C. Van Loan, *A symplectic method for approximating all the eigenvalues of a Hamiltonian matrix*, Linear Algebra Appl., 61 (1984), pp. 233–251.

SIAM J. SCI. COMPUT.
Vol. 22, No. 6, pp. 1926–1942
© 2001 Society for Industrial and Applied Mathematics

COMPARISON OF THE NODAL INTEGRAL METHOD AND NONSTANDARD FINITE-DIFFERENCE SCHEMES FOR THE FISHER EQUATION*

RIZWAN-UDDIN†

Abstract. The relationship between the so-called nonstandard finite-difference schemes and the nodal integral method (NIM) is investigated using the Fisher equation as a model problem. Exact and best finite-difference schemes are reviewed first. Next, the NIM for the Fisher equation is developed. It is shown that the NIM leads to a nonstandard evaluation of the derivatives. Moreover, the resulting scheme possesses the desirable characteristics of the nonstandard finite difference schemes, such as the nonlocal evaluation of the nonlinear terms. Thus, the NIM provides a systematic framework to obtain schemes *similar* to the *best finite-difference schemes*. Numerical results for a propagating front problem show that the NIM can capture the shape and speed of the front very accurately. Results also show that the *best finite-difference scheme* is stable for large grid sizes but only at the cost of inaccuracy in the front propagation speed. Additional results are obtained using the NIM for symmetric and asymmetric initial conditions. These describe the interaction of two fronts of advantageous genes that approach each other and merge. It is noted that the traveling fronts evolving from certain asymmetric initial conditions become indistinguishable from those that evolve from symmetric initial conditions.

Key words. nodal integral method, best finite-difference schemes, semianalytical numerical schemes, Fisher equation, wave propagation

AMS subject classifications. 65P99, 77A99

PII. S1064827597325463

1. Introduction.

Several nodal methods have been developed over the last two decades to solve partial differential equations (PDEs) [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]. A common feature among these is the analytical or semianalytical treatment of at least a part of the PDE(s). This is accomplished, for example, by introducing known solutions of one-dimensional, steady-state problems in multidimensional or time-dependent problems, or by reducing the PDEs to a set of ordinary differential equations (ODEs) via the transverse integration process, and then approximately solving the ODEs. Hence, a class of these methods is also known as *analytical nodal methods* [6, 8]. A major advantage of these methods is that they yield an accurate solution over large grid/step sizes. Consequently, these methods are especially suitable for problems with large spatial and/or time domains. The details of one particular kind of nodal method—the so-called *nodal integral* (*or integration*) *method* (NIM) [2, 5]—will be given in section 3 in the context of the Fisher equation (FE).

A different approach for the solution of PDEs (and ODEs) that also relies on analytical solutions or conservation laws leads to nonstandard finite-difference (NSFD) schemes such as the *exact finite-difference* (EFD) and/or the *best finite-difference* (BFD) schemes [12, 13, 14, 15, 16]. A finite-difference (FD) scheme is called an EFD scheme if the solution for all step sizes at all grid points is equal to the exact analytical solution of the differential equation at the corresponding grid points. It is intuitively expected that only those differential equations that have a closed form exact solution

*Received by the editors August 4, 1997; accepted for publication (in revised form) August 16, 2000; published electronically February 21, 2001.
http://www.siam.org/journals/sisc/22-6/32546.html
†Department of Nuclear, Plasma and Radiological Engineering, and the Computational Science and Engineering Program, University of Illinois at Urbana-Champaign, 214 NEL, 103 S. Goodwin Ave., Urbana, IL 61801 (rizwan@uiuc.edu).

will lead to EFD schemes. Not surprisingly, the interest in the EFD schemes—which are available for a limited number of ODEs and even fewer PDEs—is in their utility in gaining numerical insight for the development of improved FD schemes for problems that do not admit EFD schemes. For differential equations that do not admit EFD schemes, the corresponding notion of BFD schemes is developed—as the *next best thing* to an EFD scheme. While the idea of the EFD scheme is fairly well defined, the notion of the BFD scheme is a little vague. An FD scheme is called a BFD scheme if *the important properties of its solutions correspond exactly to the related properties of the solutions to the differential equation for all values of the step size* [15]. Specific important properties may include conservation laws, special cases of known exact solutions, and stability properties of fixed points, etc. BFD schemes for complex problems are developed by taking advantage of the known EFD schemes for simpler problems—imparting an analytical component to the BFD schemes.

A word on terminology: a *node* in the NIM (which roughly corresponds to the *grid point* in the FD methods) refers to a finite region in the space of independent variables. The discrete unknowns in the NIM are the phase variables *averaged* over $(n-1)$ dimensional surfaces of these nodes, where n is the number of independent variables in the problem.

Given the properties of the BFD schemes and the NIM, and their similar semi-analytical background, it is desirable to investigate the common features in these schemes with the goal of providing insight for the development of improved schemes. The fact that both schemes rely on specialized procedures specific to the problem at hand necessitates that this exercise be carried out in the context of a specific problem. The FE is chosen here to be the model problem for such a comparison.

The FE

$$\frac{\partial u(x,t)}{\partial t} = \frac{\partial^2 u(x,t)}{\partial x^2} + \lambda u(x,t)(1-u(x,t)) \tag{1}$$

describes the propagation of an advantageous gene in a one-dimensional infinite medium [17]. Kolmogoroff, Petrovsky, and Piscounov [18] studied this equation and showed, among other properties, that for all bounded initial conditions (ICs) in an infinite domain, the results remain bounded. Canosa [19] developed a second order perturbation solution for the FE for large characteristic speeds. Later, Gazdag and Canosa [20] used the *accurate space derivative* (ASD) technique [21] to numerically solve the FE and showed the evolution of various ICs into the traveling wave of minimal speed. Convergence of the ASD method required cutting off the right-hand tail of the wave moving toward the right. Moreover, the speed with which an IC evolved to the time-asymptotic minimum speed wave also depended upon the point at which the right-hand tail was cut off. As is shown below, the NIM does not have such restrictions. An excellent review of the FE (and other nonlinear diffusion problems), its properties, and its numerical solutions is given by Sachdev [22].

The rest of this paper is divided into five sections. The EFD and BFD schemes are briefly reviewed in section 2. The NIM is developed for the FE in section 3. The NIM scheme is compared in section 4 with the BFD scheme of Mickens [15], and similarities and differences between the two schemes are identified. Numerical results and discussion are in section 5. Section 6 summarizes the paper.

2. Review of EFDs and BFDs. This section is based on the material in [12, 15, 16]. Consider the following first and second order ODEs:

$$\frac{dy(t)}{dt} = \mu y(t), \tag{2}$$

$$\frac{d^2y(x)}{dx^2} + \nu y(x) = 0. \tag{3}$$

Various discrete approximations for the derivative in these equations lead to the standard FD schemes for the two equations. By standard FD schemes (SFD), we mean ones in which the derivatives are approximated by, for example, the backward, forward, or central difference formulas. With forward differences for the first derivative and central differences for the second derivative, the discrete forms of (2) and (3) are

$$\frac{y_j - y_{j-1}}{\Delta t} = \mu y_{j-1} \tag{4}$$

and

$$\frac{y_{i+1} - 2y_i + y_{i-1}}{(\Delta x)^2} + \nu y_i = 0, \tag{5}$$

where the subscript i indicates the spatial grid points and the subscript j indicates the time level. Later, for space- *and* time-dependent problems, the same subscripts, separated by a comma, will be used. Equations (4) and (5) will be referred to as the SFD schemes for (2) and (3), respectively. But, given the fact that the exact solution of (2) and (3) is known to any sophomore, one can easily construct the *exact* FD schemes for them by first (exactly) solving the differential equations with the corresponding initial and boundary conditions, and then by rewriting the solution in the same format as (4) and (5) [12]. This leads to

$$\frac{y_j - y_{j-1}}{\left(\dfrac{e^{\mu \Delta t} - 1}{\mu}\right)} = \mu y_{j-1} \tag{6}$$

and

$$\frac{y_{i+1} - 2y_i + y_{i-1}}{\dfrac{4}{\nu}\left[\sin\left(\dfrac{\sqrt{\nu}\,\Delta x}{2}\right)\right]^2} + \nu y_i = 0. \tag{7}$$

The SFD schemes—(4) and (5)—and the EFD schemes—(6) and (7)—differ from each other only by the denominators in the approximations of the derivatives. However, the latter schemes have no restrictions on the size of the time step or grid size—the results will always be exact (to machine accuracy). On the other hand, restrictions on the size of these parameters for the SFD schemes are well known. Note that the approximations for the derivatives in the SFD schemes are functions of the discrete variables and the grid size (Δt or Δx), whereas the "approximations" for the derivatives in the EFD schemes depend also on the system parameters (μ, ν).

This observation that the EFD schemes differ from the SFD schemes by so little (at least in these very simple cases), motivated a similar approach for *real* problems with no exact solution. As a generalization of the approximation for the first and second order derivatives, a general FD scheme has been suggested [15] in which the denominator in the FD approximations of the derivatives is generalized to functions of the grid size *and* problem parameters

$$\frac{y_j - y_{j-1}}{\Psi(\Delta t, \mu)} = \mu y_{j-1} \tag{8}$$

and

$$\frac{y_{i+1} - 2y_i + y_{i-1}}{\phi(\Delta x, \nu)} + \nu y_i = 0, \tag{9}$$

where $\Psi(\Delta t, \mu)$ and $\Phi(\Delta x, \nu)$ behave like

$$\Psi(\Delta t, \mu) = (\Delta t) + O((\Delta t)^2), \tag{10}$$

$$\Phi(\Delta x, \nu) = (\Delta x)^2) + O((\Delta x)^3). \tag{11}$$

The fact that (2) and (3) have known exact solutions made it easy, in fact possible, to find EFD schemes for them. For *nonlinear* ODEs that do not admit exact solutions, other conditions—some generic and some specific to each equation—are imposed to arrive at a *BFD* scheme. Consider the following equation:

$$\frac{d^2y}{dx^2} + \nu y(1 - y) = 0. \tag{12}$$

There is no EFD scheme for (12). Hence, a BFD scheme is developed by preserving the constant of motion of the above ODE (the energy, E, of the nonlinear conservative oscillator) [13]. In arriving at the BFD scheme for (12), it is also required that the BFD scheme for the nonlinear problem must reduce to the EFD for the linear problem when the nonlinear term is absent. The details are given in [13, 16].

While the development of the EFD and BFD schemes for ODEs is seemingly straightforward, extensions to PDEs have relied on somewhat ad hoc methods. One approach is to first develop the EFD/BFD schemes for the *subproblems* of the PDE in question. Subproblems are simply obtained from the original PDE by considering, for example, the steady-state case, the space-independent case, the linear version of the PDE, etc. These schemes are then combined in such a way that the discrete representation of the PDE reduces to the discrete version of the appropriate continuous subproblem in the corresponding limit [15]. For example, the BFD scheme for the FE is obtained by combining the EFD scheme for the space-independent case ($d^2u/dx^2 = 0$), the SFD scheme for the $\lambda = 0$ case, and the BFD scheme for the time-independent case ($du/dt = 0$, (12)). The BFD scheme for the FE is constructed such that *in the appropriate limit the scheme reduces to the proper discrete model of the relevant subequation* [16]. This leads to the following discretization for the FE:

$$\begin{aligned}\frac{u_{i,j} - u_{i,j-1}}{\left(\dfrac{e^{\lambda \Delta t} - 1}{\lambda}\right)} &= \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{\dfrac{4}{\lambda}\left[\sin\left(\dfrac{\sqrt{\lambda}\,\Delta x}{2}\right)\right]^2} \\ &\quad + \lambda u_{i,j-1} - \lambda\left(\frac{u_{i+1,j} + u_{i,j} + u_{i-1,j}}{3}\right) u_{i,j-1},\end{aligned} \tag{13}$$

where i indicates the spatial grid point and j is the time step counter. Here, the first order time- and the second order space derivatives are approximated by nonstandard approximations similar to those in (6) and (7) for the linear equations with EFD schemes. The nonlinear term in the discretized FE is represented *nonlocally*, i.e., it is evaluated not at one grid point but rather as the product of the *local* population density at the previous time step and a population density averaged over the three consecutive grid points at the current time step. This feature—nonlocal evaluation of

the nonlinear terms—appears repeatedly in such schemes. Despite the implicit nature of the other terms, the linear term is evaluated at the *previous* time step, and so is one of the u variables in the nonlinear (u^2) term. These features make this BFD scheme significantly different from SFD schemes.

In the next section, where the NIM is developed for the FE, it is shown that the discrete set of equations for the NIM—obtained through a much more systematic procedure than those used to obtain the BFD scheme—also has many of the same features.

3. NIM formulation for the FE. The space-time domain is divided into layers of n_x spatial *elements*, or *nodes*, each of width $2a$ in the horizontal (spatial) direction and of height Δt. Nonuniform nodes in space can easily be incorporated. A coordinate system local to each space-time node is defined with the origin at the bottom-center of the node. See Figure 1. Hence, $-a \le x' \le a$, $t_j \le t' \le t_j + \Delta t$. The space-averaged/time-dependent and time-averaged/space-dependent population densities over each node (i, j) are defined as

$$\bar{u}^x_{i,j}(t') \equiv \frac{1}{(2a)} \int_{-a}^{+a} u(x', t')\, dx', \tag{14}$$

$$\bar{u}^t_{i,j}(x') \equiv \frac{1}{\Delta t} \int_{t_j}^{t_j+\Delta t} u(x', t')\, dt'. \tag{15}$$

The discrete variables associated with the node (i, j) are $\bar{u}^x_{i,j}(\equiv \bar{u}^x_{i,j}(t' = t_j + \Delta t))$ and $\bar{u}^t_{i,j}(\equiv \bar{u}^t_{i,j}(x' = +a))$. Having defined the transverse-averaged, local phase variables, the next step in the development of the NIM is to reduce the PDE to a set of ODEs—one for each independent variable, obtained by averaging over all independent variables except one. As explained in detail below, the ODEs are then solved, and the constant(s) are eliminated in favor of the discrete variables at node interfaces. A set of coupled algebraic equations for the discrete variables is then obtained by imposing appropriate interface conditions (C^0, C^1, etc.) at node interfaces.

In general, the ODEs for transverse-averaged variables cannot be solved exactly. Hence, all the terms that can lead to an exact homogeneous solution are retained on the left-hand side, while the other terms are lumped into what is often called the *pseudosource term* on the right-hand side of the transverse-integrated equations. Complete solutions of the ODEs are written as the sum of the homogeneous and particular solutions. The pseudosource terms, being a function of the dependent variables, must be approximated before the particular solutions for the ODEs can be explicitly written. The pseudosource terms are hence projected onto a complete set of basis functions and truncated at a desired order. The number of terms kept in the expansion determine the order of the numerical scheme [5]. Since the pseudosource terms are unknown, this projection is only formal, and a scheme must be identified to determine the coefficients. Nodal schemes vary based on how these coefficients are determined. One approach is to approximate the pseudosource terms using the discrete variables from the previous iteration [2]. A second approach involves the imposition of certain constraint conditions to arrive at additional algebraic equations to eliminate the coefficients [5]. This is the procedure used below in the development of the NIM for the FE.

The FE allows us the freedom to incorporate the linear term in either the solution of the time-averaged equation, or in the solution of the space-averaged equation, or in both. A priori it is difficult to decide exactly how the linear term must be "divided"

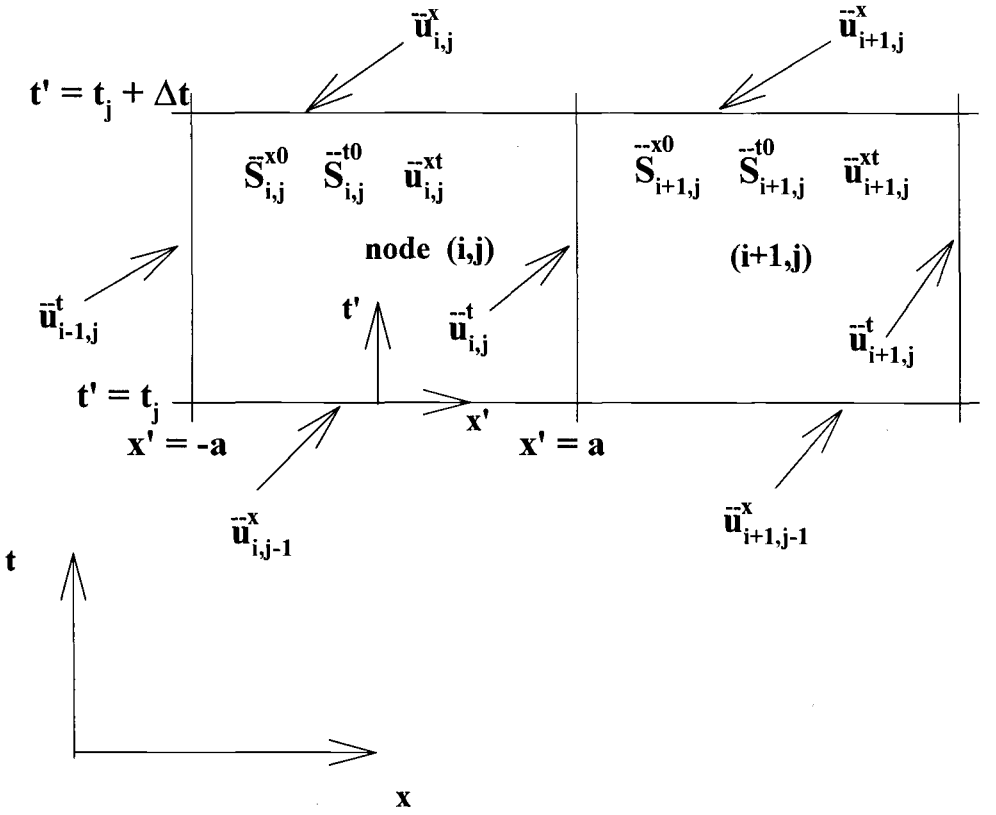


FIG. 1. *Schematic diagram of the space-time nodes and discrete variables in the nodal integral method.*

between the two ODEs for best results. Hence, we proceed with a general approach which allows us to numerically investigate the role of the linear term in each of the ODEs. The linear term in the FE is hence written as

$$\lambda u(x,t) \equiv \lambda(\alpha+\beta)u(x,t), \tag{16}$$

where $(\alpha+\beta)=1$.

Operating on the FE by $\frac{1}{\Delta t}\int_{t_j}^{t_j+\Delta t} dt'$, the equation for the time-step averaged, space-dependent, local population density becomes

$$\begin{aligned}\frac{d^2\bar{u}^t(x')}{dx'^2} + \lambda\alpha\bar{u}^t(x') &= \bar{S}^t(x') \\ &\equiv \frac{1}{\Delta t}\int_{t_j}^{t_j+\Delta t}\left(\frac{\partial u(x',t')}{\partial t'} - \lambda\beta u(x',t') + \lambda u^2(x',t')\right) dt',\end{aligned} \tag{17}$$

and the nodal space-averaged, time-dependent equation—obtained by operating on the FE by $\frac{1}{2a}\int_{-a}^{+a} dx'$—is

$$\frac{d\bar{u}^x(t')}{dt'} - \lambda\beta\bar{u}^x(t') = \bar{S}^x(t')$$

$$\equiv \frac{1}{2a}\int_{-a}^{+a}\left(\frac{\partial^2 u(x',t')}{\partial x'^2} + \lambda\alpha u(x',t') - \lambda u^2(x',t')\right)dx', \tag{18}$$

where the subscript (i,j) has been omitted for convenience. The two ODEs are now solved. Note that different combinations of α and β will yield different functional forms for node interior variation of the space-averaged/time-dependent and time-averaged/space-dependent population densities. Specifically, assuming λ is always positive, a positive, zero, or negative α will yield (for zeroth order approximation for the pseudosource term), respectively, a trigonometric, quadratic, or exponential variation over the space dimension. Similarly, a positive/negative or zero β will yield, respectively, an exponential or linear variation for the space-averaged population density over time. In the development below, α may be negative, and hence the trigonometric functions with complex arguments in those cases will actually be exponential (or hyperbolic) functions.

À la nodal integral scheme [5], the pseudosource terms $\bar{S}^t(x')$ and $\bar{S}^x(t')$ in (17) and (18) are expanded in, say, Legendre polynomials, and truncated at the zeroth order—which is known to lead to a second order scheme. Hence, $\bar{S}^t(x') \cong \bar{S}^{t0}$ and $\bar{S}^x(t') \cong \bar{S}^{x0}$. Equations (17) and (18) are now solved over the node (i,j) with initial and boundary conditions (Figure 1)

$$\begin{aligned} \bar{u}^x(t'=t_j) &= \bar{u}^x_{i,j-1}, \\ \bar{u}^t(x'=-a) = \bar{u}^t_{i-1,j}, &\qquad \bar{u}^t(x'=a) = \bar{u}^t_{i,j}. \end{aligned} \tag{19}$$

The results are

$$\bar{u}^t(x') = \left[\frac{\bar{S}^{t0}_{i,j}}{\lambda\alpha}\right] - \left[\frac{2\bar{S}^{t0}_{i,j} - (\lambda\alpha)(\bar{u}^t_{i,j} + \bar{u}^t_{i-1,j})}{2(\lambda\alpha)\cos(\sqrt{\lambda\alpha}\,a)}\right]\cos(\sqrt{\lambda\alpha}\,x')$$

$$+ \left[\frac{(\bar{u}^t_{i,j} - \bar{u}^t_{i-1,j})}{2\sin(\sqrt{\lambda\alpha}\,a)}\right]\sin(\sqrt{\lambda\alpha}\,x') \tag{20}$$

and

$$\bar{u}^x(t') = -\frac{\bar{S}^{x0}_{i,j}}{(\lambda\beta)} + \left[\frac{\bar{S}^{x0}_{i,j}}{(\lambda\beta)} + \bar{u}^x_{i,j-1}\right]e^{\lambda\beta(t'-t_j)}, \tag{21}$$

where, again, it is understood that these are *local* solutions within the space-time node (i,j).

The discrete equation for the time-step-averaged population density, $\bar{u}^t_{i,j}$, is obtained by requiring the first derivative of the time-step-averaged u to be continuous at the node interface (continuity of flux condition). Hence, two expressions are evaluated and equated for the derivative $d\bar{u}^t(x')/dx'$ at the common interface between nodes i and $(i+1)$, i.e., (20)—which is for node (i,j)—is differentiated and evaluated at $x' = a$, and the result is equated to the derivative of the corresponding equation for the node $(i+1,j)$ evaluated at $x' = -a$. This yields the three point scheme that relates $\bar{u}^t_{i-1,j}$, $\bar{u}^t_{i,j}$, and $\bar{u}^t_{i+1,j}$:

$$\frac{\bar{u}^t_{i+1,j} - 2\bar{u}^t_{i,j} + \bar{u}^t_{i-1,j}}{\dfrac{4}{\lambda\alpha}\left[\sin\left(\dfrac{\sqrt{\lambda\alpha}\,\Delta x}{2}\right)\right]^2} + \lambda\alpha\bar{u}^t_{i,j} = \frac{(\bar{S}^{t0}_{i,j} + \bar{S}^{t0}_{i+1,j})}{2}. \tag{22}$$

The discrete equation for the space-averaged population density for the node, $\bar{u}^x_{i,j}$, is obtained by simply evaluating (21) at $t' = (t_j + \Delta t)$:

$$\frac{\bar{u}^x_{i,j} - \bar{u}^x_{i,j-1}}{\left[\dfrac{e^{\lambda\beta\Delta t} - 1}{\lambda\beta}\right]} - \lambda\beta\bar{u}^x_{i,j-1} = \bar{S}^{x0}_{i,j}. \tag{23}$$

To eliminate the pseudosource terms $\bar{S}^{t0}_{i,j}$ and $\bar{S}^{x0}_{i,j}$, two conditions are imposed [5]. First the FE (1) is integrated over the space-time node

$$\frac{1}{2a\Delta t}\int_{-a}^{+a}\int_{t_j}^{t_j+\Delta t}\left(\frac{\partial u(x',t')}{\partial t'} - \frac{\partial^2 u(x',t')}{\partial x'^2}\right.$$

$$\left. - \lambda\alpha u(x',t') - \lambda\beta u(x',t') + \lambda u^2(x',t')\right) dx'\,dt' = 0, \tag{24}$$

which, using the definitions of the pseudosource terms $\bar{S}^{t0}_{i,j}$ and $\bar{S}^{x0}_{i,j}$ from (17) and (18), becomes

$$\bar{S}^{t0}_{i,j} = \bar{S}^{x0}_{i,j} + \lambda(\bar{u}^{xt}_{i,j})^2, \tag{25}$$

where the average of the product has been approximated by the product of the averages, which is known to lead to a second order error [5], and the space-time-node-averaged velocity $\bar{u}^{xt}_{i,j}$ is defined in (26). The second condition is obtained by requiring that the space-time-node-averaged population density obtained by first averaging in time and then in space should be equal to the average population density obtained by first averaging in space and then in time, i.e.,

$$\bar{u}^{tx} \equiv \frac{1}{2a}\int_{-a}^{+a}\bar{u}^t(x')\,dx' = \frac{1}{\Delta t}\int_{t_j}^{t_j+\Delta t}\bar{u}^x(t')\,dt' \equiv \bar{u}^{xt}. \tag{26}$$

Equation (26) is evaluated by substituting the expressions for $\bar{u}^t(x')$ and $\bar{u}^x(t')$ from (20) and (21), respectively.

The node-averaged population density and the pseudosource terms are eliminated using (25) and (26) from the set of discrete equations (22) and (23). Expressions for $\bar{u}^x_{i,j}$ and $\bar{u}^t_{i,j}$ obtained after eliminating the pseudosource terms, $\bar{S}^{t0}_{i,j}$ and $\bar{S}^{x0}_{i,j}$, are given in the appendix. Equations (22), (23), and (25)—*with* pseudo source terms—are, however, much more instructive in analyzing the discrete form of the FE for the NIM than those given in the appendix (with pseudosource terms eliminated). Here, to explicitly bring out the nonlinear terms in the discrete equations, (23) and (22) are combined with (25) and rewritten below.

$$\frac{\bar{u}^x_{i,j} - \bar{u}^x_{i,j-1}}{\left[\dfrac{e^{\lambda\beta\Delta t} - 1}{\lambda\beta}\right]} - \lambda\beta\bar{u}^x_{i,j-1} + \lambda(\bar{u}^{xt}_{i,j})^2 = \bar{S}^{t0}_{i,j}, \tag{27}$$

$$\frac{\bar{u}^t_{i+1,j} - 2\bar{u}^t_{i,j} + \bar{u}^t_{i-1,j}}{\dfrac{4}{\lambda\alpha}\left[\sin\left(\dfrac{\sqrt{\lambda\alpha}\,\Delta x}{2}\right)\right]^2} + \lambda\alpha\bar{u}^t_{i,j} - \lambda\left[\frac{(\bar{u}^{xt}_{i,j})^2 + (\bar{u}^{xt}_{i+1,j})^2}{2}\right] = \frac{(\bar{S}^{x0}_{i,j} + \bar{S}^{x0}_{i+1,j})}{2}. \tag{28}$$

The structure of the discrete equations and their relationship to each other can now clearly be seen. Both (27) and (28) represent a discrete version of the FE. Hence, when compared with the FE, these equations' RHSs must represent the terms missing on their LHSs. Equation (27) is missing the diffusion term and the α-fraction of the linear term. The zeroth order approximation of the time-step-averaged pseudosource term, RHS of (27), indeed represents these terms missing from the LHS of (27); see (17). Equation (28), on the other hand, is missing the time derivative and the β-fraction of the linear term. Equation (18) shows that the RHS of (28) indeed represents the terms missing on the LHS, calculated in an average sense over the nodes on both sides of the surface on which the continuity of the derivative condition (C^1) is being imposed. Additional comments on the structure of these discrete equations are made in the next section.

4. Comparison of the BFD scheme and the NIM. The most obvious difference between the NIM and the BFD scheme (or the FD schemes in general) is that unlike point values in the FD schemes, the discrete variables in the NIM are the transverse-averaged variables at node surfaces. The number of discrete unknowns per node in the NIM is equal to the number of independent variables. Consequently for the FE, while there is only one unknown, the population density $u_{i,j}$, per grid point in the FD schemes, there are two unknowns (and hence two equations) for each of the space-time node, (i, j), in the NIM: the space-averaged population density at the top surface $\bar{u}^x_{i,j}$ and the time-averaged population density at the right surface $\bar{u}^t_{i,j}$. Hence, care should be exercised when comparing the two schemes.

The implicit BFD scheme for the FE is given by (13). Corresponding discrete equations for the NIM are (27) and (28), which clearly show the similarities with the BFD scheme. In the discrete equation for $\bar{u}^x_{i,j}$, (27), the time-derivative for the space-averaged population density is represented by the nonstandard form, as is the second derivative with respect to x in the discrete equation for the time-averaged population density $\bar{u}^t_{i,j}$, (28). The linear term in the discrete equation for $\bar{u}^x_{i,j}$ (27) is, as in the BFD scheme, evaluated at the *previous* time step. On the other hand, the linear term in (28) (second term on the LHS) is, in terms of the only appropriate discrete variable, the time-averaged population density at the point where the derivative is being evaluated. For comparison, the (entire) linear term in the BFD scheme is evaluated at the previous time step.

As stated previously, one of the characteristics of the BFD schemes is that the nonlinear terms in the discrete schemes are evaluated *nonlocally*. In fact, nonlocal evaluation of the nonlinear terms is considered to be essential in a BFD scheme [15]. Precisely how the nonlinear term must be evaluated is not known a priori and varies from one problem to another. For the FE, the u^2 term is represented by the product of the *local* u variable at the *previous* time step and the average of the population density at three grid points at the *current* time step. As in the BFD scheme, the NIM also leads to a discrete set of equations that requires nonlocal evaluation of the nonlinear terms. Moreover, the form of the nonlinear terms in the NIM has a precise physical meaning—each of the u variables is simply the average u over the immediate space-time domain that influences the discrete variable being evaluated. Marching in time—given $\bar{u}^x_{i,j-1}$, find $\bar{u}^x_{i,j}$—involves only a single node, and hence the nonlinear term in (27) is simply the square of the average population density in that node, (i, j). See Figure 1. On the other hand, evaluation of the time-step-averaged population density, $\bar{u}^t_{i,j}$ (28), depends upon two nodes, (i, j) and $(i+1, j)$, the unknown being at the interface of these two nodes. The nonlinear term is hence approximated nonlocally

as the arithmetic *average* of the square of the space-time-averaged population density in the neighboring nodes on each side.

Both schemes preserve some of the constants of motion. While only the time-independent subproblem of the BFD scheme conserves energy, the general space- and time-dependent NIM does satisfy the original PDE over each space-time node in an integral sense—a direct consequence of one of the conditions imposed to evaluate expressions for the pseudosource terms.

The systematic approach of the NIM for PDEs clearly shows that FD-type schemes that have the properties of the BFD schemes can be developed via a transverse integration process. This systematic approach of the NIM is in contrast with the *combining* process of the BFD schemes that requires the *assembly* of the FD schemes for subproblems such that the full model reduces to the BFD or EFD model of the subproblems in the appropriate limits [16].

5. Results and discussion. Four numerical examples are presented. The first example is used to compare the BFD scheme and the NIM. This example is also used to carry out parametric studies on time step size, grid/node size, optimum values of α and β, etc. The other three problems are chosen to investigate the evolution from different ICs, representing symmetric and asymmetric initial distributions of the population density, leading to the asymptotic solution. Neither the BFD scheme nor the NIM required cutting off the tails of the moving fronts for convergence as required by the ASD method [20, 21].

Example 1. The initial population density is 1 for $x < 0$ and zero for $x > 0$. Boundary conditions are set at 0 for $x \to \infty$, and 1 for $x \to -\infty$. The initial step-profile in this case evolves to a front with (minimum) wave speed of $c = 2$ and propagates to the right [19]. Boundary conditions for the numerical calculations are specified at $x = -20$ and $x = 80$. The calculations were repeated with boundary conditions specified at $x = -10$ and $x = 30$, and no noticeable differences were found in the results. Results of the BFD scheme at $t = 10$ for $\Delta t = 0.01$ and for various spatial grid sizes are shown in Figure 2a. At a grid size of about 0.333, the solution reaches its asymptotic behavior, and there is no noticeable change in the solution for $\Delta x = 0.25$. The profile compares very well with the second order asymptotic solution given by Canosa [19]. There are no wiggles in the solution even at very coarse grid sizes. However, the wave speed is clearly much higher than the minimum speed of 2 for simulations with large Δx. The original motivation for the BFD schemes was to take advantage of the EFD schemes of similar equations and develop FD schemes that will be free of elementary instabilities [16]. That goal for simulations with large Δx is achieved for the FE, so it seems, only by sacrificing the correct propagation speed. The effect of the time step size on accuracy is reported in Figure 2b. The BFD scheme yields fairly good results even with $\Delta t = 0.05$, and the profile obtained using $\Delta t = 0.01$ is essentially indistinguishable from that obtained with $\Delta t = 0.005$. The population density profiles at different times, obtained with $\Delta x = 0.5$ and $\Delta t = 0.01$, are presented in Figure 2c. As expected, after the initial transient, the front moves with the minimum speed of 2 to the right.

Corresponding results for the NIM are shown in Figures 3 and 4. For all NIM results in this paper, values of the time-step-averaged u variable $\bar{u}^t_{i,j}$ are plotted at discrete values of the x coordinate. Recall that these are u values at discrete x coordinates but are averaged over the jth time step. For the NIM, the role of α and β on the accuracy of the results is first investigated with $\Delta t = 0.01$ and a relatively coarse spatial grid size of $\Delta x = 1$ (All combinations of α and β studied resulted in

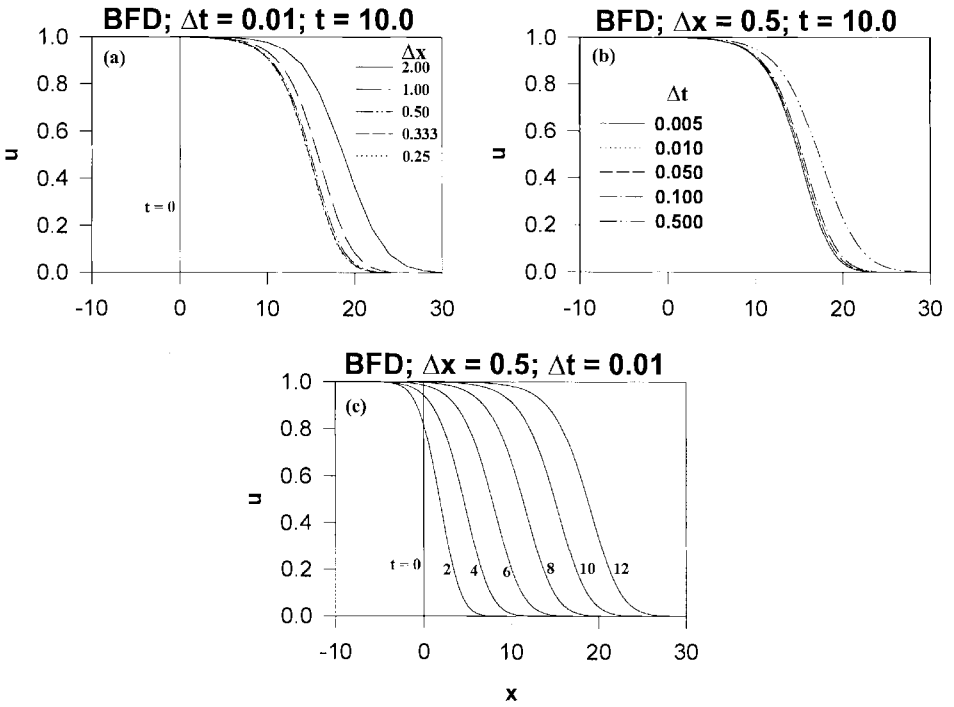


FIG. 2. *Parametric studies for the best finite difference scheme:* (a) *Grid effect,* (b) *time step size effect,* (c) *propagating front starting from a step function.*

very accurate results at fine mesh sizes.) Results in Figure 3, when compared with the fine grid solution of the BFD scheme (as well as the fine grid NIM solutions), show that the combination $(\alpha, \beta) = (-1, 2)$ yields the most accurate solution. Notice that this means that the node interior spatial solution for the time-step-averaged population density is being represented by hyperbolic functions. It is believed that this choice is dictated by the fact that local spatial segments of the propagating front are better represented (modeled) by hyperbolic functions than by trigonometric functions. Hence, for $\alpha = -1$, the (local) solution for $\bar{u}^t(x')$ is of the form

$$\bar{u}^t(x') = -\frac{\bar{S}^{t0}}{\lambda} + C_1 \sinh(\sqrt{\lambda}\, x') + C_2 \cosh(\sqrt{\lambda}\, x').$$

For problems with a more *wavy* (oscillatory) spatial solution, a positive value of α is expected to be optimal. For the rest of the calculations reported in this paper we used $(\alpha, \beta) = (-1, 2)$.

The effect of node size on the results of the NIM is shown in Figure 4a. Comparison of results in Figures 2a and 4a clearly shows that the NIM—though very slightly contaminated by an undershoot for the very large node size of 2.0—yields the same level of accuracy with $\Delta x = 1$ as the BFD scheme yields with $\Delta x = 0.25$. Recalling that there are two unknowns in the NIM for each node, compared to the single unknown per grid point for the BFD scheme, the above comparison shows that for the same number of unknowns, the results obtained using the NIM with $n_x/2$ nodes are comparable in accuracy to the results from the BFD scheme with n_x grid points.

The time step can similarly be rather large in the NIM. As shown in Figure 4b, for Δt as large as 0.2, the NIM yields very good accuracy. The "error" in the simulation

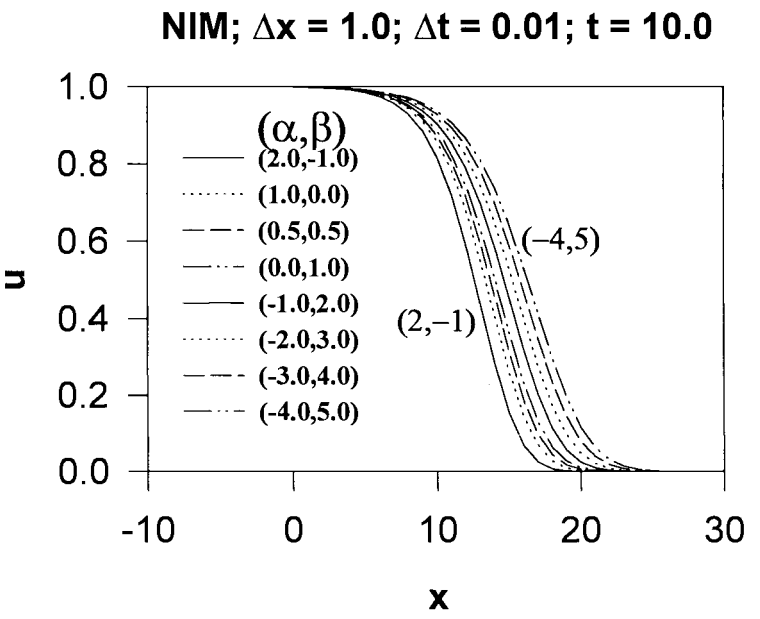


FIG. 3. *Parametric study to determine the effects of the parameters α and β on the accuracy of the numerical result for the nodal integral method. $\bar{u}^t_{i,j}$ is plotted at $t = 10$ for different combinations of α and β.*

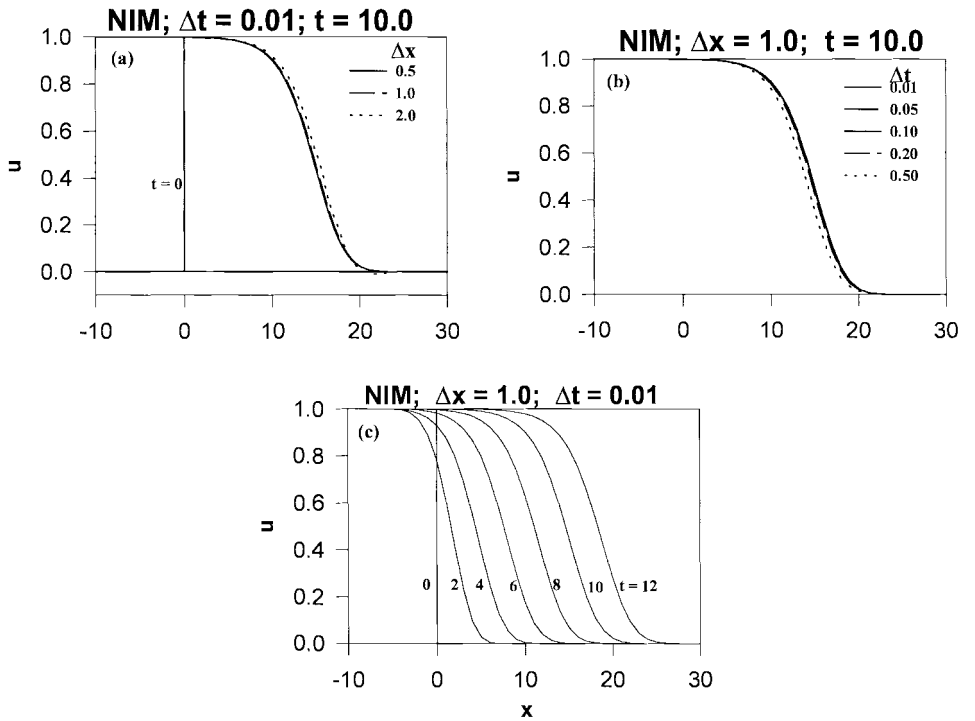


FIG. 4. *Parametric studies for the nodal integral method. Shown are values of $\bar{u}^t_{i,j}$ at discrete values of the x coordinate:* (a) *Grid effect,* (b) *time step size effect,* (c) *propagating front starting from a step function.*

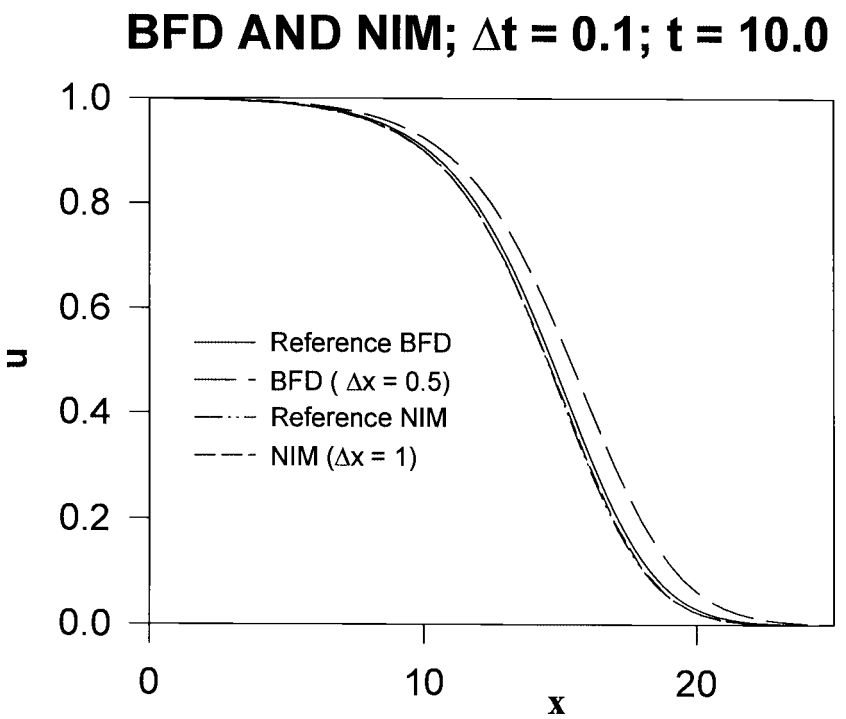


FIG. 5. *Comparison of the BFD scheme and the NIM for the propagating front problem with their respective reference solutions. For the NIM,* $\bar{u}^t_{i,j}$ *are plotted at discrete values of the* x *coordinate.*

with an even larger time-step of $\Delta t = 0.5$ is not as big as it might appear from Figure 4b. Recall that for the NIM, the discrete variable being plotted here is the u variable *averaged* over the last time-step. Hence, $\bar{u}^t_{i,j}$ for different Δt cases are averaged over different time segments: over $9.99 < t < 10.0$ for the $\Delta t = 0.01$ case, and over $9.5 < t < 10$ for the $\Delta t = 0.5$ case. Consequently, with a propagating speed of 2, the front for the latter case is *expected* to be about 0.5 units behind the former. Figure 4c shows the traveling front profile at different times obtained using the NIM. As in the BFD case, the front moves with a velocity of 2 to the right after the initial transient.

Results obtained using the BFD scheme and the NIM are compared with their corresponding reference results in Figure 5. For the BFD scheme, the population density u obtained using $\Delta t = 0.1$ and $\Delta x = 0.5$ is plotted at $t = 10$. The corresponding reference solution was obtained using the BFD scheme with $(\Delta x, \Delta t) = (0.25, 0.01)$. For the NIM, the time-step averaged population density is calculated with $\Delta t = 0.1$ and $\Delta x = 1.0$, leading to the same number of discrete unknowns as those in the BFD calculation. The reference result for the NIM was calculated with $(\Delta x, \Delta t) = (0.5, 0.01)$. (See the comment made above in reference to Figure 4b when comparing the results obtained using the NIM with two different time steps.) The propagating front calculated using the BFD scheme is moving faster than the reference solution. The NIM results, however, match very well with the corresponding reference results.

Examples 2–4. Evolution of three different ICs is studied in these problems using the NIM. All problems were solved with $\Delta x = 1.0$, $(\alpha, \beta) = (-1, 2)$, and with zero boundary conditions specified at $x = -40$ and $x = 40$.

The evolution of the initially concentrated population density of the advantageous gene, $u(x,0) = 1$ for $-1 \leq x \leq 1$ and zero elsewhere, is shown in Figure 6a. As expected, strong diffusion initially causes the population density to drop at the site of its initial concentration, but it recovers with time as a result of the (nonlinear) generation term. As time increases, two fronts of the advantageous gene, symmetric about the origin, travel to the left and right with the minimum speed.

To study the interaction of two wave fronts approaching each other, the FE is solved with ICs that are symmetric or asymmetric about the origin. The interaction of two symmetric concentrations of the population density is studied with the IC $u(x,0) = 1$ for $-5 \leq x \leq -4$ and $4 \leq x \leq 5$, and zero elsewhere. The symmetric evolution of the initial profile is shown in Figure 6b. To study the asymmetric IC and its effect on front propagation, the FE was solved with $u(x,0) = 1.0$ for $4 \leq x \leq 5$ and $u(x,0) = 0.5$ for $-5 \leq x \leq -4$, and $u(x,0) = 0$ elsewhere. The resulting population density profiles at different times are shown in Figure 6c. An interesting feature in the asymmetric IC problem is that at large times ($t > 10$), the effect of the initial asymmetry disappears and the left moving front and the right moving front have shapes and positions that are almost mirror images of each other (Figure 6c) and, more importantly, are almost the same as those at the corresponding time for the symmetric IC problem; see Figure 6b. For the asymmetric problem, the right moving front—while maintaining the evolution of its own shape and speed that matches with the right moving front in the symmetric problem—*helps* the left moving front, which is initially smaller, to gain its amplitude. This is accomplished by the asymmetry at the origin which causes diffusion from the right to the left. Being a nonzero-sum game—due to the generation term—the shape and speed of the right moving front is little affected by this *generosity*.

6. Summary and conclusions. An NIM has been developed for the FE and compared with the corresponding NSFD scheme. It was shown that the nodal integral approach is relatively more systematic than the rather ad hoc development of the NSFD scheme. The discrete variables in the two schemes are different: there are point values in the NSFD scheme and locally averaged variables in the case of the NIM. This, for the FE, leads to twice as many discrete unknowns in the NIM for the same number of grid points as in the NSFD scheme. A higher number of discrete unknowns per node, however, does not pose a limitation on the NIM due to its high accuracy even on coarse meshes.

There are many similarities in the final sets of discrete equations obtained via the two significantly different approaches. Specifically, a rearrangement of the set of discrete equations for the NIM—which is developed without explicitly approximating the derivatives as is common in the FD-type schemes—showed that the first order time- and second order space-derivatives in this approach are indeed approximated by nonstandard representations similar to those used in the NSFD scheme. Moreover, it was shown that the nodal integral scheme has some of the desirable features of the NSFD schemes such as the nonlocal evaluation of the nonlinear terms.

In reference to other possible applications, it must be pointed out that extension of the NIM to higher dimensions is fairly straightforward. Moreover, it has already been applied to sets of PDEs [2, 5, 7]. The major drawback of the (standard) NIM is that the scheme is limited to geometric domains that can be formed by a union of rectangles. This limitation results from the transverse-integration step that is used to reduce a PDE in N independent variables to a set of N ODEs. Recently developed hybrid nodal schemes for the heat conduction equation [23] and for the

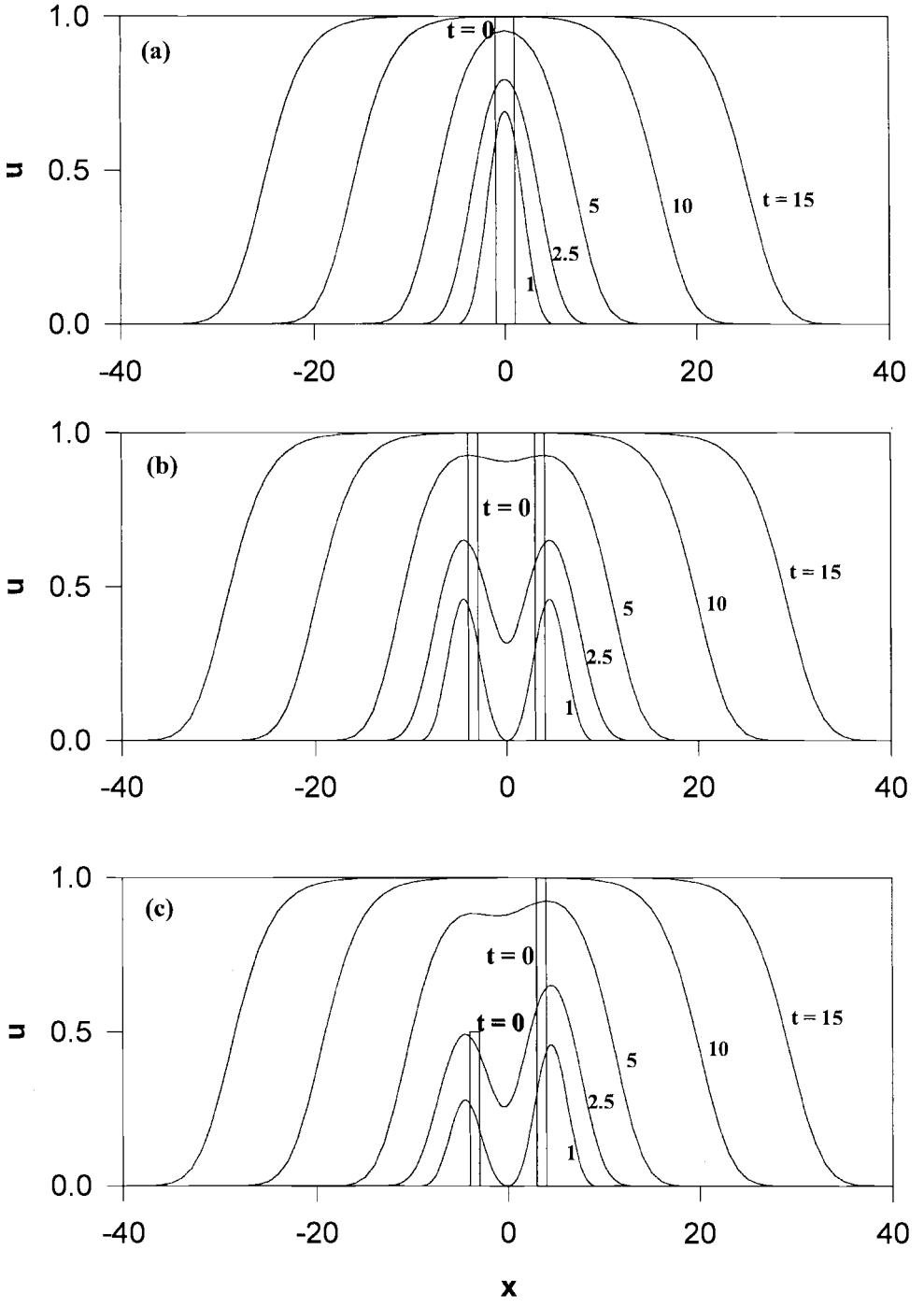


FIG. 6. *Evolution from different initial conditions (NIM results):* (a) *A single lump of concentrated gene at and around the center,* (b) *two symmetric lumps of concentrated gene,* (c) *two asymmetric lumps of concentrated gene.*

multidimensional, convection-diffusion equation [24] have, however, relaxed this restriction. Hybrid schemes applicable in irregular geometries for other sets of PDEs are under development.

Numerical results presented for the FE for various symmetric and asymmetric ICs showed that both schemes are very accurate on fine mesh sizes. The NIM, however, yields very accurate results even for a relatively coarse mesh size. In fact, results obtained using the NIM with half as many grid points (nodes) as those used in the NSFD scheme—leading to the same number of discrete unknowns in both schemes—were better than the corresponding results obtained using the NSFD scheme.

Appendix. The expressions for $\bar{u}^x_{i,j}$ and $\bar{u}^t_{i,j}$ are given by

$$\bar{u}^x_{i,j} = \bar{u}^x_{i,j-1} f_2 + f_5 f_3 \bar{S}^{x0},$$

$$\bar{u}^t_{i,j} = \frac{f_1 \left(\bar{u}^t_{i-1,j} + \bar{u}^t_{i+1,j}\right)/2 - f_{10}\bar{u}^t_{i+1,j} - \lambda\left(u^{avg\,2}_{i,j} + u^{avg\,2}_{i+1,j}\right) - (g_i + g_{i+1})}{f_{12}}$$

where

$$f_1 = \frac{\lambda\alpha}{\sin^2(\sqrt{\lambda\alpha}\,a)}, \qquad f_2 = e^{2\lambda\beta\tau}, \qquad f_3 = \frac{f_2 - 1}{2\lambda\beta\tau},$$

$$f_4 = \frac{\tan(\sqrt{\lambda\alpha}\,a)}{\sqrt{\lambda\alpha}\,a}, \qquad f_5 = 2\tau, \qquad f_6 = f_1 - 2\lambda\alpha,$$

$$f_7 = \frac{f_3 - 1}{\lambda\beta}, \qquad f_8 = \frac{f_4 - 1}{\lambda\alpha}, \qquad f_9 = f_7 + f_8,$$

$$f_{10} = \frac{f_4}{2f_9}, \qquad f_{11} = -\lambda f_8, \qquad f_{12} = f_1 - 2\lambda\alpha + f_{10},$$

$$g_i = \frac{f_4\bar{u}^t_{i-1,j}/2 + f_{11}u^{avg\,2}_{i,j} - f_3\bar{u}^x_{i,j-1}}{f_9}, \qquad \bar{S}^{x0}_{i,j} = f_{10}\bar{u}^t_{i,j} + g_i,$$

$$u^{avg}_{i,j} = \frac{f_3\bar{u}^x_{i,j-1} + f_7\left(f_{10}\bar{u}^t_{i,j} + \dfrac{f_4\bar{u}^t_{i-1,j} - f_3\bar{u}^x_{i,j-1}}{f_9}\right)}{1 - (f_7 f_{11}/f_9)\,u^{avg}_{i,j}}.$$

Acknowledgment. The author would like to acknowledge the support provided by the Center for Simulation of Advanced Rockets, University of Illinois at Urbana-Champaign. The Center is supported by the US Department of Energy through the University of California under subcontract B341494.

REFERENCES

[1] S. Langenbuch, W. Maurer, and W. Werner, *Coarse-mesh flux-expansion method for the analysis of space-time effects in large light water reactor cores*, Nuclear Science and Engineering, 63 (1977), pp. 437–456.

[2] H. D. Fischer and H. Finnemann, *The nodal integration method—a diverse solver for neutron diffusion problems*, Atomkernenergie, Kerntechnik, 39 (1981), pp. 229–236.

[3] W. C. Horak and J. J. Dorning, *A nodal coarse-mesh method for the efficient numerical solution of laminar flow problems*, J. Comput. Phys., 59 (1985), pp. 405–440.
[4] R. D. Lawrence and J. J. Dorning, *A nodal Green's function method for multidimensional neutron diffusion calculations*, Nuclear Science and Engineering, 76 (1980), pp. 218–231.
[5] Y. Y. Azmy and J. J. Dorning, *A nodal integral approach to the numerical solution of partial differential equations*, in Advances in Reactor Computations, volume II, American Nuclear Society, LaGrange Park, IL, 1983, pp. 893–909.
[6] J. P. Hennart, *A general family of nodal schemes*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 264–287.
[7] G. L. Wilson, R. A. Rydin, and Y. Y. Azmy, *Time-dependent nodal integral method for the investigation of bifurcation and nonlinear phenomena in fluid flow and natural convection*, Nuclear Science and Engineering, 100 (1988), pp. 414–425.
[8] J. P. Hennart, *On the numerical analysis of analytical nodal methods*, Numer. Methods Partial Differential Equations, 4 (1988), pp. 233–254.
[9] O. A. Elnawawy, A. J. Valocchi, and A. M. Ougouag, *The cell analytical-numerical method for solution of the advection-dispersion equation: Two-dimensional problems*, Water Resources Research, 26 (1990), pp. 2705–2716.
[10] Rizwan-uddin, *An improved coarse mesh nodal integral method for partial differential equations*, Numer. Methods Partial Differential Equations, 13 (1997), pp. 113–145.
[11] Rizwan-uddin, *A second order space and time nodal method for the one-dimensional convection-diffusion equation*, Comput. & Fluids, 26 (1997), pp. 233–247.
[12] R. B. Potts, *Differential and difference equations*, Amer. Math. Monthly, 89 (1982), pp. 402–407.
[13] R. E. Mickens, *Properties of finite-difference models of nonlinear conservative oscillators*, J. Sound Vibration, 124 (1988), pp. 194–198.
[14] R. E. Mickens, *Exact solutions to a finite-difference model of a nonlinear reaction-advection equation: Implications for numerical analysis*, Numer. Methods Partial Differential Equations, 5 (1989), pp. 313–325.
[15] R. E. Mickens, *Nonstandard Finite Difference Models of Differential Equations*, World Scientific, Singapore, 1994.
[16] R. E. Mickens, *A best finite-difference scheme for the Fisher equation*, Numer. Methods Partial Differential Equations, 10 (1994), pp. 581–585.
[17] R. A. Fisher, *The wave of advance of advantageous genes*, Ann. Eugen., 7 (1936), pp. 355–369.
[18] A. Kolmogoroff, I. Petrovsky, and N. Piscounoff, *Etude de l'equation de la diffusion avec croissance de la quantité de matière et son application à un probleme biologique*, Bull. Univ. Etat Moscou, Ser. Int. Sect. A, 1 (1937), pp. 1–25.
[19] J. Canosa, *On a nonlinear diffusion equation describing population growth*, IBM J. Res. Develop., 17 (1973), pp. 307–313.
[20] J. Gazdag and J. Canosa, *Numerical solution of Fisher's equation*, J. Appl. Probab., 11 (1974), pp. 445–457.
[21] J. Gazdag, *Numerical convective schemes based on accurate computation of space derivatives*, J. Comput. Phys., 13 (1973), pp. 100–113.
[22] P. L. Sachdev, *Nonlinear Diffusive Waves*, Cambridge University Press, New York, 1987.
[23] Rizwan-uddin, *A hybrid nodal-integral/finite-element method for arbitrary geometries*, Trans. Am. Nuc. Soc., 76 (1997), pp. 170–172.
[24] Allen J. Toreja and Rizwan-uddin, *Hybrid numerical methods for the convection-diffusion equation in arbitrary geometries*, in Proceedings of the M & C '99—International Conference on Mathematics and Computation, Reactor Physics and Environmental Analysis in Nuclear Applications, Madrid, Spain, 1999, J. M. Aragones, ed., Senda Editorial, Madrid, 1999, pp. 1705–1714.

SIAM J. SCI. COMPUT.
Vol. 22, No. 6, pp. 1943–1961
© 2001 Society for Industrial and Applied Mathematics

FAST FINITE VOLUME SIMULATION OF 3D ELECTROMAGNETIC PROBLEMS WITH HIGHLY DISCONTINUOUS COEFFICIENTS*

E. HABER† AND U. M. ASCHER‡

Abstract. We consider solving three-dimensional electromagnetic problems in parameter regimes where the quasi-static approximation applies and the permeability, permittivity, and conductivity may vary significantly. The difficulties encountered include handling solution discontinuities across interfaces and accelerating convergence of traditional iterative methods for the solution of the linear systems of algebraic equations that arise when discretizing Maxwell's equations in the frequency domain.

The present article extends methods we proposed earlier for constant permeability [E. Haber, U. Ascher, D. Aruliah, and D. Oldenburg, *J. Comput. Phys.,* 163 (2000), pp. 150–171; D. Aruliah, U. Ascher, E. Haber, and D. Oldenburg, *Math. Models Methods Appl. Sci.*, to appear.] to handle also problems in which the permeability is variable and may contain significant jump discontinuities. In order to address the problem of slow convergence we reformulate Maxwell's equations in terms of potentials, applying a Helmholtz decomposition to either the electric field or the magnetic field. The null space of the **curl** operators can then be annihilated by adding a stabilizing term, using a gauge condition, and thus obtaining a strongly elliptic differential operator. A staggered grid finite volume discretization is subsequently applied to the reformulated PDE system. This scheme works well for sources of various types, even in the presence of strong material discontinuities in both conductivity and permeability. The resulting discrete system is amenable to fast convergence of ILU-preconditioned Krylov methods.

We test our method using several numerical examples and demonstrate its robust efficiency. We also compare it to the classical Yee method using similar iterative techniques for the resulting algebraic system, and we show that our method is significantly faster, especially for electric sources.

Key words. Maxwell's equations, solution discontinuities, Helmholtz decomposition, Coulomb gauge, finite volume, Krylov methods, mixed methods, preconditioning

AMS subject classifications. 65N06, 65N22

PII. S1064827599360741

1. Introduction.

The need for calculating fast, accurate solutions of three-dimensional electromagnetic equations arises in many important application areas including, among others, geophysical surveys and medical imaging [34, 40, 3]. Consequently, a lot of effort has recently been invested in finding appropriate numerical algorithms. However, while it is widely agreed that electromagnetic phenomena are generally governed by Maxwell's equations, the choice of numerical techniques to solve these equations depends on parameter ranges and various other restrictive assumptions, and as such is to a significant degree application-dependent [24, 40, 3].

The present article is motivated by remote sensing inverse problems, e.g., in geophysics, where one seeks to recover material properties—especially conductivity—in an isotropic but heterogeneous body, based on measurements of electric and magnetic fields on or near the earth's surface. The forward model, on which we concentrate here, consists of Maxwell's equations in the frequency domain over a frequency range which excludes high frequencies. Assuming a time-dependence $e^{-\imath\omega t}$, these equations

*Received by the editors August 31, 1999; accepted for publication (in revised form) August 10, 2000; published electronically February 21, 2001.
http://www.siam.org/journals/sisc/22-6/36074.html

†Departments of Computer Science and of Earth and Oceanography Sciences, University of British Columbia, Vancouver, BC, V6T 1Z4, Canada (haber@cs.ubc.ca).

‡Department of Computer Science, University of British Columbia, Vancouver, BC, V6T 1Z4, Canada (ascher@cs.ubc.ca).

are written as

$$\boldsymbol{\nabla} \times \boldsymbol{E} - \imath\omega\mu\boldsymbol{H} = \boldsymbol{0}, \tag{1.1a}$$

$$\boldsymbol{\nabla} \times \boldsymbol{H} - \widehat{\sigma}\boldsymbol{E} = \boldsymbol{J}^s, \tag{1.1b}$$

$$\nabla \cdot (\epsilon \boldsymbol{E}) - \rho = 0, \tag{1.1c}$$

$$\nabla \cdot (\mu \boldsymbol{H}) = 0, \tag{1.1d}$$

where μ is the magnetic permeability, σ is the conductivity, ϵ is the electrical permittivity,

$$\widehat{\sigma} = \sigma - \imath\omega\epsilon, \tag{1.2}$$

$\boldsymbol{J}^s$ is a known source current density, and ρ is the (unknown) volume density of free charges. In our work we assume that the physical properties $\mu > 0$, $\epsilon > 0$, and $\sigma \geq 0$ can vary with position, and $\mu\epsilon\omega^2 L^2 \ll 1$, where L is a typical length scale (cf. [43]). The electric field $\boldsymbol{E}$ and the magnetic field $\boldsymbol{H}$ are the unknowns, with the charge density defined by (1.1c). Note that as long as $\omega \neq 0$, (1.1d) is redundant and can be viewed as an invariant of the system, obtained by taking the $\nabla\cdot$ of (1.1a). The system (1.1) is defined over a three-dimensional spatial domain Ω. In principle, the domain Ω is unbounded (i.e., $\Omega = \mathbb{R}^3$), but, in practice, a bounded subdomain of $\mathbb{R}^3$ is used for numerical approximations. In this paper we have used the boundary conditions (BCs)

$$\boldsymbol{H} \times \boldsymbol{n}\Big|_{\partial\Omega} = 0, \tag{1.3}$$

although other BCs are possible.

A number of difficulties arises when attempting to find numerical solutions for this three-dimensional PDE system. These difficulties include handling regions of (almost) vanishing conductivity, handling different resolutions in different parts of the spatial domain, handling the multiple scale lengths over which the physical properties can vary, and handling regions of highly varying conductivity, magnetic permeability, or electrical permittivity, where jumps in solution properties across interfaces may occur.

On the other hand, the nature of the data (e.g., measurements of the electric and/or magnetic fields at the surface of the earth) is such that one cannot hope to recover to a very fine detail the structure of the conductivity σ or the permeability μ. We therefore envision, in accordance with the inverse problem of interest, a possibly nonuniform tensor product grid covering the domain Ω, where $\widehat{\sigma}$ and μ are assumed to be smooth or even constant inside each grid cell, but they may have significant jump discontinuities that can occur anywhere in Ω across cell interfaces. The source $\boldsymbol{J}^s$ is, however, assumed not to have jumps across interfaces. The relative geometric simplicity resulting from this modeling assumption is key in obtaining highly efficient solvers for the forward problem.

Denoting quantities on different sides of an interface by subscripts 1 and 2, it can be easily shown [41] that across an interface

$$\boldsymbol{n} \cdot (\widehat{\sigma}_1 \boldsymbol{E}_1 - \widehat{\sigma}_2 \boldsymbol{E}_2) = 0, \tag{1.4a}$$

$$\boldsymbol{n} \times (\boldsymbol{E}_1 - \boldsymbol{E}_2) = 0, \tag{1.4b}$$

$$\boldsymbol{n} \cdot (\mu_1 \boldsymbol{H}_1 - \mu_2 \boldsymbol{H}_2) = 0, \tag{1.4c}$$

$$\boldsymbol{n} \times (\boldsymbol{H}_1 - \boldsymbol{H}_2) = 0. \tag{1.4d}$$

These conditions imply that neither $\boldsymbol{E}$ nor $\boldsymbol{H}$ are continuous in the normal direction when $\widehat{\sigma}$ and μ have a jump discontinuity across a cell face, and likewise, $\widehat{\sigma}\boldsymbol{E}$ and $\mu\boldsymbol{H}$ are not necessarily continuous in tangential directions. Care must therefore be exercised when numerical methods are employed which utilize these variables if they are to be defined where they are double-valued.

By far the most popular discretization for Maxwell's equations is Yee's method [46] (see discussions and extensions in [40, 30, 21]). This method employs a staggered grid, necessitating only short, centered first differences to discretize (1.1a) and (1.1b). In the more usual application of this method, the electric field components are envisioned on the cell's edges and the magnetic field components are on the cell's faces—see Figure 1. It is further possible to eliminate the components of the magnetic field from the

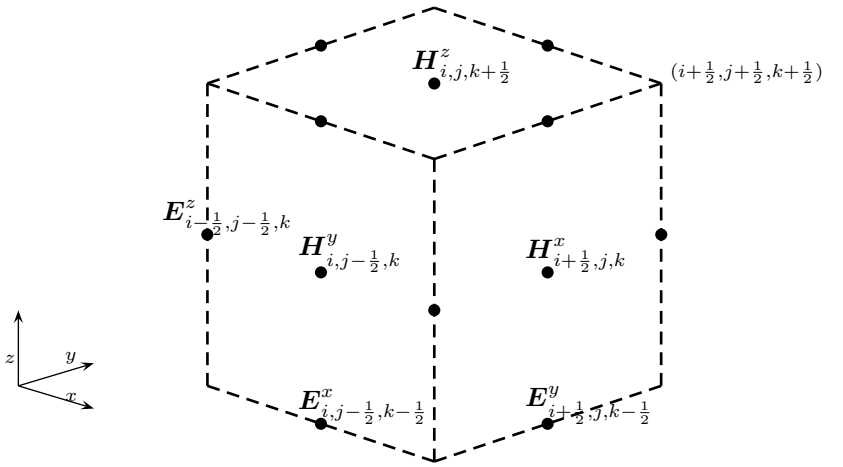


FIG. 1. *A staggered discretization of* $\boldsymbol{E}$ *and* $\boldsymbol{H}$ *in three dimensions:* $\boldsymbol{E}$*-components are on the edges, and* $\boldsymbol{H}$*-components are on the faces.*

discrete equations, obtaining a staggered discretization for the second order PDE in $\boldsymbol{E}$,

$$\boldsymbol{\nabla} \times (\mu^{-1}\boldsymbol{\nabla} \times \boldsymbol{E}) - \imath\omega\widehat{\sigma}\boldsymbol{E} = \imath\omega\boldsymbol{J}^{\boldsymbol{s}}. \tag{1.5}$$

Related methods include the finite integration technique and certain mixed finite element methods [44, 6, 29, 16]. Although these methods are often presented in the context of time-domain Maxwell's equations, the issues arising when applying an implicit time-discretization (a suitable technique under our model assumptions) are somewhat similar to the ones we are faced with here.

The popularity of Yee's method is due in part to its conservation properties and other ways in which the discrete system mimics the continuous system [21, 17, 6, 5]. However, iterative methods to solve the discrete system may converge slowly in low frequencies, due to the presence of the nontrivial null space of the **curl** operator, and additional difficulties arise when highly discontinuous coefficients are present [34, 28, 38, 19]. There are two major reasons for these difficulties. First, the conductivity can essentially vanish (for example, in the air, which forms part of Ω); from an analytic perspective, the specific subset of Maxwell's equations used typically forms an almost-singular system in regions of almost-vanishing $\widehat{\sigma}$. Even in regions where the conductivity is not close to vanishing, the resulting differential operator is strongly

coupled and not strongly elliptic [7, 2]. Second, in cases of large jump discontinuities, care must be taken to handle $\boldsymbol{H}$ and $\widehat{\boldsymbol{J}} = \widehat{\sigma}\boldsymbol{E}$ carefully, since these are located, as in Figure 1, where they are potentially discontinuous.

In [2], we addressed the often slow convergence of iterative methods when used for the equations resulting from the discretization of (1.5) by applying a Helmholtz decomposition first, obtaining a potential formulation with a Coulomb gauge condition. This change of variables (used also in [4, 13, 26, 32], among many others) splits the electric field into components in the active and the null spaces of the **curl** operator. A further reformulation, reminiscent of the pressure-Poisson equation for the incompressible Navier–Stokes equations [15, 37], yields a system of strongly elliptic, weakly coupled PDEs for which more standard preconditioned Krylov space methods are directly applicable.

In [17], we further addressed possible significant jumps in the conductivity while μ is assumed constant by employing a finite volume discretization on a staggered grid, akin to Yee's method with the locations of $\boldsymbol{E}$- and $\boldsymbol{H}$-components exchanged, as in Figure 2. The normal components of $\boldsymbol{E}$ are now double-valued, but this is taken care

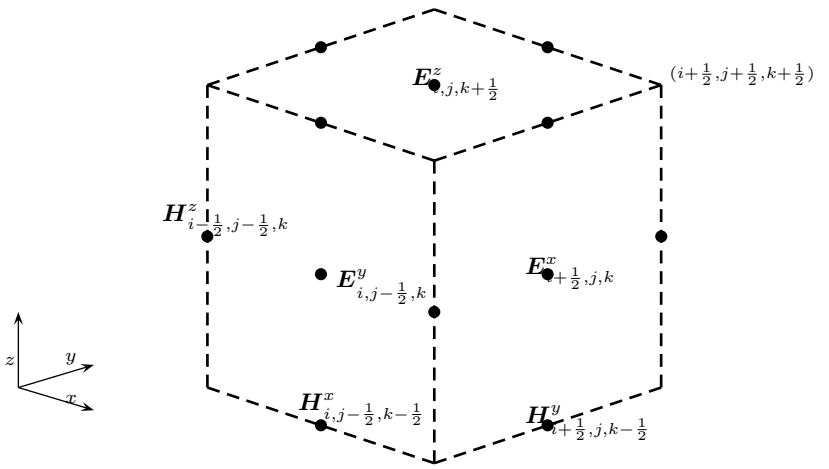


FIG. 2. *A staggered discretization of* $\boldsymbol{E}$ *and* $\boldsymbol{H}$ *in three dimensions:* $\boldsymbol{E}$*-components are on the faces, and* $\boldsymbol{H}$*-components are on the edges.*

of in an elegant way by the Helmholtz decomposition of $\boldsymbol{E}$ and by introducing the (generalized) current

$$\widehat{\boldsymbol{J}} = \widehat{\sigma}\boldsymbol{E} = (\sigma - \imath\omega\epsilon)\boldsymbol{E} \tag{1.6}$$

into the equations. The **curl** operators in (1.5) are replaced by the vector Laplacian according to the vector identity

$$\boldsymbol{\nabla} \times \boldsymbol{\nabla} \times = -\nabla^2 I + \operatorname{grad} \nabla \cdot \tag{1.7}$$

for sufficiently smooth vector functions (not $\boldsymbol{E}$).

In this paper we generalize our approach from [2, 17] to the case where the magnetic permeability μ may be highly discontinuous as well. This is a realistic case of interest in geophysical applications, although usually the jump in conductivity dominates the jump in permeability. Now the roles of $\boldsymbol{E}$ and $\boldsymbol{H}$ are essentially dual, and it is possible to apply a Helmholtz decomposition to either $\boldsymbol{E}$ or $\boldsymbol{H}$, keeping the other

unknown vector function intact. We choose to decompose the electric field $\boldsymbol{E}$, referring to Figure 2 for the locations of the $\boldsymbol{H}$-unknowns in the ensuing discretization. The major departure from our previous work is in the fact that the identity (1.7) does not directly extend for the operator $\boldsymbol{\nabla}\times(\mu^{-1}\boldsymbol{\nabla}\times)$ appearing in (1.5). We can, however, stabilize this operator by subtracting $\mathrm{grad}(\mu^{-1}\nabla\cdot)$ (see, e.g., [24]), and this forms the basis for our proposed method. In cases of constant magnetic permeability or electric conductivity the formulation can be reduced to our previous formulation in [17] or a variant thereof.

Our approach to dealing with possible discontinuities can be viewed as using a set of variables which are continuous across cell faces and another set which are continuous along cell edges. The introduction of such unknowns is strongly connected to mixed finite elements which are used for highly discontinuous problems [8, 9, 19].

The paper is laid out as follows. In section 2, we reformulate Maxwell's equations in a way which enables us to extend our methods. The resulting system is amenable to discretization using a finite volume technique described in section 3.

The extension and generalization of our approach from [2] through [17] to the present article is not without a price. This price is an added complication in the sparsity structure of the resulting discrete system and a corresponding added cost in the iterative solution of such systems. We briefly describe the application of Krylov space methods to solve the system of algebraic equations in section 4. We use BICGSTAB (see, e.g., [35]) together with one of two preconditioners: an incomplete block LU decomposition (which is a powerful preconditioner in the case of diagonally dominant linear systems) and SSOR. The system's diagonal dominance is a direct consequence of our analytic formulation.

We present the results of numerical experiments in section 5 and compare results obtained using our method with those obtained using a more traditional Yee discretization. If the source is not divergence-free, as is the case for electric (but not magnetic) sources, then our method is better by more than two orders of magnitude. The method works well also for a case where the problem coefficients vary rapidly. We conclude with a short summary and further remarks.

2. Reformulation of Maxwell's equations. Maxwell's equations (1.1a) and (1.1b) can be viewed as flux balance equations, i.e., each term describes the flux which arises from a different physical consideration, and the equations are driven by the conservation of fluxes. (In fact, this was how they were originally developed [27].) Therefore, in both (1.1a) and (1.1b) we have a flux term which should be balanced. In (1.1b) the *generalized current density* $\widehat{\boldsymbol{J}}$ defined in (1.6) is balanced with the source and the flux which arise from magnetic fields, and in (1.1a) the *magnetic flux*

$$\boldsymbol{B} = \mu\boldsymbol{H} \tag{2.1}$$

is balanced with the flux which arises from electric fields. In our context these fluxes are well defined on cell faces, but they may be multivalued at cell edges.[1]

Furthermore, the leading differential operator in (1.5), say, has a nontrivial null space. Rather than devising iterative methods which directly take this into account (as, e.g., in [3, 19, 39, 33]), we transform the equations before discretization.

We decompose $\boldsymbol{E}$ into its components in the active space and in the null space of

[1] Under appropriate conditions, $\boldsymbol{B}, \widehat{\boldsymbol{J}} \in H(div;\Omega)$, whereas $\boldsymbol{E}, \boldsymbol{H} \in H(\mathbf{curl};\Omega)$; see, e.g., [14].

the **curl** operator:

$$\boldsymbol{E} = \boldsymbol{A} + \operatorname{grad}\phi, \tag{2.2a}$$

$$\nabla \cdot \boldsymbol{A} = 0. \tag{2.2b}$$

We could decompose $\boldsymbol{H}$ instead in a similar way, but we would not decompose both. Here we have chosen to concentrate on the decomposition of $\boldsymbol{E}$. Substituting (2.2) into Maxwell's equations, we obtain

$$\boldsymbol{\nabla} \times \boldsymbol{A} - \imath\omega\mu\boldsymbol{H} = \boldsymbol{0}, \tag{2.3a}$$

$$\boldsymbol{\nabla} \times \boldsymbol{H} - \widehat{\sigma}(\boldsymbol{A} + \operatorname{grad}\phi) = \boldsymbol{J^s}, \tag{2.3b}$$

$$\nabla \cdot \boldsymbol{A} = 0. \tag{2.3c}$$

Furthermore, (1.5) becomes

$$\boldsymbol{\nabla} \times (\mu^{-1}\boldsymbol{\nabla} \times \boldsymbol{A}) - \imath\omega\widehat{\sigma}(\boldsymbol{A} + \operatorname{grad}\phi) = \imath\omega\boldsymbol{J^s}, \tag{2.4a}$$

$$\nabla \cdot \boldsymbol{A} = 0. \tag{2.4b}$$

Note that across an interface between distinct conducting media we have, in addition to (1.4),

$$\boldsymbol{n} \times (\boldsymbol{A}_1 - \boldsymbol{A}_2) = 0, \tag{2.5a}$$

$$\boldsymbol{n} \cdot (\boldsymbol{A}_1 - \boldsymbol{A}_2) = 0, \tag{2.5b}$$

$$\boldsymbol{n} \cdot (\epsilon_1 \operatorname{grad}\phi_1 - \epsilon_2 \operatorname{grad}\phi_2) = \rho_s, \tag{2.5c}$$

$$\boldsymbol{n} \cdot (\widehat{\boldsymbol{J}}_1 - \widehat{\boldsymbol{J}}_2) = 0, \tag{2.5d}$$

where ρ_s in (2.5c) is an electric surface charge density. These conditions and the differential equations (1.1) imply that while $\widehat{\boldsymbol{J}} \cdot \boldsymbol{n}$ is continuous, $\boldsymbol{E} \cdot \boldsymbol{n}$ is not. Moreover, $\operatorname{grad}\phi$ inherits the discontinuity of $\boldsymbol{E} \cdot \boldsymbol{n}$, while $\boldsymbol{A}$ is continuous, and both $\nabla \cdot \boldsymbol{A}$ and $\boldsymbol{\nabla} \times \boldsymbol{A}$ are bounded (cf. [14]).

In [17] we had the relation $\boldsymbol{\nabla} \times \boldsymbol{\nabla} \times \boldsymbol{A} = -\nabla^2\boldsymbol{A}$ holding. However, when μ varies, the identity (1.7) does not extend directly, and we must deal with the null space of $\boldsymbol{\nabla} \times \boldsymbol{A}$ in a different way.

Let us define the Sobolev spaces

$$\boldsymbol{W}(\Omega) = \{\boldsymbol{v} \in [L^2(\Omega)]^3;\ \boldsymbol{\nabla} \times \boldsymbol{v} \in [L^2(\Omega)]^3, \nabla \cdot \boldsymbol{v} \in L^2(\Omega)\} \tag{2.6a}$$

equipped with the usual norm (see, e.g., [14])

$$\|\boldsymbol{v}\|_{\boldsymbol{W}(\Omega)} = \{\|\boldsymbol{v}\|^2 + \|\boldsymbol{\nabla} \times \boldsymbol{v}\|^2 + \|\nabla \cdot \boldsymbol{v}\|^2\}^{1/2} \tag{2.6b}$$

(the $L^2(\Omega)$- and $[L^2(\Omega)]^3$- norms are used on the right-hand side of (2.6b)), and

$$\boldsymbol{W}_0(\Omega) = \left\{\boldsymbol{v} \in \boldsymbol{W}(\Omega);\ \nabla \cdot \boldsymbol{v}\Big|_{\partial\Omega} = 0,\ \boldsymbol{\nabla} \times \boldsymbol{v} \times \boldsymbol{n}\Big|_{\partial\Omega} = \boldsymbol{0}\right\}. \tag{2.6c}$$

Green's formula yields, for any $\boldsymbol{u} \in \boldsymbol{W}(\Omega)$, $\boldsymbol{v} \in \boldsymbol{W}_0(\Omega)$,

$$\begin{aligned}&(\boldsymbol{\nabla} \times (\mu^{-1}\boldsymbol{\nabla} \times \boldsymbol{u}), \boldsymbol{v}) - (\operatorname{grad}(\mu^{-1}\nabla \cdot \boldsymbol{u}), \boldsymbol{v})\\ &= \mu^{-1}[(\boldsymbol{\nabla} \times \boldsymbol{u}, \boldsymbol{\nabla} \times \boldsymbol{v}) + (\nabla \cdot \boldsymbol{u}, \nabla \cdot \boldsymbol{v})]\end{aligned}$$

(see, e.g., [24]), where the usual notation for inner product in $L^2(\Omega)$ and $[L^2(\Omega)]^3$ is used. Thus, for any $\boldsymbol{u} \in \boldsymbol{W}_0(\Omega)$, if $\boldsymbol{u} \neq \boldsymbol{0}$, then

$$(\boldsymbol{\nabla} \times (\mu^{-1}\boldsymbol{\nabla} \times \boldsymbol{u}), \boldsymbol{u}) - (\operatorname{grad}(\mu^{-1}\nabla \cdot \boldsymbol{u}), \boldsymbol{u}) > 0.$$

We may therefore stabilize (2.4a) by subtracting a term which by (2.4b) vanishes:

$$\boldsymbol{\nabla} \times (\mu^{-1}\boldsymbol{\nabla} \times \boldsymbol{A}) - \operatorname{grad}(\mu^{-1}\nabla \cdot \boldsymbol{A}) - \imath\omega\widehat{\sigma}(\boldsymbol{A} + \operatorname{grad}\phi) = \imath\omega\boldsymbol{J}^{\boldsymbol{s}}, \tag{2.7}$$

obtaining a strongly elliptic operator for $\boldsymbol{A}$, provided $\boldsymbol{A} \in \boldsymbol{W}_0(\Omega)$. The latter condition is guaranteed by the choice (2.10b) together with (2.10a), as elaborated below.

A similar stabilization (or penalty) approach was studied with mixed success in [11, 24]. However, our experience and thus our recommendation are more positive because of the discretization we utilize. We elaborate further upon this point in the next section.

Using (2.3c), we can write (2.3b) as

$$\boldsymbol{\nabla} \times \boldsymbol{H} - \operatorname{grad}\psi - \widehat{\boldsymbol{J}} = \boldsymbol{J}^{\boldsymbol{s}}, \tag{2.8a}$$

$$\psi - \mu^{-1}\nabla \cdot \boldsymbol{A} = 0. \tag{2.8b}$$

This may be advantageous in the case of discontinuous μ, similarly to the mixed formulation used for the simple *div-grad* system

$$\nabla \cdot (\sigma \operatorname{grad}\phi) = q$$

in [8, 31, 9, 12] and elsewhere.

Our final step in the reformulation is to replace, as in [17], the gauge condition (2.3c) on $\boldsymbol{A}$ by an indirect one, obtained upon taking $\nabla\cdot$ of (2.8a) and simplifying using (2.8b) and (2.3c). This achieves only a weak coupling in the resulting PDE system. We note that the replacing of the gauge condition (2.3c) is similar to the pressure-Poisson equation in computational fluid dynamics [37, 15]. The complete system, to be discretized in the next section, can now be written as

$$\boldsymbol{\nabla} \times \boldsymbol{H} - \operatorname{grad}\psi - \widehat{\boldsymbol{J}} = \boldsymbol{J}^{\boldsymbol{s}}, \tag{2.9a}$$

$$\imath\omega\mu\boldsymbol{H} - \boldsymbol{\nabla} \times \boldsymbol{A} = \boldsymbol{0}, \tag{2.9b}$$

$$\mu\psi - \nabla \cdot \boldsymbol{A} = 0, \tag{2.9c}$$

$$-\nabla \cdot \widehat{\boldsymbol{J}} = \nabla \cdot \boldsymbol{J}^{\boldsymbol{s}}, \tag{2.9d}$$

$$\widehat{\boldsymbol{J}} - \widehat{\sigma}(\boldsymbol{A} + \operatorname{grad}\phi) = \boldsymbol{0}. \tag{2.9e}$$

In order to complete the specification of this system, we must add appropriate BCs. First, we note that the original BC (1.3) can be written as

$$(\boldsymbol{\nabla} \times \boldsymbol{A}) \times \boldsymbol{n}\Big|_{\partial\Omega} = \boldsymbol{0}. \tag{2.10a}$$

An additional BC on the normal components of $\boldsymbol{A}$ is required for the Helmholtz decomposition (2.2) to be unique. Here we choose (corresponding to (2.6c))

$$\boldsymbol{A} \cdot \boldsymbol{n}\Big|_{\partial\Omega} = 0. \tag{2.10b}$$

This, together with (2.2), determines $\boldsymbol{A}$ for a given $\boldsymbol{E}$.

Moreover, since (2.9d) was obtained by taking the $\nabla\cdot$ of (2.9a), additional BCs are required on either $\partial\phi/\partial n$ or ϕ. For this we note that the original BC (1.3) together with the PDE (1.1b) imply also

$$\widehat{\boldsymbol{J}}\cdot\boldsymbol{n}\Big|_{\partial\Omega} = -\boldsymbol{J^s}\cdot\boldsymbol{n}\Big|_{\partial\Omega}. \tag{2.10c}$$

The latter relation (2.10c), together with (2.10b), implies $\frac{\partial\phi}{\partial n}=0$ at the boundary.[2,3]

The above conditions determine ϕ up to a constant. We pin this constant down arbitrarily, e.g., by requiring

$$\int_\Omega \phi\, dV = 0. \tag{2.10d}$$

Finally, we note that (2.10c), together with (2.9a) and (1.3), imply, in turn, that $\frac{\partial\psi}{\partial n}=0$ at the boundary. Since (2.9a) and (2.9d) imply that

$$\nabla^2\psi = 0,$$

we obtain $\psi\equiv 0$, and thus we retrieve (2.3c) by pinning ψ down to 0 at one additional point. From this and (2.10a) it then follows that $\boldsymbol{A}\in\boldsymbol{W}_0(\Omega)$ (see (2.6c)).

The system (2.9) subject to the BCs (2.10) and ψ pinned at one point is now well-posed.

3. Deriving a discretization. As in [17], we employ a finite volume technique on a staggered grid, where $\widehat{\boldsymbol{J}}$ and $\boldsymbol{A}$ are located at the cell's faces, $\boldsymbol{H}$ is at the cell's edges, and ϕ and ψ are located at the cell's center. Correspondingly, the discretizations of (2.9d) and (2.9c) are centered at cell centers, those of (2.9e) and (2.9a) are centered at cell faces, and that of (2.9b) is centered at cell edges. The variables distribution over the grid cell is summarized in Table 1.

TABLE 1

Summary of the discrete grid functions. Each scalar field is approximated by the grid functions at points slightly staggered in each cell $e_{i,j,k}$ of the grid.

$$\begin{array}{cc}
A^x{}_{i+\frac12,j,k} \approx A^x(x_{i+\frac12},y_j,z_k) & \hat{J}^x_{i+\frac12,j,k} \approx \hat{J}^x(x_{i+\frac12},y_j,z_k)\\
A^y{}_{i,j+\frac12,k} \approx A^y(x_i,y_{j+\frac12},z_k) & \hat{J}^y_{i,j+\frac12,k} \approx \hat{J}^y(x_i,y_{j+\frac12},z_k)\\
A^z{}_{i,j,k+\frac12} \approx A^z(x_i,y_j,z_{k+\frac12}) & \hat{J}^z_{i,j,k+\frac12} \approx \hat{J}^z(x_i,y_j,z_{k+\frac12})\\
H^x_{i,j+\frac12,k+\frac12} \approx H^x(x_i,y_{j+\frac12},z_{k+\frac12}) & \\
H^y_{i+\frac12,j,k+\frac12} \approx H^y(x_{i+\frac12},y_j,z_{k+\frac12}) & \\
H^z_{i+\frac12,j+\frac12,k} \approx H^z(x_{i+\frac12},y_{j+\frac12},z_k) & \\
\psi_{i,j,k} \approx \psi(x_i,y_j,z_k) & \\
\phi_{i,j,k} \approx \phi(x_i,y_j,z_k) &
\end{array}$$

To approximate $\nabla\cdot\boldsymbol{u}$ for $\boldsymbol{u}=(u^x,u^y,u^z)^T$ over a grid cell $e_{i,j,k}$, we integrate at first over the cell using the Gauss divergence theorem

$$\frac{1}{|e_{i,j,k}|}\int_{e_{i,j,k}} \nabla\cdot\boldsymbol{u}\, dV = \frac{1}{|e_{i,j,k}|}\int_{\partial e_{i,j,k}} \boldsymbol{u}\cdot\boldsymbol{n}\, dS,$$

[2] In cases where the original BC is different from (1.3) we still use the BC $\partial\phi/\partial n=0$ as an asymptotic value for an infinite domain. Alternatively, note the possibility of applying the Helmholtz decomposition to $\boldsymbol{H}$, although generally there are good practical reasons to prefer the decomposition (2.2) of $\boldsymbol{E}$ (for one, because the jumps in σ are typically much larger than in μ—see section 5).

[3] In our geophysical applications, $\boldsymbol{J^s}\cdot\boldsymbol{n}$ vanishes at the boundary.

and then we use midpoint quadrature on each face to evaluate each component of the surface integrals appearing on the right-hand side above. Thus, define

$$\nabla_{i,j,k} \cdot \boldsymbol{u} = \frac{u^x_{i+\frac{1}{2},j,k} - u^x_{i-\frac{1}{2},j,k}}{h^x_i} + \frac{u^y_{i,j+\frac{1}{2},k} - u^y_{i,j-\frac{1}{2},k}}{h^y_j} + \frac{u^z_{i,j,k+\frac{1}{2}} - u^z_{i,j,k-\frac{1}{2}}}{h^z_k} \tag{3.1}$$

and express the discretization of (2.9d) and (2.9c) on each grid cell as

$$\nabla_{i,j,k} \cdot \widehat{\boldsymbol{J}} + \nabla_{i,j,k} \cdot \boldsymbol{J^s} = 0, \tag{3.2a}$$

$$\psi_{i,j,k} = \mu^{-1}_{i,j,k} \nabla_{i,j,k} \cdot \boldsymbol{A}, \tag{3.2b}$$

where $\mu_{i,j,k} = \mu(x_i, y_j, z_k)$. Note that we are not assuming a uniform grid: each cell may have different widths in each direction. The BCs (2.10) are used at the end points of (3.2).

Next, consider the discretization at cell faces. Following [17], we define the *harmonic average* of $\widehat{\sigma}$ between neighboring cells in the x-direction by

$$\widehat{\sigma}_{i+\frac{1}{2},j,k} = h^x_{i+\frac{1}{2}} \left(\int_{x_i}^{x_{i+1}} \widehat{\sigma}^{-1}(x, y, z) dx \right)^{-1}, \tag{3.3a}$$

where $h^x_{i+\frac{1}{2}} = x_{i+1} - x_i = (h^x_{i+1} + h^x_i)/2$. If $\widehat{\sigma}$ is assumed constant over each cell, this integral evaluates to

$$\widehat{\sigma}_{i+\frac{1}{2},j,k} = h^x_{i+\frac{1}{2}} \left(\frac{h^x_i}{2\widehat{\sigma}_{i,j,k}} + \frac{h^x_{i+1}}{2\widehat{\sigma}_{i+1,j,k}} \right)^{-1}. \tag{3.3b}$$

Then, the resulting approximation for the x-component of (2.9e) is (see [17])

$$\hat{J}^x_{i+\frac{1}{2},j,k} = \widehat{\sigma}_{i+\frac{1}{2},j,k} \left(A^x{}_{i+\frac{1}{2},j,k} + \frac{\phi_{i+1,j,k} - \phi_{i,j,k}}{h^x_{i+\frac{1}{2}}} \right). \tag{3.3c}$$

Next, we discretize (2.9a) as in [46]. Writing the x-component of these equations,

$$(\partial_z H^y - \partial_y H^z) - \partial_x \psi - \hat{J}^x = s^x,$$

where we denote $\boldsymbol{J^s} = (s^x, s^y, s^z)^T$, a discretization centered at the center of the cell's x-face results in

$$\begin{aligned} &\frac{H^y_{i+\frac{1}{2},j,k+\frac{1}{2}} - H^y_{i+\frac{1}{2},j,k-\frac{1}{2}}}{h^z_k} - \frac{H^z_{i+\frac{1}{2},j+\frac{1}{2},k} - H^z_{i+\frac{1}{2},j-\frac{1}{2},k}}{h^y_j} \\ &- \frac{\psi_{i+1,j,k} - \psi_{i,j,k}}{h^x_{i+\frac{1}{2}}} - \hat{J}^x_{i-\frac{1}{2},j,k} = s^x_{i+\frac{1}{2},j,k}. \end{aligned} \tag{3.4}$$

Similar expressions to those in (3.3c) and (3.4) can be derived in the y- and z-directions. The BCs (1.3) are used to close (3.4). Using (3.3c), we can eliminate $\widehat{\boldsymbol{J}}$ from (3.2a) and obtain a discrete equation in which the dominant terms all involve ϕ. The resulting stencil for ϕ has seven points. We also apply the obvious quadrature for the single condition (2.10d).

Finally, we discretize the edge-centered (2.9b). Consider, say, the x-component of (2.9b), written as

$$\partial_z A^y - \partial_y A^z = \imath\omega\mu H^x.$$

Integrating this equation over the surface of a rectangle with corners

$$(x_i, y_j, z_k),\ (x_i, y_{j+1}, z_k),\ (x_i, y_{j+1}, z_{k+1}),\ (x_i, y_j, z_{k+1}),$$

the expression on the left-hand side is integrated using the Gauss **curl** theorem, and μ on the right-hand side is averaged over the rectangular area to obtain a value on the edge. We do not divide through by μ before this integration because we wish to integrate the magnetic flux, which is potentially less smooth around an edge than the magnetic field. This yields

$$\mu_{i,j+\frac{1}{2},k+\frac{1}{2}} = [h^y_{j+\frac{1}{2}} h^z_{k+\frac{1}{2}}]^{-1} \left(\int_{y_j}^{y_{j+1}} \int_{z_k}^{z_{k+1}} \mu(x_i, y, z) dy dz \right). \tag{3.5a}$$

If μ is assumed to be constant over each cell, this integral evaluates to

$$\begin{aligned} &\mu_{i,j+\frac{1}{2},k+\frac{1}{2}} \\ &= [h^y_{j+\frac{1}{2}} h^z_{k+\frac{1}{2}}]^{-1} \left(\frac{\mu_{i,j,k} h^y_j h^z_k + \mu_{i,j+1,k} h^y_{j+1} h^z_k + \mu_{i,j+1,k+1} h^y_{j+1} h^z_{k+1} + \mu_{i,j,k+1} h^y_j h^z_{k+1}}{4} \right). \end{aligned} \tag{3.5b}$$

Then, the resulting approximation for the x-component of (2.9b) is

$$\begin{aligned} H^x_{i,j+\frac{1}{2},k+\frac{1}{2}} = (\imath\omega\mu_{i,j+\frac{1}{2},k+\frac{1}{2}})^{-1} \Bigg(&\frac{A^y_{i,j+\frac{1}{2},k+1} - A^y_{i,j+\frac{1}{2},k}}{h^z_{k+\frac{1}{2}}} \\ &- \frac{A^z_{i,j+1,k+\frac{1}{2}} - A^z_{i,j,k+\frac{1}{2}}}{h^y_{j+\frac{1}{2}}} \Bigg). \end{aligned} \tag{3.5c}$$

Using (3.2b) as well as (3.5c) and similar expressions derived in the y- and z-directions, we substitute for $\boldsymbol{H}$ and ψ in (3.4) and obtain a discrete system of equations for $\boldsymbol{A}$. The resulting stencil for $\boldsymbol{A}$ has 19 points and the same structure as for the discretization of the operator

$$\boldsymbol{\nabla} \times (\mu^{-1} \boldsymbol{\nabla} \times) - \operatorname{grad}(\mu^{-1} \nabla \cdot).$$

The difference between this discretization and a direct discretization of the latter is that μ at the interface is naturally defined as an arithmetic average and not a harmonic average.

The discretization described above can be viewed as a careful extension of Yee's method, suitable for discontinuous coefficients and amenable to fast iterative solution methods. It is centered, conservative, and second order accurate. Note that throughout we have used a consistent, compact discretization of the operators $\nabla \cdot$, $\boldsymbol{\nabla} \times$, and grad. We can denote the corresponding discrete operators by $\nabla_h \cdot$, $\boldsymbol{\nabla}_h \times$, and grad_h and immediately obtain the following identities (cf. [38, 23, 22, 20]):

$$(\boldsymbol{\nabla}_h \times) \operatorname{grad}_h = \boldsymbol{0}, \tag{3.6a}$$

$$(\nabla_h \cdot)(\boldsymbol{\nabla}_h \times) = 0, \tag{3.6b}$$

$$\operatorname{grad}_h(\nabla_h \cdot) - (\boldsymbol{\nabla}_h \times)(\boldsymbol{\nabla}_h \times) = (\nabla_h \cdot) \operatorname{grad}_h. \tag{3.6c}$$

These are, of course, analogues of vector calculus identities which hold for sufficiently differentiable vector functions. The BCs (2.10) are discretized using these discrete operators as well.

Next, note that upon applying $\nabla_h \cdot$ to (3.4) and using (3.6b) and (3.2a), we obtain

$$\nabla_h \cdot \operatorname{grad}_h \psi = 0.$$

Moreover, from (3.4) and (2.10c), the discrete normal derivative of ψ vanishes at the boundaries as well. Setting ψ to 0 arbitrarily at one point,

$$\psi_{1,1,1} = 0$$

then determines that $\psi \equiv 0$ throughout the domain (as a grid function). We obtain another conservation property of our scheme, namely, a discrete divergence-free $\boldsymbol{A}$:

$$\nabla_h \cdot \boldsymbol{A} = 0. \tag{3.7}$$

Recall the stabilizing term added in (2.7). For the exact solution this term obviously vanishes. Now, (3.7) assures us that the corresponding discretized term vanishes as well (justifying use of the term "stabilization," rather than "penalty"). This is not the case for the nodal finite element method which was considered in [24, 11]. For an approximate solution that does not satisfy (3.7), the stabilization term may grow in size when μ varies over a few orders of magnitude, or else $\nabla_h \cdot \boldsymbol{A}$ grows undesirably in an attempt to keep $\mu^{-1}\nabla_h \cdot \boldsymbol{A}$ approximately constant across an interface [24].

The particular way we average across discontinuities, namely, arithmetic averaging of μ at cell edges and harmonic averaging of $\widehat{\sigma}$ at cell faces, can be important. The averaging can be viewed as a careful approximation of the constitutive relationship for discontinuous coefficients (see, e.g., [36, 38, 1] and many others). To show that, we look first at the relation

$$\hat{J}^x = \widehat{\sigma} E^x$$

across a face whose normal direction is x. This flux flows in series, and therefore an approximate $\widehat{\sigma}$ that represents the bulk property of the flow through the volume is given by the harmonic average (corresponding to an arithmetic average of the resistivities). Next, we look at the relation

$$B^x = \mu H^x,$$

where μ is an edge variable and B^x is the flux through four cells which share that edge. Here the flow is in parallel, which implies that we need to approximate μ on the edge by an arithmetic average.

Note also that if we use the more common implementation of Yee's method (i.e., $\boldsymbol{H}$ on the cell's faces and $\boldsymbol{E}$ on the cell's edges), then the roles of $\widehat{\sigma}$ and μ interchange and we need to average μ harmonically and $\widehat{\sigma}$ arithmetically (see also [1]).

4. Solution of the discrete system. After the elimination of $\boldsymbol{H}$, ψ, and $\widehat{\boldsymbol{J}}$ from the discrete equations, we obtain a large, sparse system for the grid functions corresponding to $\boldsymbol{A}$ and ϕ:

$$K\ u\ = \begin{pmatrix} H_1 - \imath\omega S & \imath\omega S D^T \\ -DS & H_2 \end{pmatrix} \begin{pmatrix} \boldsymbol{A} \\ \phi \end{pmatrix} = \begin{pmatrix} b_A \\ b_\phi \end{pmatrix} = b. \tag{4.1}$$

Here

$$H_1 = C^T M C + D^T M_c D$$

is the result of the discretization of the operator $\boldsymbol{\nabla} \times \mu^{-1} \boldsymbol{\nabla} \times \; - \operatorname{grad} \mu^{-1} \nabla \cdot$, C corresponds to the discretization of the operator $\boldsymbol{\nabla} \times$, D likewise corresponds to the discretization of the operator $\nabla \cdot$, the diagonal matrix S results from the discretization of the operator $\widehat{\sigma}(\cdot)$, M and M_c similarly arise from the operator $\mu^{-1}(\cdot)$ at cell edges and at cell centers, respectively, and $H_2 = DSD^T$ represents the discretization of $\nabla \cdot (\widehat{\sigma} \operatorname{grad}(\cdot))$. In regions of constant μ the matrix H_1 simplifies into a discretization of the vector Laplacian. The blocks are weakly coupled through interfaces in μ. A typical sparsity structure of H_1 for variable μ, as contrasted with constant μ, is displayed in Figure 3. The structure of the obtained system is similar to that in [17],

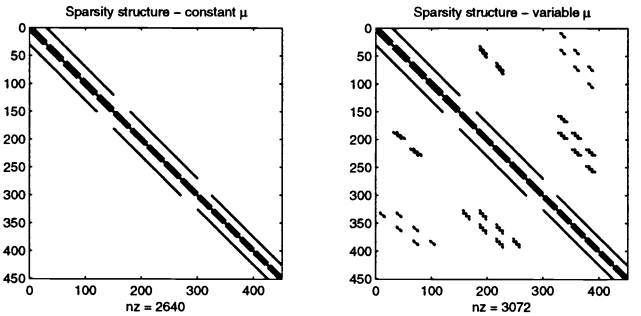


FIG. 3. *Sparsity structure of the matrix H_1 corresponding to variable μ (right) and to constant μ (left).*

although the main block diagonal is somewhat less pleasant.

Note that as long as the frequency is low enough that the diffusion number satisfies

$$d = \omega \mu \sigma h \ll 1$$

(where h is the maximum grid spacing), the matrix is dominated by the diagonal blocks. This allows us to develop a block preconditioner good for low frequencies ω based on the truncated ILU (ILU(t)) decomposition of the major blocks [35]. Thus, we approximate K by the block diagonal matrix

$$\hat{K} = \begin{pmatrix} H_1 & 0 \\ 0 & H_2 \end{pmatrix} \tag{4.2}$$

and then use ILU(t) to obtain a sparse approximate decomposition of the matrix $\hat{K}$. This decomposition is used as a preconditioner for the Krylov solver. Note that, although K is complex, the approximation $\hat{K}$ is real, and therefore we need to apply the ILU decomposition only to two real matrices, which saves memory.

The block approximation (4.2) makes sense for our discretization but not for the direct staggered grid discretization of (1.1). Thus, the reformulation and subsequent discretization of Maxwell's equations allow us to easily obtain a good block preconditioner in a modular manner, while the discretization of (1.1) does not.

5. Numerical examples. In this section we present numerical results using our method with standard Krylov-type methods and preconditioners for solving the

resulting discrete systems. We vary the type of source used, the size of jumps in the coefficients σ and μ, the preconditioner, and the grid size.

We have verified second order convergence of the method and compared our results to those from another code in [17, 18]. Here our goal is to demonstrate the efficacy of our proposed method for variable μ.

In the tables below, "iterations" denotes the number of BICGSTAB iterations needed for achieving a relative accuracy of 10^{-7}, "operations" denotes the number of giga-flops required, the SSOR parameter (when used) equals 1, and the ILU threshold (when used) equals 10^{-2}. The latter threshold is such that in our current MATLAB implementation iterations involving these two preconditioners cost roughly the same.

5.1. Example set 1. We derive the following set of experiments. Let the air's permeability be $\mu_0 = 4\pi \cdot 10^{-7}$ H/m and its permittivity be $\epsilon_0 = 8.85 \cdot 10^{-12}$ F/m. We assume a cube of constant conductivity σ_c and permeability μ_c embedded in an otherwise homogeneous earth with conductivity $\sigma_e = 10^{-3}$ S/m and permeability $\mu_e = \mu_0$; see Figure 4. In a typical geophysical application, the conductivity may range over four orders of magnitude and more, whereas the permeability rarely changes over more than one order of magnitude. Therefore, we experiment with conductivity σ_c ranging from 10^{-2} S/m to 10^3 S/m and permeability μ_c ranging from μ_0 to $1000\mu_0$.

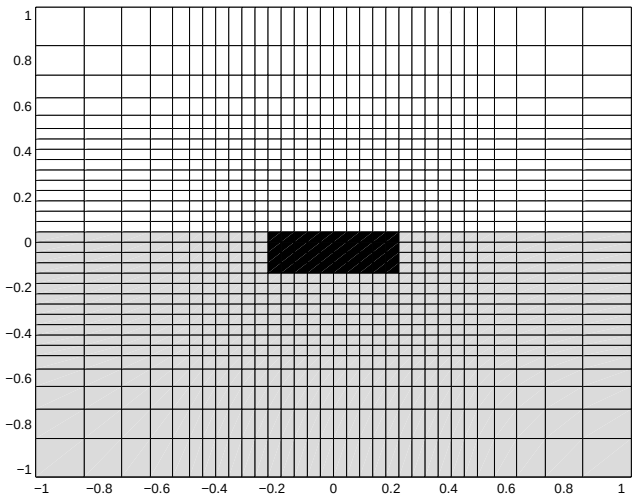


FIG. 4. *The setting of our first numerical experiment. A cube of conductivity σ_c and permeability μ_c is embedded inside a homogeneous earth with conductivity σ_e and permeability μ_e. Also, in the air $\widehat{\sigma} = -\imath\omega\epsilon_0$ and $\mu = \mu_0$.*

We experiment with two different sources: (i) a magnetic source (a plane wave), and (ii) an electric dipole source in the x direction centered at $(0, 0, 0)$. The fact that the first source is magnetic implies that it is divergence-free. This source lies entirely in the active space of the **curl** operator. In contrast, the electric source is not divergence-free.

Both sources are assumed to oscillate with different frequencies ranging from 1 to 10^6 Hz. The solution is obtained, unless otherwise noted, on a nonuniform tensor grid (see Figure 4) consisting of 32^3 cells. There are 95232 (complex) $\boldsymbol{E}$ unknowns corresponding to this grid and 128000 (complex) $\boldsymbol{A}, \phi$ unknowns. We then solve the system using the method described in the previous section.

5.1.1. Example 1a. In order to be able to compare the resulting linear algebra solver with that corresponding to Yee's method, we discretize the system

$$\boldsymbol{\nabla} \times [\mu^{-1} \boldsymbol{\nabla} \times (\widehat{\boldsymbol{J}}/\widehat{\sigma})] - \imath\omega \widehat{\boldsymbol{J}} = \imath\omega \boldsymbol{J}^s \tag{5.1}$$

using the staggered grid depicted in Figure 2, i.e., where $\widehat{\boldsymbol{J}}$ is on the cell's faces, which is similar to the discretization in [28]. This yields the discrete system

$$(C^T M C - \imath\omega S)e = \hat{b}$$

for the unknown vector e corresponding to grid values of $\widehat{\boldsymbol{J}}/\widehat{\sigma}$, where the matrices C, M, and S are defined in section 4 and $\hat{b}$ depends on the source. In order to solve this system as well as ours, we use BICGSTAB and an SSOR preconditioner. The comparison between the methods for the case $\sigma_c = 10^3\sigma_e$, $\mu_c = 100\mu_0$, and different frequencies is summarized in Table 2.

TABLE 2

Iteration counts and computational effort for our method $(\boldsymbol{A}, \phi)$ and the traditional implementation of Yee's method (applied to $\boldsymbol{E}$, or $\widehat{\boldsymbol{J}}$) for example set 1 using both an electric source and a magnetic source.

	Electric source				Magnetic source			
ω	Iterations		Operations		Iterations		Operations	
	$\boldsymbol{A}, \phi$	$\boldsymbol{E}$	$\boldsymbol{A}, \phi$	$\boldsymbol{E}$	$\boldsymbol{A}, \phi$	$\boldsymbol{E}$	$\boldsymbol{A}, \phi$	$\boldsymbol{E}$
10^0	46	9856	3.8	640	77	86	5.0	5.6
10^2	66	10323	4.3	670	82	91	5.3	5.9
10^4	77	10878	5.0	706	89	103	5.8	6.7
10^6	97	11212	6.3	728	99	113	6.4	7.3

Table 2 shows that our method converges in a moderate number of iterations for both sources, despite the presence of significant jumps in μ and σ. On the other hand, the more traditional discretization performs poorly for the electric source and comparably to our method for the magnetic source. Slow convergence of the direct staggered discretization of Maxwell's equations in the case of an electric source will happen also when $\boldsymbol{E}$ is defined on the grid's edges.

These results clearly show an advantage of our formulation over the original Yee formulation, even for a simple preconditioning, especially for electric sources and in low frequencies. In such circumstances, the discretized first term on the left-hand side of (5.1) strongly dominates the other term, and the residual in the course of the BICGSTAB iteration has a nontrivial component in the null space of that operator; hence its reduction is very slow. The magnetic source, on the other hand, yields a special situation where, so long as the discrete divergence and the roundoff error are relatively small, the residual component in the null space of the leading term operator is also small; hence the number of iterations using the traditional method is not much larger than when using our method.

5.1.2. Example 1b. Next, we test the effect of discontinuities on our method. We use the electric source and record the number of BICGSTAB iterations needed for our method to converge for various values of $\tilde{\mu} = \mu_c/\mu_e$ and $\tilde{\sigma} = \sigma_c/\sigma_e$, when using block ILU(t) preconditioning as described in the previous section. The results are summarized in Table 3.

Note that large jump discontinuities in σ do not significantly affect the rate of convergence of the iterative linear system solver for our method, but large jump discontinuities in μ have a decisive effect. Results in a similar spirit were reported in [19]

TABLE 3
Iteration counts for different frequencies, conductivities, and permeabilities in example set 1. *The conductivity/permeability structure is a cube in a half-space.*

		$\tilde{\sigma}$					
ω	$\tilde{\mu}$	10^1	10^2	10^3	10^4	10^5	10^6
0	10^1	35	36	36	31	27	28
	10^2	67	55	54	52	47	51
	10^3	143	143	140	142	140	148
1	10^1	41	40	35	34	35	32
	10^2	63	51	50	53	63	51
	10^3	165	158	169	160	161	165
10	10^1	42	39	39	39	35	36
	10^2	75	59	60	61	61	60
	10^3	201	166	182	178	185	187
10^2	10^1	46	43	44	40	47	39
	10^2	71	64	66	64	63	63
	10^3	201	190	199	184	220	177
10^3	10^1	44	44	39	38	38	41
	10^2	75	69	59	61	67	69
	10^3	219	224	202	240	216	204
10^4	10^1	47	44	41	44	44	44
	10^2	85	72	80	80	65	68
	10^3	238	246	201	259	293	252
10^5	10^1	57	50	44	44	47	48
	10^2	91	79	78	82	75	81
	10^3	321	232	245	244	261	242
10^6	10^1	62	56	50	52	57	53
	10^2	105	95	89	89	91	80
	10^3	365	290	301	270	286	287

regarding the effect of discontinuities in μ on a specialized multigrid method for an edge-element discretization. However, even for large discontinuities in μ the number of iterations reported in Table 3 remains relatively small compared with similar experiments reported in [34] (for constant μ) and [10]. We attribute the increase in the number of iterations as the jump in μ increases in size to a corresponding degradation in the condition number of K in (4.1). This degradation, however, does not depend strongly on grid size, as we verify next.

5.1.3. Example 1c. In the next experiment, we use the cube model with the electric source to evaluate the influence of the grid on the number of iterations. We fix $\omega = 10^2$ and test our method on a case with a modest coefficient jump $\tilde{\mu} = 10$ and on a case with a large jump $\tilde{\mu} = 10^3$. A set of uniform grids in the interval $[-1, 1]^3$ is considered. For each grid we record the resulting number of iterations using both the SSOR and the block ILU preconditioners. The results of this experiment are gathered in Table 4.

TABLE 4
Iteration counts for different grids, for two sets of problem coefficients and using two preconditioners.

	$\tilde{\mu} = \tilde{\sigma} = 10$		$\tilde{\mu} = \tilde{\sigma} = 1000$	
Grid size	ILU	SSOR	ILU	SSOR
8^3	6	20	58	268
16^3	10	32	108	453
32^3	23	52	213	789
64^3	31	76	396	1302

We observe that the number of iterations increases as the number of unknowns increases. The increase appears to be roughly proportional to the number of unknowns to the power 1/3. The growth in number of iterations as a function of grid size is also roughly similar for both preconditioners, although the block ILU requires fewer iterations (about 1/4 for $\tilde{\mu} = 1000$) for each grid. However, ILU requires more memory than SSOR, which may prohibit its use for very large problems. The increase rate is also similar, as expected, for both values of $\tilde{\mu}$. The rate of increase in number of iterations as $\tilde{\mu}$ increases appears essentially independent of the grid size. Practically, however, this increase is substantial and may be hard to cope with for (perhaps unrealistically) large values of $\tilde{\mu}$ using the present techniques.

5.2. Example 2. In our next experiment we consider a more complicated earth structure. We employ a random model, which is used by practitioners in order to simulate stochastic earth models [25]. Two distinct value sets (σ_1, μ_1), (σ_2, μ_2) and a probability P are assumed for the conductivities and permeabilities: for each cell, the probability of having values $\sigma_1,\ \mu_1$ is P, and the probability of having values $\sigma_2,\ \mu_2$ is $1 - P$. This can be a particularly difficult model to work with, as the conductivity and permeability may jump anywhere from one cell to the next, not necessarily just across a well-defined manifold. A cross-section of such a model is plotted in Figure 5. We then carry out experiments as before for frequencies ranging from 0 to 10^6 Hz.

FIG. 5. *The setting of example set* 2.

We use the random model with different conductivities, permeabilities, and frequencies and the electric source, with $P = 0.5$ on the domain $[-1, 1]^3$ (in km). We employ a uniform grid of size 44^3 and use both the block ILU and the SSOR preconditioners. The results of this experiment in terms of iteration counts are summarized in Table 5. The results show that our solution method is effective even for highly varying conductivity. As before, the method deteriorates when very large variations in μ are present.

We can also see from Table 5 that the block ILU preconditioner works very well for low frequencies, but it is not very effective for high frequencies. It is easy to check that in all cases where the block ILU preconditioner fails to achieve convergence (denoted "nc," meaning failure to achieve a residual of 10^{-7} in 800 iterations) the maximum grid spacing h satisfies $\omega\mu\sigma\ h \gg 1$. In such a case the discretization of the leading terms of the differential operator no longer yields the dominant blocks in the matrix equations (4.1), and therefore our block ILU preconditioner fails. Thus, for high frequency and high conductivity we require more grid points in order for this preconditioner to be effective. This is also consistent with the physics, as the skin

TABLE 5
Iteration counts for different frequencies, conductivities, and permeabilities for example set 2.

		σ_2/σ_1					
		10^2		10^4		10^6	
ω	μ_2/μ_1	ILU	SSOR	ILU	SSOR	ILU	SSOR
0	10^1	12	35	13	32	13	48
	10^2	27	85	29	69	31	83
	10^3	143	215	142	272	148	229
1	10^1	14	34	13	35	15	47
	10^2	28	80	34	81	26	81
	10^3	158	270	160	306	165	269
10	10^1	18	47	17	39	16	44
	10^2	33	90	34	94	35	105
	10^3	201	329	182	303	185	354
10^2	10^1	17	47	19	45	19	59
	10^2	37	104	35	101	34	119
	10^3	190	393	184	380	177	365
10^3	10^1	21	44	22	45	25	60
	10^2	42	141	47	113	57	136
	10^3	224	434	240	453	204	434
10^4	10^1	24	46	23	51	46	65
	10^2	60	140	41	149	nc	151
	10^3	246	503	259	508	252	487
10^5	10^1	25	58	27	54	nc	76
	10^2	66	151	72	149	nc	157
	10^3	232	581	244	588	242	591
10^6	10^1	26	62	55	62	nc	77
	10^2	56	180	nc	170	nc	157
	10^3	290	641	270	643	241	655

depth [45] decreases and the attenuated wave can be simulated with fidelity only on a finer grid.

6. Summary and further remarks. In this paper we have developed a fast finite volume algorithm for the solution of Maxwell's equations with discontinuous conductivity and permeability. The major components of our approach are as follows.

- Reformulation of Maxwell's equations. The Helmholtz decomposition is applied to $\boldsymbol{E}$; then a stabilizing term is added, resulting in a strongly elliptic system; the system is written in first order form to allow flexibility in the ensuing discretization; and finally, the divergence-free Coulomb gauge condition is eliminated using differentiation and substitution, which yields a weakly coupled PDE system enabling an efficient preconditioner for the large, sparse algebraic system which results from the discretization.
- Discretization using staggered grids, respecting continuity conditions, and carefully averaging material properties across discontinuities. For this discretization, the stabilizing term vanishes at the exact solution of the discrete equations, which is important for cases with large contrasts in μ.
- Solution of the resulting linear system using fast preconditioned Krylov methods.

The resulting algorithm was tested on a variety of problems. We have shown dramatic improvement over the more standard algorithm when the source is electric. Good performance was obtained even when the coefficients μ and σ were allowed to vary rapidly on the grid scale—a case which should prove challenging for multigrid methods.

The project that has motivated us is electromagnetic inverse problems in geophysical prospecting [42]. Solving the forward problem, i.e., Maxwell's equations as in the present article, is a major bottleneck for the data inversion methods—indeed, it may have to be carried out dozens, if not hundreds, of times for each data inversion. Thus, extremely fast solvers of the problem discussed in our paper are needed. Based on the algorithm described here an implementation has been carried out which solves realistic instances of this forward problem in less than two minutes on a single-processor PC, enabling derivation of realistic algorithms at low cost for the inverse problem.

Acknowledgments. We wish to thank Drs. Doug Oldenburg and Dave Moulton for fruitful discussions and an anonymous referee for valuable comments on the first two sections of our exposition.

REFERENCES

[1] D. Alumbaugh, G. Newman, L. Prevost, and J. Shadid, *Three dimensional wideband electromagnetic modeling on a massively parallel computer*, Radio Science, 31 (1996), pp. 1–23.

[2] D. Aruliah, U. Ascher, E. Haber, and D. Oldenburg, *A method for the forward modelling of 3D electromagnetic quasi-static problems*, Math. Models Methods Appl Sci., to appear.

[3] R. Beck, P. Deuflhard, R. Hiptmair, R. Hoppe, and B. Wohlmuth, *Adaptive multilevel methods for edge element discretizations of Maxwell's equations*, Surveys Math. Indust., 8 (1999), pp. 271–312.

[4] O. Bíró and K. Preis, *On the use of the magnetic vector potential in the finite element analysis of three-dimensional eddy currents*, IEEE Trans. Magnetics, 25 (1989), pp. 3145–3159.

[5] A. Bossavit, *Whitney forms: A class of finite elements for three-dimensional computations in electromagnetism*, IEEE Proc. A, 135 (1988), pp. 493–500.

[6] A. Bossavit and L. Kettunen, *Yee-like schemes on staggered cellular grids: A synthesis between fit and fem approaches*, COMPUMAG, (1999).

[7] A. Brandt and N. Dinar, *Multigrid solution to elliptic flow problems*, in Numerical Methods for PDE's, Academic Press, New York, 1979, pp. 53–147.

[8] F. Brezzi and M. Fortin, *Mixed and Hybrid Finite Element Methods*, Springer-Verlag, New York, 1991.

[9] G. Carey and T. Oden, *Finite Elements, a Second Course*, Prentice-Hall, Englewood Cliffs, NJ, 1983.

[10] M. Clemens and T. Weiland, *Numerical algorithms for the FDITD and FDFD simulation of slowly varying electromagnetic fields*, Int. J. Numer. Modelling, to appear.

[11] N. A. Demerdash and R. Wang, *Theoretical and numerical difficulties in 3D vector potential methods in finite element magnetostatic computations*, IEEE Trans. Magnetics, MAG-26 (1990), pp. 1656–1658.

[12] J. E. Dendy, *Black box multigrid*, J. Comput. Phys., 48 (1982), pp. 366–386.

[13] M. Everett and A. Schultz, *Geomagnetic induction in a heterogenous sphere: Azimuthally symmetric test computations and the response of an undulating 660-km discontinuity*, J. Geophys. Res., 101 (1996), pp. 2765–2783.

[14] V. Girault and P. Raviart, *Finite Element Methods for Navier-Stokes Equations*, Springer-Verlag, Berlin, Heidelberg, 1986.

[15] P. M. Gresho and R. L. Sani, *On pressure boundary conditions for the incompressible Navier-Stokes equations*, Internat. J. Numer. Methods Fluids, 7 (1987), pp. 1111–1145.

[16] E. Haber, *A mixed finite element method for the solution of the magnetostatic problem in 3D*, Comput. Geosci., to appear.

[17] E. Haber, U. Ascher, D. Aruliah, and D. Oldenburg, *Fast simulation of 3D electromagnetic using potentials*, J. Comput. Phys., 163 (2000), pp. 150–171.

[18] E. Haber, U. Ascher, and D. Oldenburg, *Solution of the 3D electromagnetic problem*, in Proceedings of the Society of Exploration Geophysics, Calgary, 2000, pp. 304–307.

[19] R. Hiptmair, *Multigrid method for Maxwell's equations*, SIAM J. Numer. Anal., 36 (1998), pp. 204–225.

[20] J. Hyman and M. Shashkov, *The adjoint operators for natural discretizations for the divergence, gradient and curl on logically rectangular grids*, IMACS J. Applied Numerical Math., 25 (1997), pp. 413–442.

[21] J. Hyman and M. Shashkov, *Mimetic discretizations for Maxwell's equations*, J. Comput. Phys., 151 (1999), pp. 881–909.
[22] J. M. Hyman and M. Shashkov, *Natural discretizations for the divergence, gradient and curl on logically rectangular grids*, Comput. Math. Appl., 33 (1997), pp. 81–104.
[23] J. M. Hyman and M. Shashkov, *The orthogonal decomposition theorems for mimetic finite difference methods*, SIAM J. Numer. Anal., 36 (1999), pp. 788–818.
[24] J. Jin, *The Finite Element Method in Electromagnetics*, John Wiley and Sons, New York, 1993.
[25] R. Knight and P. Tercier, *Electrical conduction in sandstones: Examples of two dimensional and three dimensional percolation*, Transactions, American Geophysical Union, 75 (1992), pp. 118–121.
[26] D. LaBrecque, *A scalar-vector potential solution for 3D EM finite-difference modeling*, in Proceedings of the International Symposium on Three-Dimensional Electromagnetics, M. Oristaglio and B. Spies, eds., Schlumberger-Doll Research, Ridgefield, CT, 1995, pp. 143–149.
[27] M. S. Longer, *Theoretical Concepts in Physics*, Cambridge University Press, Cambridge, UK, 1991.
[28] R. Mackie, T. Madden, and P. Wannamaker, *Three dimensional magnetotelluric modeling using difference equations—theory and comparison to integral equation solutions*, Geophysics, 58 (1993), pp. 215–226.
[29] C. Mattiussi, *An analysis of finite volume, finite element, and finite difference methods using some concepts from algebraic topology*, J. Comput. Phys., 133 (1997), pp. 289–309.
[30] P. Monk and E. Suli, *A convergence analysis of Yee's scheme on nonuniform grids*, SIAM J. Numer. Anal., 31 (1994), pp. 393–412.
[31] J. D. Moulton, *Numerical Implementation of Nodal Methods*, Ph.D. thesis, Institute of Applied Mathematics, University of British Columbia, Vancouver, Canada, 1996.
[32] T. Nakata, N. Takahashi, and K. Fujiwara, *Solutions of 3D eddy current problems by finite elements*, in Finite Element Electromagnetics and Design, Elsevier, New York, 1995, pp. 36–72.
[33] G. Newman, *Three dimensional magnetotelluric modeling and inversion*, in Proceedings of the Second Symposium on 3D Electromagnetics, Salt Lake City, UT, 1999, pp. 157–161.
[34] G. Newman and D. Alumbaugh, *Frequency-domain modelling of airborne electromagnetic responses using staggered finite differences*, Geophys. Prospecting, 43 (1995), pp. 1021–1042.
[35] Y. Saad, *Iterartive Methods for Sparse Linear Systems*, PWS Publishing, New York, 1996.
[36] S. Selberherr, *Analysis and Simulation of Semiconductor Devices*, Springer-Verlag, New York, 1984.
[37] D. Sidilkover and U. Ascher, *A multigrid solver for the steady state Navier-Stokes equations using the pressure-Poisson formulation*, Comput. Appl. Math., 14 (1995), pp. 21–35.
[38] T. J. Smith, *Conservative modeling of 3D electromagnetic fields; part* I: *Properties and error analysis*, Geophysics, 61 (1996), pp. 1308–1318.
[39] T. J. Smith, *Conservative modeling of 3D electromagnetic fields; part* II: *Biconjugate gradient solution and an accelerator*, Geophysics, 61 (1996), pp. 1319–1324.
[40] A. Taflove, *Computational Electrodynamics: The Finite-Difference Time-Domain Method*, Artech House, Norwood, MA, 1995.
[41] Y. Tai, *Dyadic Green Functions in Electromagnetics*, Elsevier, New York, 1991.
[42] S. Ward and G. Hohmann, *Electromagnetic theory for geophysical applications*, Electromagnetic Methods in Applied Geophysics, 1 (1988), pp. 131–311.
[43] C. Weaver, *Mathematical Methods for Geo-Electromagnetic Induction*, John Wiley and Sons, New York, 1994.
[44] T. Weiland, *Time domain electromagnetic field computation with finite difference methods*, Int. J. Numer. Modelling, 9 (1996), pp. 295–319.
[45] K. P. Whittall and D. W. Oldenburg, *Inversion of Magnetotelluric Data for a One Dimensional Conductivity*, vol. 5, Society of Exploration Geophysics monograph, 1992.
[46] K. Yee, *Numerical solution of initial boundary value problems involving Maxwell's equations in isotropic media*, IEEE Trans. Antennas and Propagation, 14 (1966), pp. 302–307.

SIAM J. SCI. COMPUT.
Vol. 22, No. 6, pp. 1962–1973
© 2001 Society for Industrial and Applied Mathematics

EXACT PREDICTION OF QR FILL-IN BY ROW-MERGE TREES*

SUELY OLIVEIRA†

Abstract. Row-merge trees for forming the QR factorization of a sparse matrix A are closely related to elimination trees for the Cholesky factorization of A^TA. Row-merge trees predict the exact fill-in (assuming no numerical cancellation) provided A satisfies the strong Hall property, but overestimate the fill-in in general. However, here a fast and simple postprocessing step for row-merge trees is presented that predicts the exact fill-in for sparse QR factorization using Householder reflectors for general matrices.

Key words. row-merge trees, elimination trees, QR factorization

AMS subject classifications. 65F50, 65F25

PII. S1064827599333965

1. Introduction.

Matrix factorizations of sparse matrices typically result in creating further nonzero entries, or *fill-in*. If this fill-in can be accurately predicted in advance, then the factorization can be performed in less time, as the additional memory needed can be allocated once in advance. Notice that fill-in can be reduced with some matrix reordering algorithms. After that, the algorithms presented in this paper can be used. The algorithms discussed here do not affect the amount of fill-in but predict the fill-in before the factorization [17]. Performance comparisons between our algorithms and the Dulmage–Mendelsohn decomposition [20, 19] will be presented at the end of the paper, but notice that even though our new algorithms perform well in these comparisons, the Dulmage–Mendelsohn decomposition can be used to predict fill-in and to achieve other goals as well (such as re-ordering rows and columns to give a block-triangular form); see, for example, [15, 20, 19]. Also notice that the algorithms in [19] can compute the fill-in in time $O(n\,nz(A))$ if no column permutation is allowed, where $nz(A)$ is the number of nonzeros in A. The elimination tree algorithms presented in this paper appear to be significantly faster than this. Perhaps what is most significant about the algorithms presented here is that they are developed by modifying the construction of elimination trees or row-merge trees and may give new insight into the graphical prediction of fill-in. Furthermore, elimination trees [13] can be used to compute symbolic factorizations, to direct the factorization of matrices in multifrontal methods [13, 14], and also to assist in parallelizing sparse systems solvers. The QR fill-in problem considered in this paper is similar to the sparse Gaussian elimination with partial pivoting considered in [7] in the sense that the fill-in for QR factorization is the union of the fill-in for Gaussian elimination with partial pivoting, over all pivot sequences.

The elimination tree of a symmetric positive definite matrix B is readily defined in terms of its Cholesky factors: if $B = R^TR$ with R upper triangular, then $ET(B)$ is the forest with nodes $\{1, 2, \ldots, n\}$, where node p is the parent of node j if $p = \min\{\, i \mid i > j,\ r_{ji} \neq 0 \,\}$. That is, p is the first row/column to be updated at the jth stage of the Cholesky factorization of B. The elimination tree for Cholesky factorizations can

*Received by editors October 29, 1999; accepted for publication (in revised form) September 26, 2000; published electronically February 21, 2001. This research was partially supported by NSF grant DMS-9720607.

http://www.siam.org/journals/sisc/22-6/33396.html

†Department of Computer Science and Department of Mathematics, The University of Iowa, Iowa City, IA 52242-1419 (oliveira@cs.uiowa.edu).

be efficiently computed from the original matrix B; indeed, it can be computed in $O(nz(B)\,\alpha(nz(B)))$ time, where $nz(B)$ is the number of nonzeros of the $n \times n$ matrix B, and $\alpha(\cdot)$ is the inverse Ackermann function of Tarjan [22]. Whenever $j \geq i$, if $b_{jk} \neq 0$ and i is an ancestor of k in the elimination tree, then $r_{ij} \neq 0$ [13]. Furthermore, there are no other nonzeros in R. Note that each node is regarded as an ancestor of itself. Unless there is numerical cancellation, $ET(B)$ does not depend on the numerical values in B. The elimination tree is sometimes called the *elimination forest* since it is not necessarily connected.

The graph of an $n \times n$ symmetric matrix B (denoted $G(B)$) is the graph on n nodes $\{1, 2, \ldots, n\}$ with an edge connecting nodes i and j (denoted $i \sim j$) if and only if $b_{ij} \neq 0$ and $i \neq j$; nodes are labeled by the row or column number. Unsymmetric and rectangular matrices are represented by bipartite graphs. For a symmetric positive definite matrix B, unless there is numerical cancellation in the Cholesky factorization $B = R^T R$, the symbolic factorization is the graph $G(R + R^T)$ and is denoted $G^+(B)$. Forming $G^+(B)$ can be defined in graph theoretic terms: $i \sim j$ in $G^+(B)$ if and only if there is a path from i to j in $G(B)$ whose nodes are all less than $\min(i, j)$.

For the QR factorization of an $m \times n$ matrix A it is more appropriate to consider the *bipartite graph* of A ($BG(A)$) whose nodes consist of the rows $\mathcal{R} = \{ r_1, r_2, \ldots, r_m \}$ and columns $\mathcal{C} = \{ c_1, c_2, \ldots, c_n \}$ (which are disjoint sets), and $r_i \sim c_j$ if and only if $a_{ij} \neq 0$. The symbolic QR factorization of A can be computed similarly to the way the Cholesky factorization of a symmetric positive definite matrix is computed and is based on the *row-merge tree of* A, which we define in section 1.2. We consider QR factorizations obtained by Householder orthogonalization. Each jth step of a Householder orthogonalization of A is achieved by multiplying matrix $A^{(j)}$ by a Householder reflector $H_j = I - \beta_j w_j w_j^T$ [10]. The matrices H_j are constructed to zero all the column entries below the jth diagonal of matrix $A^{(j)}$. We then set $A^{(j+1)} = H_j A^{(j)}$. At the nth step we are finally left with an upper triangular matrix $R = A^{(n)}$. The $Q = H_1 H_2 \ldots H_n$ factor is usually represented by the matrix W of Householder vectors (column j of W is the w_j vector used to construct the Householder reflector H_j) and the vector β. Thus, the main problem is to determine the sparsity structures of W and R. In this paper, the fill-in for the QR factorization will be determined from the graphs and related row-merge trees mentioned above, for any full-rank matrix.

1.1. Hall matrices and graphs. A bipartite graph $G = (V_1, V_2, E)$ is a *Hall graph* if for every set $N \subset V_2$, there are no fewer nodes (in V_1) adjacent to N than are in N. For example, if $N = V_2$, then the set of nodes adjacent to N must be no smaller than N; thus $|V_1| \geq |V_2|$. The bipartite graph G is *strong Hall* if for all $N \subset V_2$ such that $N \neq V_2$, the set of nodes adjacent to N has size at least $|N| + 1$.

A matrix A is said to be a Hall matrix (resp., a strong Hall matrix) if $BG(A)$ is a Hall graph (resp., strong Hall graph). Consider V_1 to be $\mathcal{R}$ and V_2 to be $\mathcal{C}$. If A is a full-rank matrix with $m \geq n$, then it is a Hall matrix [8, Cor. 2.4]. In addition, if A is a strong Hall matrix, then the fill pattern of the Cholesky factor R of the symbolic product $A^T A$ is equal to the fill pattern for the symbolic R factor in the QR factorization of A [1]. Problems arise for matrices A that are not strong Hall matrices. Consider, for example,

$$A = \begin{bmatrix} \times & \times & \times & \cdots & \times \\ & \times & & & \\ & & \times & & \\ & & & \ddots & \\ & & & & \times \end{bmatrix}, \tag{1.1}$$

for $i = 1, \ldots, m$: $S(u_i) \leftarrow \{ c_j \mid a_{ij} \neq 0 \} \cup \{c_i\}$
for $j = 1, \ldots, n$: $S(p_j) \leftarrow S(u_j)$
for $j = 1, \ldots, n$: $S(p_j) \leftarrow \bigcup_{x \in \mathcal{U} \cup \mathcal{P} : c_j \in S(x)} S(x) \bigcap \{c_{j+1}, \ldots, c_n\}$

FIG. 1.1. *Symbolic QR factorization.*

which is a matrix with nonzeros in the first row and diagonal only. This matrix is a Hall matrix but not a strong Hall matrix. The QR factorization of A is trivial. However, forming $A^T A$ symbolically gives a dense matrix (due to the dense row in A). Thus from the nonzero structure of $A^T A$, it appears that the R in the QR factorization is dense, although it clearly is not. On the other hand,

$$A' = \begin{bmatrix} \times & \times & \times & \ldots & \times \\ \times & \times & \times & \ldots & \times \\ & \times & & & \\ & & \times & & \\ & & & \ddots & \\ & & & & \times \end{bmatrix} \tag{1.2}$$

is a strong Hall matrix, and the R in the QR factorization *is* dense. As we will show in this paper, distinguishing these cases can be done by postprocessing the row-merge tree. A paper which finds the fill-in structure of Q and R is [11]. Their methods are based on matchings and Hall matrices properties, which differ from the methods used here. In this paper, the fill-in for the QR factorization for nonstrong Hall matrices will be determined from the graphs and related row-merge trees.

1.2. QR factorizations and row-merge trees. Hereafter, we assume that A is an $m \times n$ Hall matrix with $m \geq n$, and the rows and columns have been matched so that $a_{ii} \neq 0$ for all i. As mentioned before, if A is not a Hall matrix, it cannot be full rank.

The appropriate generalization of elimination trees to QR factorizations is the *row-merge tree* [12, 18] of a nonsymmetric matrix A. This can be derived from the crude symbolic QR factorization of A. Let A be $m \times n$, with $m \geq n$. Let $\mathcal{U} = \{u_1, u_2, \ldots, u_m\}$ denote the *unprocessed* rows of A, and let $\mathcal{P} = \{p_1, p_2, \ldots, p_n\}$ denote the processed rows of A (existing rows of R, after the QR factorization has been performed). Define $index(u_l) = l$ and $index(p_k) = k$. Define $S(x)$ to be the set of nonzero entries in x, and specifically consider $S(u_i)$ to be the set of nonzero entries columns in the ith row *before* the factorization algorithm is started, plus the diagonal column (vertex number for the diagonal column). The set $S(p_j)$ is built through the last loop in Figure 1.1 pseudocode. It gives an upper bound on the set of nonzero entries in row j ($j \leq m$) after the jth step of Householder QR factorization, less the entries before the $(j+1)$th column. We define the set $\mathcal{R}_j \subset \mathcal{R}$ to be the set of rows x which have a nonzero in the jth column ($c_j \in S(x)$) at the jth step of the Householder QR with $index(x) \geq j$. At the jth step of this loop, the union is over all rows in $\mathcal{U}$ or in $\mathcal{P}$ such that the row x is in $\mathcal{R}_j$. The set of nonzero entries of row p_j ($S(p_j)$) is then defined as this union intersected with the set of posterior columns (i.e., $c_{j+1}, \ldots, c_n$). Unless there is numerical cancellation, if any row in $\mathcal{R}_j$ has a nonzero in column $k > j$, after performing the jth step, all rows in this set will have a nonzero entry in column k.

The symbolic QR factorization described in Figure 1.1 gives an upper bound on the sparsity patterns of the matrix R: $r_{ij} \neq 0$ only if $i = j$ or $c_j \in S(p_i)$. It is not

```
for k = 1, 2, ..., n
    for each i such that a_ik ≠ 0
        j ← max{ r < k | a_ir ≠ 0 }
        s ← top of tree containing j
        if s ≠ k
            parent(s) ← k
```

FIG. 1.2. *Algorithm for constructing* ET(CIG(A)).

exact in general, since it assumes $|\mathcal{R}_j| > 1$ for all j; in other words, a row with a nonzero in column j (sparsity pattern $S(p_j)$) is assumed to participate in succeeding stages of the QR factorization, even if it is the only row with a nonzero in column j. The algorithm of Figure 1.1 is exact if A is a strong Hall matrix (since $|\mathcal{R}_j| > 1$ for all j, in this case); however, if A is a Hall matrix but not a strong Hall matrix, then the fill-in predicted by Figure 1.1 can greatly overestimate the true fill-in. In addition, the cost of this algorithm is similar to the cost of actually performing the QR factorization, so row-merge trees have been developed [12].

Note that if A has nonzeros on the diagonal, then the first line of Figure 1.1 can be replaced by: **for** $i = 1, \ldots, m$: $S(u_i) \leftarrow \{\, c_j \mid a_{ij} \neq 0 \,\}$. Nonzeros on the diagonal can usually be achieved by computing a maximal matching of columns to rows; in fact, for Hall matrices the maximal matching will also be a complete matching (that is, every column will be matched to a row). Consequently, for Hall matrices, swapping the row matched to column j with row j for all j will put nonzeros in all of the diagonal entries without creating additional fill-in.

The *row-merge tree* on nodes $\mathcal{R} \cup \mathcal{C}$ can then be defined by the condition that c_j is the parent node of r_i if $j = \min\{\, k \mid c_k \in S(u_i) \,\}$, and the parent of node c_k is c_j where $j = \min\{\, l \mid c_l \in S(p_k) \,\}$. Hereafter, we denote the *row-merge tree* of A as $RM(A)$.

Related to these graphs is the *column intersection graph* of A [8]. Providing there is no numerical cancellation in forming A^TA, the column intersection graph $CIG(A)$ is just $G(A^TA)$. More formally, the column intersection graph of A is defined as the graph on the nodes $\{c_1, \ldots, c_n\}$, where $c_k \sim c_l$ (c_k is connected to c_l) if there is a j where both a_{jk} and a_{jl} are nonzero. Liu [12] showed that the elimination tree of $CIG(A)$ ($ET(CIG(A))$) is equal to $RM(A)$ minus all nodes in $\mathcal{R}$ and edges incident to $\mathcal{R}$. Computationally, $ET(CIG(A))$ is a useful tool because it can be computed directly from A without needing to construct $CIG(A)$ or A^TA, by using the algorithm in Figure 1.2. We also refer the reader to the algorithm of [9, Fig. 2], which achieves the same task and is similar in nature. Other papers where this task is achieved are [3, 4], which use an unrelated algorithm.

The algorithm in Figure 1.2 is a modification of Liu's algorithm for the symmetric case [13], which is shown in Figure 1.3. Note that the line "$s \leftarrow$ top of tree containing j" can be computed using a union-find data structure with path compression in amortized time $O(\log(n))$ time [2, p. 449], or in amortized time $O(\alpha(nz(A), n))$, where α is an inverse of the Ackermann function if the method also uses the union-by-rank heuristic [2, pp. 450–457]. However, the union-by-rank heuristic requires additional work since the representative of a subset is determined by the algorithm, and is not necessarily the root of a component of the currently constructed elimination tree or forest.

Notice that the elimination tree $T = ET(CIG(A))$ is the smallest tree or forest T

```
for k = 1, 2, ..., n
    for each i < k where b_ik ≠ 0
        s ← top of tree containing i
        if s ≠ k
            parent(s) ← k
```

FIG. 1.3. *Liu's algorithm for computing* ET(B).

for $j = 1, 2, \ldots, n$: [$count(j) \leftarrow 0$]
for $i = 1, 2, \ldots, m$: [$k \leftarrow \min\{\, s \mid a_{is} \neq 0 \,\}$; $count(k) \leftarrow count(k) + 1$]
(Note: if the set is empty, jump to the end of the loop.)
for $j = 1, 2, \ldots, n$: [$count(j) \leftarrow count(j) + \left(\sum_{x:x \text{ col. child of } j} count(x)\right) - 1$]
for $j = 1, 2, \ldots, n$: **if** $count(j) = 0$ **then** [$parent(j) \leftarrow$ **null**]

FIG. 2.1. *Postprocessing step.*

that satisfies the relations "l is an ancestor of k" whenever $l > k$ and $a_{jk}, a_{jl} \neq 0$ for some j. The row-merge tree $RM(A)$ can be formed from T by adding $\mathcal{R}$ to the set of nodes and adding edges from r_i to c_k if $k = \min\{\, s \mid a_{is} \neq 0 \,\}$. For an arbitrary tree or forest T, let $RM(T)$ denote the tree or forest formed by adding $\mathcal{R}$ to the set of nodes and by adding edges from r_i to c_k, where $k = \min\{\, s \mid a_{is} \neq 0 \,\}$.

Given the elimination tree $T = ET(CIG(A))$, the fill-in can be estimated as follows [13]: Column j of R has nonzeros in rows r_k, where c_k is an ancestor of some row r_i in $RM(T)$, where $a_{ij} \neq 0$ and $k \leq j$. Row i of W has nonzeros in columns c_k, where c_k is an ancestor of r_i in $RM(T)$ and $k \leq i$. The estimated fill-in based on the $ET(CIG(A))$ is the same as the fill-in predicted by the crude symbolic QR factorization algorithm in Figure 1.1.

The fill-in for the QR factorization can be computed from $ET(CIG(A))$ and A with just $O(n)$ storage overhead and $O(nz(R) + nz(W) + n)$ time; just to count the number of nonzeros requires just $O(1)$ additional storage and $O(nz(R) + nz(W) + n)$ time.

2. Postprocessing the row-merge tree. If $|\mathcal{R}_k| > 1$ for all k, then the assumptions underlying the algorithm in Figure 1.1 are correct, and the methods of the previous paragraph for computing the sparsity patterns of R and W give the correct fill-in. However, if $|\mathcal{R}_k| = 1$ for some k, then we must have $\mathcal{R}_k = \{r_k\}$. In this case, the elimination tree T must be postprocessed to give the correct fill-in.

In this section we introduce a postprocessing algorithm which should be applied when $|\mathcal{R}_k| = 1$ for some k. This algorithm should be performed after constructing $ET(CIG(A))$ and adding the row nodes to construct the row-merge tree of A as described in the previous section. This postprocessed row-merge tree predicts fill-in at least as well as the original row-merge tree does. How to predict fill-in given a row-merge tree was described near the end of the last section. We have to thread the tree so that the children of a given node can be found in $O(1)$ time per child node. Threading a tree of n nodes (defined by a *parent* array) requires just $O(n)$ operations and $O(n)$ storage. The postprocessing step for an $m \times n$ matrix A takes just $O(m+n)$ operations, including the tree threading step.

The postprocessing algorithm is shown in Figure 2.1. Note that the final computed value of $count(j)$ is the number of rows that participate in the jth stage of the QR

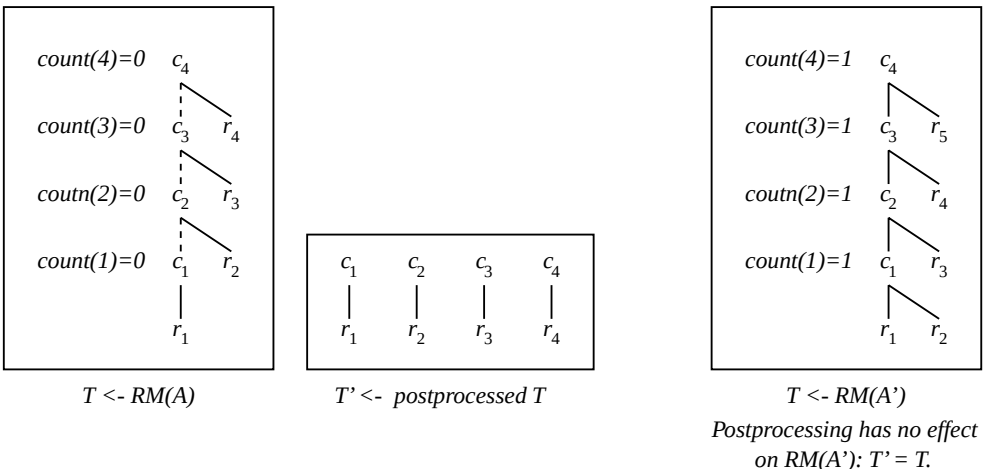


FIG. 2.2. *Row-merge trees and the effects of postprocessing on example matrices A and A′.*

factorization minus one. It is also the number of row nodes in the row-merge tree rooted at c_j minus the number of column nodes in this tree. The first loop in Figure 2.1 initializes the array $count(j)$ for all n columns. The second loop finds out for each row i what is the column number of the first nonzero appearance in that row and adds 1 to the count for that column. In other words, $count(k)$ states how many rows have the first nonzero in column k (the number of row children of k). In example (1.1) in section 1 we have $count(1) = count(2) = count(3) = count(4) = 1$ after the second loop. The third loop finds the number of row descendents of node j minus the number of its column descendents (including itself). The final values of $count(j)$ are shown for examples (1.1) and (1.2) in Figure 2.2. If the value of $count(j)$ is equal to zero, then that node is disconnected from the tree above. For the example matrices A and A' in ((1.1) and (1.2)), the row-merge trees and the effect of postprocessing are shown in Figure 2.2.

The basis for this postprocessing algorithm is a more refined understanding of the symbolic Householder QR factorization shown in Figure 1.1. As mentioned earlier, that algorithm is expensive and previous approaches based on row-merge trees exist. Our new algorithms show that we can do better by postprocessing the row merge trees. The set $S(p_j)$ is the set of nonzero entries in row j after the jth step of the QR algorithm, minus the c_j entry. Note that the jth row no longer participates in the succeeding steps of the QR factorization after the jth step. If only row j has a nonzero in column j at the jth stage of the QR factorization, then no Householder operation needs to be performed, and thus $S(p_j)$ is irrelevant to the calculation of $S(p_l)$ for $l > j$. This fact is ignored by the symbolic QR factorization in Figure 1.1.

If we reconsider the algorithm in Figure 1.1 with this in mind, then for $j = 1, 2, \ldots, n$ there should be a count of the number of rows in $\mathcal{R}_j$ (which is $count(j)+1$). If $|\mathcal{R}_j| \leq 1$, then the jth step of the QR factorization is trivial, and $S(p_j)$ is irrelevant for future steps in the factorization. If we add one to $count(j)$ at the end of the algorithm in Figure 2.1, we obtain the number of rows participating in the jth stage of the factorization. This explains why we should disconnect the tree when $count(j) = 0$, creating the postprocessed tree T'. Now we have new row-merge trees from which we can obtain improved estimates of the fill-in.

```
for k = 1, 2, ..., n:   parent(j) ← NULL
for k = 1, 2, ..., n
    for each i such that a_ik ≠ 0
        p ← min{ q | a_iq ≠ 0 }
        j ← max{ r < k | a_ir ≠ 0 }
        s ← top of tree containing j
        if k, s, and p all belong to the same component of T' and s ≠ k
            parent(s) ← k
```

FIG. 2.3. *Reprocessing step which constructs T'' based on T'.*

After postprocessing, $nz(W)$ can be computed from $ET(CIG(A))$ and A in $O(n)$ time by the formula

$$nz(W) = \sum_{j=1}^{n} (count(j) + 1). \tag{2.1}$$

If we postprocess the elimination tree T' again, we do not change the value of the *count* array and have no affect on $nz(W)$. However, this postprocessed elimination tree T' can be used to compute a new elimination tree T'' which will give more information about $nz(W)$. This reprocessing creates a new elimination tree or forest T''. To illustrate the necessity of reprocessing the postprocessed elimination tree, consider the matrix

$$A'' = \begin{bmatrix} \times & \times & \times & \times \\ & \times & & \\ & & \times & \\ & \times & & \times \end{bmatrix}. \tag{2.2}$$

The steps of the reprocessing algorithm are shown in Figure 2.3. The algorithm mirrors the earlier algorithm in Figure 1.2 for computing $ET(CIG(A))$, except that T' is used to filter the construction of the elimination tree. This means that if two nodes are connected in T'', they must be connected in T'. The elimination tree $ET(CIG(A''))$ and the results of postprocessing and reprocessing the elimination tree using the algorithm of Figure 2.3 are shown in Figure 2.4.

To explain how the algorithm proceeds for the this example, start by looking at T' (the tree obtained after the postprocessing scheme is applied). The generated tree T'' by the reprocessing algorithm is similar to the original tree T' until the second loop reaches the value $k = 3$. When $k = 3$ ($p = 1$, $j = 2$, and $s = 2$), the statement $parent(s) \leftarrow k$ of algorithm 2.3 is not executed, since k, s, and p do not belong to the same component of T'. Nevertheless, during the next iteration of this loop, when $k = 4$, that same statement will connect node 2 to its new parent, node 4.

The postprocessing and subsequent reprocessing steps can be applied recursively until postprocessing does not alter the elimination tree or forest (that is, $count(j) > 0$ for all j). Define $T'' = ET^*(A, T)$ to be the elimination forest that results from the following two-step process: (1) applying postprocessing to the elimination tree T and then (2) reprocessing the resulting forest T' to produce T'', using algorithms in Figures 2.1 and 2.3, respectively.

Since $ET^*(A, T)$ is either unchanged ($T'' = T$) or more disconnected than T, recursive applications of $T^{(k+1)} \leftarrow ET^*(A, T^{(k)})$ will terminate after at most n iterations. In practice, the number of iterations needed to obtain $T^{(k+1)} = T^{(k)}$ is very

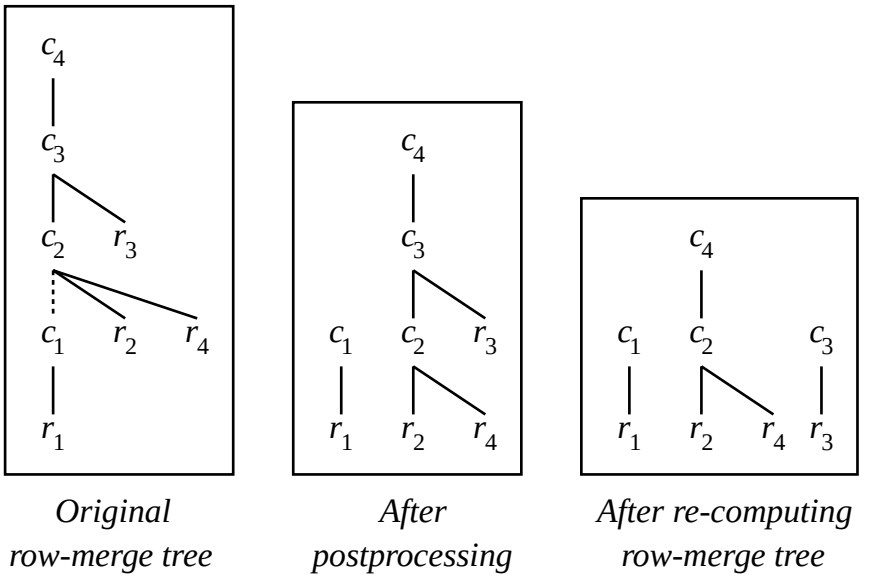


FIG. 2.4. *Original, postprocessed, and recomputed elimination trees for* A''.

small. Let T^* be this final elimination forest where $T^* = ET^*(A, T^*)$. Then, for this forest, $count(j) > 0$ for all nodes j that are not root nodes. Thus, the predicted sparsity structure of the rows in each tree of T^* is correct, and the sparsity structure predicted by T^* is the exact fill-in pattern for the Householder QR factorization.

Determining if two nodes are in a common connected component of a tree or forest T (as done in the core of loop k) can be performed in $O(1)$ time after an *ancestor* array is constructed containing the top-most ancestor of each node in T, which can be done in $O(n \log n)$ time, where n is the number of nodes of T.

3. Computational results. The following operations were implemented: computing the elimination tree of the column intersection graph $ET(CIG(A))$ (including threading the trees), the postprocessing algorithms of Figures 2.1 and 2.3, a maximal matching algorithm [6] for bipartite graphs, and the Dulmage–Mendelsohn decomposition [20]. Note that only path compression was used for the union-find data structures in the construction of the elimination trees [2, p. 447], and not union-by-rank. This is the approach used for constructing $ET(CIG(A))$ in the recent SuperLU code under development by Demmel et al., which is referred to in [5]. The implementation of the maximal matching algorithm follows the recommendations of [6]. These algorithms were implemented in terms of the Meschach matrix library in C [21]. Our algorithms predicted the exact fill-in for all test matrices, as expected. Our test matrices included all rectangular matrices plus some square matrices from the Harwell–Boeing collection. Matrices which had more columns than rows were transposed before performing any calculations. No column or row reordering was performed on these matrices, since we aim solely to analyze the performance of our fill-in prediction algorithms.

Other test problems were 10×10 matrices with nonzeros only in the first row and the diagonal. Our new routines correctly deduced the number of nonzero entries for R, which for this matrix has just 19 nonzeros, using the postprocessed $ET(CIG(A))$ tree (i.e., no fill-in), while the unprocessed tree predicted 55 nonzeros

TABLE 3.1
Data for test matrices and their QR factorizations.

Name	$m \times n$	$\#nz$	$nz(R)$	$nz(W)$
impcol_b	59×59	312	877	671
watson1	58×59	340	1711	696
watson2	66×67	409	2211	1233
impcol_c	137×137	411	3835	3233
ash219	219×85	438	1238	7367
impcol_a	207×207	572	3615	2216
ash331	331×104	662	1026	13984
watson3	124×125	780	7750	3840
lop163	163×163	935	3251	2046
fs183_1	183×183	1069	15889	14440
ash608	608×188	1216	2970	43637
impcol_e	225×225	1308	6603	5594
impcol_d	425×425	1339	21626	13677
abb313	313×176	1557	6794	22928
ash958	958×292	1916	4516	104314
1138_bus	1138×1138	2596	99137	62572
fs680_1	680×680	2646	204152	203518
mcca	180×180	2659	5882	1730
watson4	467×468	2869	109278	88452
wm1	207×277	2909	18222	22776
wm2	207×260	2942	19879	23120
wm3	207×260	2948	19925	23121
bcspwr07	1612×1612	3718	66519	43260
bcspwr08	1624×1624	3837	87029	54749
bcspwr09	1723×1723	4117	122463	109684
illc1033	1033×320	4732	8756	92309
gre_1107	1107×1107	5664	328891	130060
fs760_1	760×760	5976	235707	223292
illc1850	1850×712	8758	71849	474111
watson5	1853×1854	10847	1717731	447291
bcspwr10	5300×5300	13571	2653153	2432762
zenios	2873×2873	15032	97430	94444
add20	2395×2395	17319	2565125	2178211
add32	4960×4960	23884	9381844	8687422
mcfe	765×765	24382	91277	24548
gemat12	4929×4929	33111	5407842	5071559
gemat11	4929×4929	33185	5415469	5071185
beause	497×507	44551	120727	116106
gemat1	4929×10595	47369	235707	223292
beacxc	497×506	50409	100373	110571
beaflw	497×507	53403	120855	114601
orani678	2529×2529	90158	2776652	1835474
memplus	17758×17758	126150	156383209	141158005

(i.e., complete fill-in, as was expected from $A^T A$). Also, matrix A'' of (2.2) was used as a test matrix, and the results confirmed expectations. Our routines were compiled and run on a Hewlett–Packard Visualize C3000 with standard optimization switches set on.

The basic data for the matrices are listed in Table 3.1, and the timings for the operations are listed in Table 3.2. In Table 3.1, $\#nz$ is the number of nonzeros of the original matrix, $nz(R)$ is the number of nonzeros of R in the QR factorization, and $nz(W)$ is the number of nonzeros in the matrix of Householder vectors that represents Q. In Table 3.2, $ETCIG1$ refers to the time to compute the elimination

TABLE 3.2
Timings for operations on various matrices. Entry $x.xx(yy)$ means $x.xx \times 10^{yy}$ seconds.

Name	$ETCIG1$	$ETCIG2$	Thread/Postproc	Coarse/Fine	MM
impcol_b	1.05(−4)	1.31(−4)	1.20(−5)/2.60(−5)	2.95(−4)/3.97(−4)	1.38(−4)
watson1	1.10(−4)	—	1.10(−5)/2.40(−5)	4.02(−4)/3.87(−4)	1.41(−4)
watson2	1.29(−4)	—	1.20(−5)/2.90(−5)	4.27(−4)/4.45(−4)	1.31(−4)
impcol_c	1.43(−4)	2.36(−4)	1.50(−5)/4.20(−5)	3.09(−4)/6.14(−4)	2.09(−4)
ash219	1.38(−4)	—	1.30(−5)/3.50(−5)	5.70(−4)/7.60(−4)	1.82(−4)
impcol_a	1.90(−4)	3.58(−4)	2.00(−5)/6.10(−5)	3.66(−4)/1.41(−3)	6.30(−4)
ash331	1.83(−4)	—	1.40(−5)/4.20(−5)	7.00(−4)/1.05(−3)	2.58(−4)
watson3	2.03(−4)	—	1.50(−5)/3.80(−5)	5.60(−4)/7.52(−4)	1.84(−4)
lop163	2.63(−4)	3.76(−4)	1.70(−5)/4.80(−5)	3.30(−4)/8.83(−4)	1.24(−4)
fs183_1	2.89(−4)	4.25(−4)	1.80(−5)/5.20(−5)	3.75(−4)/1.03(−3)	1.31(−4)
ash608	3.13(−4)	—	1.80(−5)/8.80(−5)	1.04(−3)/1.82(−3)	4.04(−4)
impcol_e	3.28(−4)	—	2.00(−5)/6.60(−5)	3.86(−4)/1.83(−3)	3.01(−4)
impcol_d	3.65(−4)	7.20(−4)	2.80(−5)/1.04(−4)	4.61(−4)/1.68(−3)	4.28(−4)
abb313	3.40(−4)	—	1.80(−5)/5.70(−5)	8.85(−4)/1.54(−3)	2.84(−4)
ash958	4.57(−4)	—	2.30(−5)/9.90(−5)	1.42(−3)/2.87(−3)	6.70(−4)
1138_bus	8.51(−4)	1.84(−3)	6.20(−5)/2.70(−4)	7.63(−4)/3.86(−3)	6.95(−4)
fs680_1	6.70(−4)	2.75(−4)	4.00(−5)/1.65(−4)	5.85(−4)/2.87(−3)	4.49(−4)
mcca	5.05(−4)	1.51(−3)	1.90(−5)/5.20(−5)	3.77(−4)/1.67(−3)	1.61(−4)
watson4	8.01(−4)	—	3.00(−5)/1.19(−4)	1.25(−3)/2.52(−3)	3.88(−4)
wm1	5.65(−4)	7.70(−4)	1.90(−5)/6.10(−5)	1.07(−3)/1.92(−3)	3.25(−4)
wm2	5.72(−4)	7.78(−4)	1.90(−5)/6.00(−5)	1.09(−3)/1.92(−3)	5.62(−4)
wm3	5.78(−4)	8.28(−4)	1.80(−5)/6.10(−5)	1.12(−3)/1.97(−3)	5.80(−4)
bcspwr07	1.09(−3)	—	8.10(−5)/3.74(−4)	9.13(−4)/5.45(−3)	8.23(−4)
bcspwr08	1.18(−3)	3.43(−3)	8.20(−5)/3.86(−4)	9.43(−4)/6.37(−3)	8.35(−4)
bcspwr09	1.14(−3)	2.80(−3)	9.00(−5)/4.12(−4)	9.44(−4)/5.83(−3)	8.81(−4)
illc1033	8.78(−4)	1.21(−3)	2.50(−5)/1.06(−4)	2.18(−3)/4.34(−3)	9.46(−4)
gre_1107	1.74(−3)	2.81(−3)	5.90(−5)/2.68(−4)	8.40(−4)/5.43(−3)	7.71(−4)
fs760_1	1.54(−3)	2.29(−3)	4.40(−5)/1.90(−4)	6.75(−4)/4.61(−3)	2.76(−4)
illc1850	1.72(−3)	2.50(−3)	4.10(−5)/2.22(−4)	3.65(−3)/7.92(−3)	2.67(−3)
watson5	2.50(−3)	—	9.40(−5)/4.67(−4)	4.12(−3)/9.69(−3)	1.37(−3)
bcspwr10	4.47(−3)	1.02(−2)	2.40(−4)/1.30(−3)	2.21(−3)/1.85(−2)	2.31(−3)
zenios	3.27(−3)	5.94(−3)	1.44(−4)/6.93(−4)	1.61(−3)/1.36(−2)	1.53(−3)
add20	4.95(−3)	7.25(−3)	1.14(−4)/6.05(−4)	1.60(−3)/1.45(−2)	1.51(−3)
add32	7.07(−3)	1.20(−2)	2.77(−4)/1.32(−3)	2.70(−3)/2.45(−2)	2.66(−3)
mcfe	4.53(−4)	5.66(−3)	4.30(−5)/2.71(−4)	1.33(−3)/1.47(−2)	8.28(−4)
gemat12	1.03(−2)	1.50(−2)	2.30(−4)/1.27(−3)	2.79(−3)/3.46(−2)	6.71(−3)
gemat11	9.35(−3)	1.45(−2)	2.26(−4)/1.63(−3)	2.85(−3)/3.62(−2)	6.51(−3)
beause	2.05(−2)	2.83(−2)	3.60(−5)/2.89(−4)	1.15(−2)/3.60(−2)	7.68(−3)
gemat1	1.72(−2)	—	2.52(−4)/2.29(−3)	3.07(−2)/6.82(−2)	2.71(−2)
beacxc	2.26(−2)	2.96(−2)	3.20(−5)/2.74(−4)	1.49(−2)/4.25(−2)	1.00(−2)
beaflw	2.39(−2)	3.15(−2)	3.10(−5)/2.85(−4)	1.51(−2)/4.30(−2)	1.04(−3)
orani678	4.38(−2)	5.60(−2)	1.43(−3)/1.10(−3)	8.62(−3)/1.01(−1)	7.17(−2)
memplus	8.20(−2)	1.14(−1)	1.07(−3)/7.83(−3)	2.51(−2)/3.69(−1)	1.84(−2)

tree of the column intersection graph of the matrix, and $ETCIG2$ is the amount of additional time needed for all subsequent computations of $ET^*(A,T)$ in order to compute the final elimination tree, T^*. The columns "Thread" and "Postproc" give the time needed to thread the elimination tree and to perform the postprocessing of the algorithm in Figure 2.1, respectively.

The Dulmage–Mendelsohn decomposition gives an alternative way of predicting the exact fill-in of QR factorization. The decomposition itself is a reordering of the rows and columns of the matrix so that the matrix is put into block-upper-triangular form with strong Hall diagonal blocks. The columns "Coarse" and "Fine" of Table 3.2

give the times needed for the coarse and fine phases of the Dulmage–Mendelsohn decomposition [20] after the computation of a maximal matching. Finally, the column "MM" gives the time needed to compute a maximal matching, which is an essential step in the Dulmage–Mendelsohn algorithm. Note that the times given do not include the times for computing the fill-in for the QR factorization using either approach.

When we add the times $ETCIG1$ and $ETCIG2$ and compare with the sum of the times for the coarse and fine phases of the Dulmage–Mendelsohn method, we always get smaller times for the sum of elimination tree times ($ETCIG1$ plus $ETCIG2$). Notice that after the Dulmage–Mendelsohn decomposition, there are still other steps for the prediction of the QR fill-in which are not taken into account. Our results show that our new algorithms based on postprocessed row-merge trees are usually faster than methods based on Dulmage–Mendelsohn approaches.

Acknowledgments. I would like to thank David Stewart for proofreading the paper, Alex Pothen and Esmond Ng for interesting conversations, and, finally, the referees for comments which improved the paper.

REFERENCES

[1] T. F. Coleman, A. Edenbrandt, and J. R. Gilbert, *Predicting fill for sparse orthogonal factorization*, J. ACM, 33 (1986), pp. 517–532.

[2] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*, MIT Press, Cambridge, MA, McGraw-Hill, New York, 1990.

[3] T. A. Davis and W. W. Hager, *Modifying a sparse Cholesky factorization*, SIAM J. Matrix Anal. Appl., 20 (1999), pp. 606–627.

[4] T. A. Davis and W. W. Hager, *Multiple-rank modifications of a sparse Cholesky factorization*, SIAM J. Matrix Anal. Appl., 22 (2001), pp. 997–1013.

[5] J. W. Demmel, S. C. Eisenstat, J. R. Gilbert, X. S. Li, and J. W. H. Liu, *A supernodal approach to sparse partial pivoting*, SIAM J. Matrix Anal. Appl., 20 (1999), pp. 720–755.

[6] I. S. Duff and T. Wiberg, *Remarks on implementations of $O(n^{1/2}\tau)$ assignment algorithms*, ACM Trans. Math. Software, 14 (1988), pp. 267–287.

[7] A. George and E. Ng, *Symbolic factorization for sparse Gaussian elimination with partial pivoting*, SIAM J. Sci. Statist. Comput., 8 (1987), pp. 877–898.

[8] J. R. Gilbert and E. G. Ng, *Predicting structure in nonsymmetric sparse matrix factorizations*, in Graph Theory and Sparse Matrix Computation, IMA Vol. Math. Appl. 56, A. George, J. R. Gilbert, and J. W. H. Liu, eds., Springer-Verlag, New York, 1993, pp. 107–139.

[9] J. R. Gilbert, X. S. Li, E. Ng, and B. Peyton, *Predicting the Structure of Sparse Orthogonal Factors*, in preparation.

[10] G. Golub and C. V. Loan, *Matrix Computations*, 3rd ed., Johns Hopkins University Press, Baltimore, MD, 1996.

[11] D. R. Hare, C. R. Johnson, D. D. Olesky, and P. van den Driessche, *Sparsity analysis of the QR factorization*, SIAM J. Matrix Anal. Appl., 14 (1993), pp. 655–669.

[12] J. W. H. Liu, *On general row merging schemes for sparse Givens transformations*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 1190–1211.

[13] J. W. H. Liu, *The role of elimination trees in sparse factorization*, SIAM J. Matrix Anal. Appl., 11 (1990), pp. 134–172.

[14] J. W. H. Liu, *The multifrontal method for sparse matrix solution: Theory and practice*, SIAM Rev., 34 (1992), pp. 82–109.

[15] E. G. Ng and B. W. Peyton, *Some results on structure prediction in sparse QR factorization*, SIAM J. Matrix Anal. Appl., 17 (1996), pp. 443–459.

[16] S. Oliveira, *A new parallel chasing algorithm for transforming arrowhead matrices to tridiagonal form*, Math. Comp., 67 (1998), pp. 221–235.

[17] S. Oliveira, *Reprocessing a Postprocessed Elimination Tree to Obtain Exact Sparsity Prediction in QR Factorization*, Tech. report TR-127, The University of Iowa, Iowa City, IA, 1999.

[18] P. E. Plassmann, *Sparse Jacobian estimation and factorization on a multiprocessor*, in Large-Scale Numerical Optimization, T. F. Coleman and Y. Li, eds., SIAM, Philadelphia, 1990, pp. 152–179.

[19] A. Pothen, *Predicting the structure of sparse orthogonal factors*, Linear Algebra Appl., 194 (1993), pp. 183–203.

[20] A. Pothen and C.-J. Fan, *Computing the block triangular form of a sparse matrix*, ACM Trans. Math. Software, 16 (1990), pp. 303–324.

[21] D. E. Stewart and Z. Leyk, *Meschach: Matrix Computations in C*, Proc. Centre Math. Appl. Austral. Nat. Univ. 32, Australian National University, Canberra, Australia, 1994.

[22] R. E. Tarjan, *Data Structures and Network Algorithms*, CBMS-NSF Regional Conf. Ser. in Appl. Math. 44, SIAM, Philadelphia, 1983.

SIAM J. SCI. COMPUT.
Vol. 22, No. 6, pp. 1974–1986
© 2001 Society for Industrial and Applied Mathematics

ON SCHWARZ ALTERNATING METHODS FOR THE INCOMPRESSIBLE NAVIER–STOKES EQUATIONS*

S. H. LUI†

Abstract. The Schwarz alternating method can be used to solve linear elliptic boundary value problems on domains which consist of two or more overlapping subdomains. The solution is approximated by an infinite sequence of functions which result from solving a sequence of elliptic boundary value problems in each of the subdomains.

This paper considers four Schwarz alternating methods for the N-dimensional, steady, viscous, incompressible Navier–Stokes equations, $N \leq 4$. It is shown that the Schwarz sequences converge to the true solution provided that the Reynolds number is sufficiently small.

Key words. domain decomposition, Schwarz alternating method, Navier–Stokes

AMS subject classifications. 65N25, 65N55

PII. S1064827598347411

1. Introduction.

The Schwarz alternating method was devised by H. A. Schwarz more than one hundred years ago to solve linear boundary value problems. It has garnered interest recently because of its potential as a very efficient algorithm for parallel computers. In Tai and Espedal [11] and Dryja and Hackbusch [4], the authors show convergence of Schwarz methods for some nonlinear problems. In Lui [10], proofs of convergence of Schwarz alternating methods for some 2nd-order nonlinear elliptic PDEs were attained. In this sequel, we prove convergence of four Schwarz methods for the N-dimensional, steady, incompressible, viscous Navier–Stokes equations, $N \leq 4$, provided that the Reynolds number is sufficiently small.

Many authors have demonstrated numerically the effectiveness of Schwarz methods in solving fluid problems at moderate Reynolds numbers. See, for example, the proceedings of the annual domain decomposition conferences, beginning with [7], [3], [2], [8], [6], and references therein. This list is of course far from complete.

This paper appears to be the first attempt to prove convergence of Schwarz methods for the Navier–Stokes equations, though only for small Reynolds numbers. It follows closely the framework developed in the fundamental paper of Lions [9], where the convergence of the Schwarz method for the Stokes equations is proved. We treat the Navier–Stokes equations as a nonlinear perturbation of the Stokes equations. Although we concentrate mostly on the two-subdomain case, we shall derive an additive version which converges for multiple subdomains.

Let Ω be a bounded connected domain in $\mathbb{R}^N$ with a smooth boundary. Suppose $\Omega = \Omega_1 \cup \Omega_2$, where the subdomains Ω_i have smooth boundaries and are overlapping. Let $H_0^1(\Omega)^N$ denote the Cartesian product of N Sobolev spaces $H_0^1(\Omega)$ (consisting of all functions whose first derivatives are in $L^2(\Omega)$ and whose value vanishes on $\partial\Omega$ in the trace sense), and define $L^2(\Omega)^N$ as the Cartesian product of N copies of $L^2(\Omega)$. The space $H^{1/2}(\partial\Omega)^N$ is defined similarly. Let

$$V = \{u \in H_0^1(\Omega)^N,\ \text{div } u = 0\}$$

*Received by the editors November 17, 1998; accepted for publication (in revised form) February 12, 2000; published electronically February 21, 2001.
http://www.siam.org/journals/sisc/22-6/34741.html

†Hong Kong University of Science and Technology, Department of Mathematics, Clear Water Bay, Kowloon, Hong Kong (shlui@ust.hk). This work was supported in part by a grant from RGC CERG HKUST726/96E and HKUST6171/99P.

and

$$V_i = \{u \in H_0^1(\Omega_i)^N,\ \text{div } u = 0\}, \qquad i = 1, 2.$$

Let u_i denote the ith component of the vector u. Denote the inner product in the Sobolev space $H_0^1(\Omega)^N$ by

$$[u, v] = \sum_{i=1}^{N} \int_\Omega \nabla u_i \cdot \nabla v_i,$$

and let $\|u\|_1 = [u, u]^{1/2}$. In this paper, a function in $H_0^1(\Omega_i)^N$ is considered as a function defined on the whole domain by extension by zero.

Given f from the dual space $H^{-1}(\Omega)^N$ with norm $\|\cdot\|_{-1}$, the Stokes problem for $u \in H_0^1(\Omega)^N$ and $p \in L_0^2(\Omega) \equiv \{q \in L^2(\Omega),\ \int_\Omega q = 0\}$ is

$$-\triangle u + \nabla p = f \ \text{ and } \ \text{div } u = 0 \text{ on } \Omega.$$

It is well known ([12], for instance) that there is a unique solution $(u, p) \in H_0^1(\Omega)^N \times L_0^2(\Omega)$. Write $u = S^{-1}f$, where $S^{-1} : H^{-1}(\Omega)^N \to V$ is the solution operator for the velocity u. It is again well known that $(\|u\|_1 + \|p\|_{L^2(\Omega)}) \le C\,\|f\|_{-1}$ for some constant C. In particular, $\|S^{-1}\|_1 \le C$. Similarly, define S_i^{-1} as the solution operator for the velocity for the Stokes problem on Ω_i.

The subdomains Ω_i are said to overlap if $H_0^1(\Omega)^N = H_0^1(\Omega_1)^N + H_0^1(\Omega_2)^N$. In this case, the work of Lions [9] shows that $V = V_1 + V_2$. Let P_i denote the orthogonal (with respect to the inner product $[\cdot, \cdot]$) projection from V onto V_i, $i = 1, 2$. It is well known that

$$d \equiv \max\left(\|(I - P_2)(I - P_1)\|_1,\ \|(I - P_1)(I - P_2)\|_1\right) < 1. \tag{1.1}$$

See Lions [9] and Bramble et al. [1]. Throughout this paper, C will denote a positive constant which may not be the same at different occurrences.

In the next section, we give a statement of the problem including an estimate for the nonlinear term. Following that, we prove convergence for a nonlinear Schwarz sequence where each subdomain problem is a nonlinear one. In the next three sections, we develop three variations of the nonlinear Schwarz sequence and prove convergence. These sequences are more practical in that only linear subdomain problems are encountered and that in two of these, the subdomain problems are independent so that they can be solved concurrently. In the final section, we discuss the case of many subdomains and nonhomogeneous boundary conditions. This paper concludes with some suggestions for future work.

2. Navier–Stokes equations. The N-dimensional, steady, viscous, incompressible Navier–Stokes equations in nondimensional form are

$$\begin{aligned} (u \cdot \nabla)u &= -\nabla p + \frac{1}{R}\triangle u + f \text{ on } \Omega, \\ \text{div } u &= 0 \text{ on } \Omega, \end{aligned}$$

with the boundary conditions

$$u = g \text{ on } \partial\Omega,$$

where $u \in H^1(\Omega)^N$, g is the given velocity on the boundary with $\|g\|_{H^{1/2}(\partial\Omega)^N} \leq 1$, f is the forcing term in $L^2(\Omega)^N$, p is the pressure, and R is the Reynolds number. We assume that g satisfies the compatibility condition $\int_{\partial\Omega} g \cdot \nu = 0$, where ν denotes the (outward) unit normal. We use u to denote the unique solution to this equation.

We shall need the following lemma. A similar version states the continuity of a certain trilinear form containing the convective term of the Navier–Stokes equations. See, for instance, Temam [12].

LEMMA 1. *Let $U = \{u \in H^1(\Omega)^N,\ div\ u = 0\}$. For $N \leq 4$, define the function $F : U \times U \to V$ by $F(u,v) = S^{-1}(u \cdot \nabla)v$. Then $\|F(u,v)\|_1 \leq C\|u\|_1\ \|v\|_1$.*

Proof. Let $z = F(u,v) \in V$. Then $Sz = (u \cdot \nabla)v$, which implies

$$[z,z] = \int_\Omega z \cdot (u \cdot \nabla)v = -\int_\Omega v \cdot (u \cdot \nabla)z.$$

By Holder's inequality, we have for any $0 < \epsilon \leq 1/2$

$$\left| \int_\Omega v_j u_i \frac{\partial z_j}{\partial x_i} \right| \leq \|v_j\|_{L^{2/\epsilon}(\Omega)} \|u_i\|_{L^{2/\epsilon}(\Omega)} \|z_j\|_{H_0^1(\Omega)}.$$

By the Sobolev embedding theory, we have a continuous embedding $H^s(\Omega) \to L^{2/\epsilon}(\Omega)$ for $s = N(1-\epsilon)/2$. When $N \leq 4$, we can arrange $s = 1$ by choosing an appropriate ϵ. (For $N = 2$, let $\epsilon \to 0$; for $N = 3$ or 4, pick $\epsilon = 1 - 2/N$.) Thus

$$\|z\|_1^2 \leq C\|z\|_1\ \|u\|_1\ \|v\|_1,$$

and the conclusion of the lemma follows. ∎

In light of this lemma, we restrict our work to the case $N \leq 4$ in the remainder of this paper. Initially, *we only discuss the case $g \equiv 0$*, which has a clearer exposition. The general case will be addressed in the last section. In the next section, we define a Schwarz sequence and prove its convergence.

3. Nonlinear Schwarz sequence. Let $u^{(0)} \in V$. For $n = 0, 1, 2, \ldots$, define the nonlinear Schwarz sequence as

$$\begin{aligned} (u^{(n+\frac{1}{2})} \cdot \nabla)u^{(n+\frac{1}{2})} &= -\nabla p^{(n+\frac{1}{2})} + \frac{1}{R}\triangle u^{(n+\frac{1}{2})} + f \text{ on } \Omega_1, \\ \text{div } u^{(n+\frac{1}{2})} &= 0 \text{ on } \Omega_1, \\ u^{(n+\frac{1}{2})} &= u^{(n)} \text{ on } \partial\Omega_1 \end{aligned}$$

and

$$\begin{aligned} (u^{(n+1)} \cdot \nabla)u^{(n+1)} &= -\nabla p^{(n+1)} + \frac{1}{R}\triangle u^{(n+1)} + f \text{ on } \Omega_2, \\ \text{div } u^{(n+1)} &= 0 \text{ on } \Omega_2, \\ u^{(n+1)} &= u^{(n+\frac{1}{2})} \text{ on } \partial\Omega_2. \end{aligned}$$

Here, $u^{(n+\frac{1}{2})}$ is considered as a function in V by defining it to be $u^{(n)}$ on $\overline{\Omega} \setminus \overline{\Omega}_1$, and $u^{(n+1)}$ is defined as $u^{(n+\frac{1}{2})}$ on $\overline{\Omega} \setminus \overline{\Omega}_2$. The first thing to check is that the compatibility conditions

$$\int_{\partial\Omega_1} u^{(n)} \cdot \nu = 0 = \int_{\partial\Omega_2} u^{(n+\frac{1}{2})} \cdot \nu$$

are satisfied. We claim that they are satisfied for all n provided that

$$\int_{\Gamma_1} u^{(0)} \cdot \nu = 0 \tag{3.1}$$

holds, where $\Gamma_1 = \partial\Omega_1 \cap \Omega_2$.

The claim is proved by induction. For $n = 0$,

$$\int_{\partial\Omega_1} u^{(0)} \cdot \nu = \int_{\Gamma_1} u^{(0)} \cdot \nu = 0$$

by (3.1). Since div $u^{(\frac{1}{2})} = 0$ on $\Omega_1 \cap \Omega_2$,

$$0 = \int_{\Gamma_1} u^{(\frac{1}{2})} \cdot \nu + \int_{\Gamma_2} u^{(\frac{1}{2})} \cdot \nu = \int_{\Gamma_1} u^{(0)} \cdot \nu + \int_{\Gamma_2} u^{(\frac{1}{2})} \cdot \nu = \int_{\Gamma_2} u^{(\frac{1}{2})} \cdot \nu.$$

Here $\Gamma_2 = \partial\Omega_2 \cap \Omega_1$. The induction step is proved by a similar calculation.

The main result is that this Schwarz sequence converges to the true solution provided that the Reynolds number is sufficiently small.

THEOREM 1. *Assuming that $u^{(0)} \in V$ satisfies* (3.1) *and that R is sufficiently small (see* (3.9 *and* 3.10)*, the nonlinear Schwarz sequence converges geometrically to the true solution u in the norm $\|\cdot\|_1$.*

Proof. Let $v_1 \in V_1$. Multiply the PDE in Ω_1 by v_1, and then integrate by parts to obtain the weak form

$$\int_{\Omega_1} v_1 \cdot (u^{(n+\frac{1}{2})} \cdot \nabla) u^{(n+\frac{1}{2})} = -\frac{1}{R}[u^{(n+\frac{1}{2})}, v_1] + \int_{\Omega_1} f \cdot v_1.$$

Note that v_1 is divergence-free and hence the pressure term drops out. Now

$$[u^{(n+\frac{1}{2})} - u^{(n)}, v_1] = [-P_1 u^{(n)}, v_1] - R \int_{\Omega_1} v_1 \cdot (u^{(n+\frac{1}{2})} \cdot \nabla) u^{(n+\frac{1}{2})} + R \int_{\Omega_1} f \cdot v_1.$$

Since $u^{(n+\frac{1}{2})} - u^{(n)} \in V_1$, we have

$$u^{(n+\frac{1}{2})} - u^{(n)} = -P_1 u^{(n)} - R S_1^{-1} (u^{(n+\frac{1}{2})} \cdot \nabla) u^{(n+\frac{1}{2})} + R S_1^{-1} f.$$

Define $e^{(n)} = u^{(n)} - u$ and $e^{(n+\frac{1}{2})} = u^{(n+\frac{1}{2})} - u$. Since $P_1 u = -R S_1^{-1}(u \cdot \nabla) u + R S_1^{-1} f$, we obtain

$$e^{(n+\frac{1}{2})} - e^{(n)} = P_1 u + R S_1^{-1}(u \cdot \nabla) u - R S_1^{-1}(u^{(n+\frac{1}{2})} \cdot \nabla) u^{(n+\frac{1}{2})} - P_1 u^{(n)}$$

or

$$e^{(n+\frac{1}{2})} = (I - P_1) e^{(n)} + R F_1(e^{(n+\frac{1}{2})}), \tag{3.2}$$

where

$$\begin{aligned} F_1(e^{(n+\frac{1}{2})}) &= -S_1^{-1}((u^{(n+\frac{1}{2})} \cdot \nabla) u^{(n+\frac{1}{2})} - (u \cdot \nabla) u), \\ &= -S_1^{-1}((e^{(n+\frac{1}{2})} \cdot \nabla) e^{(n+\frac{1}{2})} + (u \cdot \nabla) e^{(n+\frac{1}{2})} + (e^{(n+\frac{1}{2})} \cdot \nabla) u). \end{aligned} \tag{3.3}$$

Similarly,

$$e^{(n+1)} = (I - P_2) e^{(n+\frac{1}{2})} + R F_2(e^{(n+1)}), \tag{3.4}$$

where

$$F_2(e^{(n+1)}) = -S_2^{-1}\left((e^{(n+1)}\cdot\nabla)e^{(n+1)} + (u\cdot\nabla)e^{(n+1)} + (e^{(n+1)}\cdot\nabla)u\right). \tag{3.5}$$

Combining these equations, we obtain

$$e^{(n+\frac{1}{2})} = (I-P_1)(I-P_2)e^{(n-\frac{1}{2})} + R(I-P_1)F_2(e^{(n)}) + RF_1(e^{(n+\frac{1}{2})}) \tag{3.6}$$

and

$$e^{(n+1)} = (I-P_2)(I-P_1)e^{(n)} + R(I-P_2)F_1(e^{(n+\frac{1}{2})}) + RF_2(e^{(n+1)}). \tag{3.7}$$

Having developed the error equations, the rest of the proof can be divided into three steps. The first is to derive an estimate for $\|F_i\|_1$, and then show that $\{\|e^{(n)}\|_1\}$ is bounded and, finally, that it is convergent to zero.

In the first step, we give a bound on $\|F_1(e^{(n+\frac{1}{2})})\|_1$. This is accomplished by directly applying the lemma (on Ω_1):

$$\|F_1(e^{(n+\frac{1}{2})})\|_1 \le C\|e^{(n+\frac{1}{2})}\|_1(2\|u\|_1 + \|e^{(n+\frac{1}{2})}\|_1).$$

A similar bound holds for $\|F_2(e^{(n)})\|_1$.

In the second step of the program, we show by induction that, provided R is sufficiently small, $\|e^{(n+\frac{1}{2})}\|_1, \|e^{(n)}\|_1 \le M$ for every n, where

$$M = \max(\|e^{(0)}\|_1, \|e^{(\frac{1}{2})}\|_1).$$

Suppose $\|e^{(n-\frac{1}{2})}\|_1, \|e^{(n)}\|_1 \le M$. From (1.1) and (3.6),

$$\begin{aligned}\|e^{(n+\frac{1}{2})}\|_1 \le{}& d\|e^{(n-\frac{1}{2})}\|_1 + RC\|e^{(n)}\|_1(2\|u\|_1 + \|e^{(n)}\|_1)\\ &+RC\|e^{(n+\frac{1}{2})}\|_1(2\|u\|_1 + \|e^{(n+\frac{1}{2})}\|_1)\end{aligned} \tag{3.8}$$

or

$$0 \le \epsilon a^2 - (1-2\epsilon\|u\|_1)a + \alpha,$$

where $\epsilon = RC$, $a = \|e^{(n+\frac{1}{2})}\|_1$, and $\alpha = dM + \epsilon M(M+2\|u\|_1)$. The roots of the associated quadratic equation are

$$\begin{aligned} r_\pm &= \frac{1-2\|u\|_1\epsilon \pm \sqrt{(1-2\|u\|_1\epsilon)^2 - 4\epsilon\alpha}}{2\epsilon}\\ &= \frac{1}{\epsilon} - (dM + 2\|u\|_1) + O(\epsilon), \qquad dM + O(\epsilon).\end{aligned}$$

Hence the inequality is equivalent to $a \ge r_+$ or $a \le r_-$. The first root, r_+, is extraneous because in this case, as $\epsilon \to 0$, $a = \|e^{(n+\frac{1}{2})}\|_1 \to \infty$, which is absurd. Hence we have $\|e^{(n+\frac{1}{2})}\|_1 \le r_- \le M$ for ϵ sufficiently small. Here, sufficiently small means $\epsilon \le (1-d)/2(M+2\|u\|_1)$ or

$$R \le \frac{1-d}{2C(M+2\|u(R)\|_1)}. \tag{3.9}$$

Here, we emphasize the dependence of u on R by writing $u(R)$. From (3.7), with the same bound on R, we also have $\|e^{(n+1)}\|_1 \le M$. Hence by induction, $\|e^{(n)}\|_1, \|e^{(n+\frac{1}{2})}\|_1 \le M$ for every n.

Finally, we are ready to show the convergence of the Schwarz sequence. As a consequence of the above, we have

$$\|F_1(e^{(n+\frac{1}{2})})\|_1 \le C(M + 2\|u\|_1)\|e^{(n+\frac{1}{2})}\|_1$$

and similarly for $\|F_2(e^{(n)})\|_1$. By applying $(I - P_2)$ to (3.4) with n replaced by $n-1$, we obtain $(I - P_2)e^{(n)} = (I - P_2)e^{(n-\frac{1}{2})}$. From (3.6)

$$\begin{aligned}\|e^{(n+\frac{1}{2})}\|_1^2 &= \|P_1 e^{(n+\frac{1}{2})}\|_1^2 + \|(I - P_1)e^{(n+\frac{1}{2})}\|_1^2 \\ &\le \epsilon^2\|e^{(n+\frac{1}{2})}\|_1^2 + (d + \epsilon)^2\|e^{(n)}\|_1^2,\end{aligned}$$

where $\epsilon = RC(2\|u\|_1 + M) < 1$ by (3.9). From this, we obtain

$$\|e^{(n+\frac{1}{2})}\|_1 \le \frac{d + \epsilon}{\sqrt{1 - \epsilon^2}}\|e^{(n)}\|_1.$$

Similarly,

$$\|e^{(n+1)}\|_1 \le \frac{d + \epsilon}{\sqrt{1 - \epsilon^2}}\|e^{(n+\frac{1}{2})}\|_1.$$

Hence a sufficient condition for convergence is that $d + \epsilon < \sqrt{1 - \epsilon^2}$ or

$$R < \frac{\sqrt{2 - d^2} - d}{2C(2\|u(R)\|_1 + M)}.$$

This condition is weaker than (3.9).

We now strengthen (3.9) slightly to ensure that the resulting condition does not lead to a vacuous set of problems. It is known ([12], for instance) that if

$$R^2\|f\|_{-1} < C_1,$$

where C_1 is a known constant, then the Navier–Stokes equations have a unique solution u with the bound $\|u\|_1 \le R\|f\|_{-1} < \sqrt{C_1\,\|f\|_{-1}}$. Hence (3.9) can be replaced by the stronger condition

$$R \le \min\left(\frac{1 - d}{2C(M + 2\sqrt{C_1\,\|f\|_{-1}})},\ \sqrt{\frac{C_1}{\|f\|_{-1}}}\right), \tag{3.10}$$

where the right-hand side consists of known quantities. Thus we conclude that if (3.10) is satisfied, the nonlinear Schwarz sequence converges geometrically. □

Strictly speaking, $e^{(\frac{1}{2})}$ is not part of the data of the problem. Instead of the definition of M above, we can define $M = 2\|e^{(0)}\|_1$. Using (3.2), $\|e^{(\frac{1}{2})}\|_1 \le M$ provided that $R^{-1} \ge 4C(u + \|e^{(0)}\|_1)$. An inequality corresponding to (3.9) is

$$R \le \frac{1 - d}{4C(\|e^{(0)}\|_1 + \|u(R)\|_1)}.$$

4. Linear Schwarz sequence. In the previous section, each Schwarz iteration requires the solution of a nonlinear PDE in each subdomain. From a practical point of view, we of course prefer to solve linear problems whenever possible. Now, we demonstrate a version of the Schwarz method where only linear PDEs need to be solved.

Let $u^{(0)} \in V$, and assume it satisfies (3.1). For $n = 0, 1, 2, \ldots$, define the linear Schwarz sequence as

$$\begin{aligned}(u^{(n)} \cdot \nabla)u^{(n+\frac{1}{2})} &= -\nabla p^{(n+\frac{1}{2})} + \frac{1}{R}\triangle u^{(n+\frac{1}{2})} + f \text{ on } \Omega_1,\\ \text{div } u^{(n+\frac{1}{2})} &= 0 \text{ on } \Omega_1,\\ u^{(n+\frac{1}{2})} &= u^{(n)} \text{ on } \partial\Omega_1\end{aligned}$$

and

$$\begin{aligned}(u^{(n+\frac{1}{2})} \cdot \nabla)u^{(n+1)} &= -\nabla p^{(n+1)} + \frac{1}{R}\triangle u^{(n+1)} + f \text{ on } \Omega_2,\\ \text{div } u^{(n+1)} &= 0 \text{ on } \Omega_2,\\ u^{(n+1)} &= u^{(n+\frac{1}{2})} \text{ on } \partial\Omega_2.\end{aligned}$$

Notice that the above equations are linear and that the compatibility conditions are satisfied.

The main result is that this Schwarz sequence converges to the true solution provided that the Reynolds number is sufficiently small.

THEOREM 2. *Assuming that $u^{(0)} \in V$ satisfies* (3.1) *and that R is sufficiently small, the linear Schwarz sequence converges to the true solution u in the norm $\|\cdot\|_1$.*

Proof. The proof is similar to before, and we only give the key steps. We have expressions for the error sequence that are similar to those used before:

$$\begin{aligned}e^{(n+\frac{1}{2})} &= (I - P_1)e^{(n)} + RF_1(e^{(n+\frac{1}{2})}, e^{(n)}),\\ e^{(n+1)} &= (I - P_2)e^{(n+\frac{1}{2})} + RF_2(e^{(n+1)}, e^{(n+\frac{1}{2})}),\end{aligned}$$

where

$$\begin{aligned}F_1(e^{(n+\frac{1}{2})}, e^{(n)}) &= -S_1^{-1}((e^{(n)} \cdot \nabla)e^{(n+\frac{1}{2})} + (u \cdot \nabla)e^{(n+\frac{1}{2})} + (e^{(n)} \cdot \nabla)u),\\ F_2(e^{(n+1)}, e^{(n+\frac{1}{2})}) &= -S_2^{-1}((e^{(n+\frac{1}{2})} \cdot \nabla)e^{(n+1)} + (u \cdot \nabla)e^{(n+1)} + (e^{(n+\frac{1}{2})} \cdot \nabla)u).\end{aligned}$$

Note that we have the estimates

$$\begin{aligned}\|F_1(e^{(n+\frac{1}{2})}, e^{(n)})\|_1 &\le C(\|e^{(n+\frac{1}{2})}\|_1\|e^{(n)}\|_1 + \|u\|_1\|e^{(n+\frac{1}{2})}\|_1 + \|e^{(n)}\|_1),\\ \|F_2(e^{(n+1)}, e^{(n+\frac{1}{2})})\|_1 &\le C(\|e^{(n+1)}\|_1\|e^{(n+\frac{1}{2})}\|_1 + \|u\|_1\|e^{(n+1)}\|_1 + \|e^{(n+\frac{1}{2})}\|_1).\end{aligned}$$

Combining the error sequences, we obtain the error equations

$$\begin{aligned}e^{(n+\frac{1}{2})} &= (I - P_1)(I - P_2)e^{(n-\frac{1}{2})} + R(I - P_1)F_2(e^{(n)}, e^{(n-\frac{1}{2})}) + RF_1(e^{(n+\frac{1}{2})}, e^{(n)}),\\ e^{(n+1)} &= (I - P_2)(I - P_1)e^{(n)} + R(I - P_2)F_1(e^{(n+\frac{1}{2})}, e^{(n)}) + RF_2(e^{(n+1)}, e^{(n+\frac{1}{2})}).\end{aligned}$$

As before, we show that, provided R is sufficiently small, ($R \le (1 - d)/2C(M + \|u\|_1 + 1)$) for every n, $\|e^{(n)}\|_1, \|e^{(n+\frac{1}{2})}\|_1 \le M$, where $M = \max(\|e^{(0)}\|_1, \|e^{(\frac{1}{2})}\|_1)$. From this we obtain

$$\begin{aligned}\|F_1(e^{(n+\frac{1}{2})}, e^{(n)})\|_1 &\le C(M + \|u\|_1)\|e^{(n+\frac{1}{2})}\|_1 + C\|e^{(n)}\|_1,\\ \|F_2(e^{(n+1)}, e^{(n+\frac{1}{2})})\|_1 &\le C(M + \|u\|_1)\|e^{(n+1)}\|_1 + C\|e^{(n+\frac{1}{2})}\|_1.\end{aligned}$$

Applying these inequalities to the error equations, we have

$$\begin{bmatrix} \|e^{(n+1)}\|_1 \\ \|e^{(n+\frac{1}{2})}\|_1 \end{bmatrix} \le A \begin{bmatrix} \|e^{(n+\frac{1}{2})}\|_1 \\ \|e^{(n)}\|_1 \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} \|e^{(n+\frac{1}{2})}\|_1 \\ \|e^{(n)}\|_1 \end{bmatrix} \le A \begin{bmatrix} \|e^{(n)}\|_1 \\ \|e^{(n-\frac{1}{2})}\|_1 \end{bmatrix},$$

where

$$A = \begin{bmatrix} \frac{\epsilon(M+\|u\|_1+1)}{1-\epsilon(M+\|u\|_1)} & \frac{d+\epsilon}{1-\epsilon(M+\|u\|_1)} \\ 1 & 0 \end{bmatrix},$$

where $\epsilon = RC$. For convergence, it is sufficient that the spectral radius of A is less than one:

$$\frac{\epsilon(M+\|u\|_1+1) \pm \sqrt{\epsilon^2(M+\|u\|_1+1)^2 + 4(d+\epsilon)(1-\epsilon(M+\|u\|_1))}}{2(1-\epsilon(M+\|u\|_1))} < 1.$$

This is equivalent to

$$R < \frac{1-d}{2C(M+\|u\|_1+1)}. \tag{4.1}$$

Of course, a variation of (3.10) can be used as an upper bound. ∎

5. Parallel Schwarz sequence. For the two previous Schwarz methods, the iterates must be computed sequentially. In this section, we suggest a Schwarz sequence where the two subdomain problems are independent, and thus they can be solved simultaneously. This sequence also converges provided that the Reynolds number is sufficiently small.

Let $u^{(0)} \in V$, and assume that it satisfies

$$\int_{\Gamma_i} u^{(0)} \cdot \nu = 0, \qquad i = 1, 2. \tag{5.1}$$

Define $u^{(-\frac{1}{2})} = u^{(0)}$. For $n = 0, 1, 2, \ldots$, define the parallel Schwarz sequence as

$$\begin{aligned} (u^{(n)} \cdot \nabla) u^{(n+\frac{1}{2})} &= -\nabla p^{(n+\frac{1}{2})} + \frac{1}{R} \triangle u^{(n+\frac{1}{2})} + f \text{ on } \Omega_1, \\ \text{div } u^{(n+\frac{1}{2})} &= 0 \text{ on } \Omega_1, \\ u^{(n+\frac{1}{2})} &= u^{(n)} \text{ on } \partial\Omega_1 \end{aligned}$$

and

$$\begin{aligned} (u^{(n-\frac{1}{2})} \cdot \nabla) u^{(n+1)} &= -\nabla p^{(n+1)} + \frac{1}{R} \triangle u^{(n+1)} + f \text{ on } \Omega_2, \\ \text{div } u^{(n+1)} &= 0 \text{ on } \Omega_2, \\ u^{(n+1)} &= u^{(n-\frac{1}{2})} \text{ on } \partial\Omega_2. \end{aligned}$$

We define $u^{(n+1)}$ as $u^{(n-\frac{1}{2})}$ on $\overline{\Omega} \setminus \overline{\Omega}_2$. Notice that the above equations are linear and can be solved independently. We again can check that the compatibility conditions are satisfied.

THEOREM 3. *Assuming that $u^{(0)} \in V$ satisfies* (5.1) *and that R is sufficiently small, the parallel Schwarz sequence converges to the true solution u in the norm $\|\cdot\|_1$.*

Proof. The proof is similar to before, and we only give the key steps. We have expressions for the error sequence that are similar to those used before:

$$e^{(n+\frac{1}{2})} = (I - P_1)e^{(n)} + RF_1(e^{(n+\frac{1}{2})}, e^{(n)}),$$
$$e^{(n+1)} = (I - P_2)e^{(n-\frac{1}{2})} + RF_2(e^{(n+1)}, e^{(n-\frac{1}{2})}).$$

Combining the error sequences, we obtain the error equations

$$e^{(n+\frac{1}{2})} = (I - P_1)(I - P_2)e^{(n-\frac{3}{2})} + R(I - P_1)F_2(e^{(n)}, e^{(n-\frac{3}{2})}) + RF_1(e^{(n+\frac{1}{2})}, e^{(n)}),$$
$$e^{(n+1)} = (I - P_2)(I - P_1)e^{(n-1)} + R(I - P_2)F_1(e^{(n-\frac{1}{2})}, e^{(n-1)}) + RF_2(e^{(n+1)}, e^{(n-\frac{1}{2})}),$$

where $e^{(-\frac{3}{2})} \equiv e^{(0)}$.

As before, we show that, provided R is sufficiently small, ($R \le (1-d)/2C(M + \|u\|_1 + 1)$) for every n, $\|e^{(n)}\|_1, \|e^{(n+\frac{1}{2})}\|_1 \le M$, where $M = \max(\|e^{(0)}\|_1, \|e^{(\frac{1}{2})}\|_1)$. Hence we have the estimates

$$\|F_1(e^{(n+\frac{1}{2})}, e^{(n)})\|_1 \le C(M + \|u\|_1)\|e^{(n+\frac{1}{2})}\|_1 + C\|e^{(n)}\|_1,$$
$$\|F_2(e^{(n+1)}, e^{(n-\frac{1}{2})})\|_1 \le C(M + \|u\|_1)\|e^{(n+1)}\|_1 + C\|e^{(n-\frac{1}{2})}\|_1.$$

Applying these inequalities to the error equations, we have

$$\begin{bmatrix} \|e^{(n+1)}\|_1 \\ \|e^{(n+\frac{1}{2})}\|_1 \\ \|e^{(n)}\|_1 \\ \|e^{(n-\frac{1}{2})}\|_1 \end{bmatrix} \le A \begin{bmatrix} \|e^{(n)}\|_1 \\ \|e^{(n-\frac{1}{2})}\|_1 \\ \|e^{(n-1)}\|_1 \\ \|e^{(n-\frac{3}{2})}\|_1 \end{bmatrix},$$

where

$$A = \begin{bmatrix} 0 & \frac{\epsilon(M+\|u\|_1+1)}{1-\epsilon(M+\|u\|_1)} & \frac{d+\epsilon}{1-\epsilon(M+\|u\|_1)} & 0 \\ \frac{\epsilon(M+\|u\|_1+1)}{1-\epsilon(M+\|u\|_1)} & 0 & 0 & \frac{d+\epsilon}{1-\epsilon(M+\|u\|_1)} \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix},$$

and $\epsilon = RC$. For convergence, it is sufficient that the spectral radius of A is less than one. This is equivalent to

$$R < \frac{1-d}{2C(M + \|u\|_1 + 1)}, \tag{5.2}$$

which is the same estimate as that of the linear Schwarz method. □

6. Additive Schwarz sequence. In this section, we demonstrate the convergence of the additive Schwarz sequence, which was originally introduced in [5] for linear elliptic equations and has been used successfully in practice.

Let $u^{(0)} \in V$. For $n = 0, 1, 2, \ldots$, define $d^{(n+\frac{1}{2})} \in V_1$ and $d^{(n+1)} \in V_2$ as the solutions of

$$[d^{(n+\frac{1}{2})}, v_1] = R\int_{\Omega_1} f \cdot v_1 - R\int_{\Omega_1} v_1 \cdot (u^{(n)} \cdot \nabla)u^{(n)} - [u^{(n)}, v_1], \qquad v_1 \in V_1,$$

and of

$$[d^{(n+1)}, v_2] = R\int_{\Omega_2} f \cdot v_2 - R\int_{\Omega_2} v_2 \cdot (u^{(n)} \cdot \nabla)u^{(n)} - [u^{(n)}, v_2], \qquad v_2 \in V_2.$$

The additive Schwarz sequence is defined as

$$u^{(n+1)} = u^{(n)} + \omega(d^{(n+\frac{1}{2})} + d^{(n+1)}),$$

where ω is a relaxation parameter. Roughly speaking, $d^{(n+\frac{1}{2})}$ and $d^{(n+1)}$ are corrections to the iterate $u^{(n)}$ in the subdomains Ω_1 and Ω_2, respectively, and the right-hand sides of the above equations defining the corrections are the residuals of $u^{(n)}$ in the subdomains. Ignoring the nonlinear terms, the above sequence is precisely the additive Schwarz sequence for the Stokes problem. Notice that the above Stokes equations can be solved independently and no additional assumption on $u^{(0)}$ is required.

THEOREM 4. *Suppose $u^{(0)} \in V$. Assuming that $0 < \omega < 1/2$ and that R is sufficiently small, the additive Schwarz sequence converges geometrically to the true solution u in the norm $\|\cdot\|_1$.*

Proof. We sketch only the proof, recording the key equations. From the defining equations, we have

$$d^{(n+\frac{1}{2})} = -P_1 e^{(n)} + RF_1(e^{(n)}),$$
$$d^{(n+1)} = -P_2 e^{(n)} + RF_2(e^{(n)}),$$

from which we obtain

$$e^{(n+1)} = (I - \omega\,(P_1 + P_2))e^{(n)} + \omega R(F_1(e^{(n)}) + F_2(e^{(n)})),$$

where F_1 and F_2 were defined in (3.3) and (3.5). When $0 < \omega < 1/2$, $\|I - \omega\,(P_1 + P_2)\|_1 < 1$. Applying the estimates for F_i and the condition

$$R < \frac{1 - \|I - \omega\,(P_1 + P_2)\|_1}{2\omega C(\|u^{(0)} - u\|_1 + 2\|u\|_1)}, \tag{6.1}$$

we can show that $\|e^{(n)}\|_1 \leq \|e^{(0)}\|_1$ for all n. This condition (or a variation of (3.10)) is also sufficient to guarantee geometric convergence of the additive Schwarz sequence. ∎

7. Discussion and conclusion. In this paper, we show the convergence of four Schwarz alternating methods for the N-dimensional, steady, incompressible Navier–Stokes equations provided that the Reynolds number is sufficiently small and $N \leq 4$. Our results are valid only for the two-subdomain case. It is interesting to see that the bounds (3.9, 4.1, 5.2, 6.1) on the Reynolds number for the four methods are approximately the same. (The constant C appearing in these four estimates can be taken to be the same number.)

The result for finitely many subdomains is again that the Schwarz sequence converges provided that R is sufficiently small. Except for the additive Schwarz scheme, we cannot give an explicit bound for R. The difficulty is in obtaining an explicit expression for the spectral radius of a certain matrix. We elaborate on this point below. To simplify the analysis, assume that any three distinct subdomains have an empty intersection.

First, we need the following lemma which was shown by Lions [9] for the two-subdomain case. The general case is true by induction. We include a proof for completeness.

LEMMA 2. *Suppose $H_0^1(\Omega)^N = H_0^1(\Omega_1)^N + \cdots + H_0^1(\Omega_m)^N$. Let V and V_i be subspaces of $H_0^1(\Omega)^N$ and $H_0^1(\Omega_i)^N$, respectively, consisting of divergence-free functions. Then $V = V_1 + \cdots + V_m$.*

Proof. First consider the case when $m = 2$. Let $u \in V$. We need to show that $u = v_1 + v_2$ for some $v_i \in V_i$, $i = 1, 2$. Since the domains are overlapping, we have $u = u_1 + u_2$ for some $u_i \in H_0^1(\Omega_i)^N$. Let $h = \text{div } u_1$. Let $\Omega_{12} = \Omega_1 \cap \Omega_2$ and $\Omega_{11} = \Omega_1 \cap \overline{\Omega_2}^c$. We claim that $\int_{\Omega_{12}} h = 0$, and so the PDE

$$\text{div } w = h \text{ on } \Omega_{12}$$

has a solution $w \in H_0^1(\Omega_{12})^N$. By extending w by 0 to Ω, we can define $v_1 = u_1 - w$ and $v_2 = u_2 + w$. By construction, div $v_1 = 0$ and div $v_2 = -$ div $u_1 +$ div $w = 0$.

The claim can be shown by the following calculation. Let ν be the unit outward normal to $\partial\Omega_{12}$, and let ν_1 be the unit outward normal to $\partial\Omega_{11}$. Then

$$\int_{\Omega_{12}} h = \int_{\partial\Omega_{12}} u_1 \cdot \nu = \int_{\Gamma_2} u \cdot \nu = -\int_{\partial\Omega_{11}} u \cdot \nu_1 = -\int_{\Omega_{11}} \text{div } u = 0.$$

Now consider the m subdomain case. By induction hypothesis and the fact that the subdomains are overlapping,

$$U \equiv V_1 + \cdots + V_{m-1} = \{v \in H_0^1(\Omega_1 \cup \cdots \cup \Omega_{m-1}), \quad \text{div } v = 0\}.$$

Since $H_0^1(\Omega)^N = H_0^1(\Omega_1 \cup \cdots \cup \Omega_{m-1})^N + H_0^1(\Omega_m)^N$, the result of the two-subdomain case yields the desired conclusion $V = U + V_m$. ∎

For m subdomains, we can obtain an error equation analogous to (3.6). For instance,

$$e^{(n+1)} = (I - P_m) \cdots (I - P_1) e^{(n)} + \cdots,$$

where P_i is the orthogonal projection (with respect to $[\cdot, \cdot]$) of V onto V_i and $[+ \cdots]$ is a sum of m quadratic terms. Armed with Lemma 2, it is well known [9], [1] that

$$d = \|(I - P_m) \cdots (I - P_1)\|_1 < 1$$

for overlapping subdomains. In a similar way as before, we can show that the nonlinear Schwarz sequence is bounded and that

$$\begin{bmatrix} \|e^{(n+\frac{m-1}{m})}\|_1 \\ \|e^{(n+\frac{m-2}{m})}\|_1 \\ \vdots \\ \|e^{(n+\frac{1}{m})}\|_1 \\ \|e^{(n)}\|_1 \end{bmatrix} \le A^{mn} \begin{bmatrix} \|e^{(\frac{m-1}{m})}\|_1 \\ \|e^{(\frac{m-2}{m})}\|_1 \\ \vdots \\ \|e^{(\frac{1}{m})}\|_1 \\ \|e^{(0)}\|_1 \end{bmatrix}, \qquad A = \begin{bmatrix} \frac{\epsilon}{1-\epsilon} & \frac{\epsilon}{1-\epsilon} & \cdots & \frac{\epsilon}{1-\epsilon} & \frac{d}{1-\epsilon} \\ 1 & 0 & \cdots & \cdots & 0 \\ 0 & 1 & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & 1 & 0 \end{bmatrix},$$

where $\epsilon = RC(M + 2\|u\|_1)$ and $e^{(n+\frac{i}{m})}$ is the error of the iterate defined on the ith subdomain. Hence a sufficient condition for convergence is that the spectral radius of A is less than one. The spectral radius is $d^{1/m} + o(\epsilon)$, and thus for all ϵ sufficiently small, the Schwarz sequence converges. Because it does not seem possible to write down an explicit expression for the spectral radius, we cannot give an explicit bound on R.

The additive Schwarz method generalizes directly to the m-subdomain case. It converges if

$$R < \frac{1 - \|I - \omega\,(P_1 + \cdots + P_m)\|_1}{m\omega C(\|u^{(0)} - u\|_1 + 2\|u\|_1)},$$

where $0 < \omega < 1/K$, where K is the minimum number of colors needed to color the subdomains in such a way that overlapping subdomains are assigned different colors.

Nonhomogeneous boundary conditions can also be treated by making a change of variable. Define $w = u - g$, where g is the Dirichlet data on $\partial\Omega$ which is extended to be a divergence-free function in $H^1(\Omega)^N$. Then the Navier–Stokes equations become

$$(w \cdot \nabla)w + (g \cdot \nabla)w + (w \cdot \nabla)g = -\nabla p + \frac{1}{R}\triangle w + G$$

for $w \in V$ and

$$G = \frac{1}{R}\triangle g + f - (g \cdot \nabla)g.$$

The analysis goes through as before. For instance, the nonlinear Schwarz sequence for $w^{(n+\frac{1}{2})}$ is defined by

$$\begin{aligned}(w^{(n+\frac{1}{2})} \cdot \nabla)w^{(n+\frac{1}{2})} + (g \cdot \nabla)w^{(n+\frac{1}{2})} + (w^{(n+\frac{1}{2})} \cdot \nabla)g \\ = -\nabla p^{(n+\frac{1}{2})} + \frac{1}{R}\triangle w^{(n+\frac{1}{2})} + G \text{ on } \Omega_1, \\ \text{div } w^{(n+\frac{1}{2})} = 0 \text{ on } \Omega_1, \\ w^{(n+\frac{1}{2})} = w^{(n)} \text{ on } \partial\Omega_1,\end{aligned}$$

the existence of which is guaranteed provided that the compatibility condition

$$\int_{\Gamma_1} w^{(0)} \cdot \nu = 0$$

is satisfied. In a similar manner as before, we obtain $e^{(n+\frac{1}{2})} = (I - P_1)e^{(n)} + RG_1(e^{(n+\frac{1}{2})})$, where

$$G_1(e^{(n+\frac{1}{2})}) = -S_1^{-1}((g \cdot \nabla)e^{(n+\frac{1}{2})} + (e^{(n+\frac{1}{2})} \cdot \nabla)g) + F_1(e^{(n+\frac{1}{2})})$$

and $e^{(n+\frac{1}{2})} = w^{(n+\frac{1}{2})} - (u - g)$. Hence we have the estimate

$$\|G_1(e^{(n+\frac{1}{2})})\|_1 \le C\,\|e^{(n+\frac{1}{2})}\|_1(2(\|u\|_1 + \|g\|_1) + \|e^{(n+\frac{1}{2})}\|_1).$$

Following the analysis as before, we obtain convergence provided that

$$R < \frac{1 - d}{2C(M + 2(\|u\|_1 + \|g\|_1))}.$$

To prove convergence for a Reynolds number which is not small, we need to get a sharper norm estimate for (3.6) than the simple one given by (3.8) (or a completely different approach than the perturbational one taken here). For example, for the nonlinear Schwarz method, if we estimate (3.6) by (3.8), then, even if we assume a stronger inequality than (3.8) by dropping the quadratic terms

$$\|e^{(n+\frac{1}{2})}\|_1 \le d\|e^{(n-\frac{1}{2})}\|_1 + 2RC\|u\|_1\|e^{(n)}\|_1 + 2RC\|u\|_1\|e^{(n+\frac{1}{2})}\|_1,$$

convergence still requires $R < (1 - d)/4C\|u\|_1$.

Other future work includes extending our results to the time dependent case and the consideration of Schwarz methods on nonoverlapping subdomains.

Acknowledgments. I thank Professor Alfio Quarteroni for an extremely beneficial discussion. I am also grateful to the referees for making many excellent suggestions.

REFERENCES

[1] J.H. Bramble, J. E. Pasciak, J. Wang, and J. Xu, *Convergence estimates for product iterative methods with applications to domain decomposition*, Math. Comp., 57 (1991), pp. 1–21.

[2] X.-C. Cai, W. D. Gropp, D. E. Keyes, R. G. Melvin, and D. P. Young, *Parallel Newton–Krylov–Schwarz algorithms for the transonic full potential equation*, SIAM J. Sci. Comput., 19 (1998), pp. 246–265.

[3] E. J. Dean, Q. V. Dinh, R. Glowinski, J. He, T. W. Pan, and J. Periaux, *Least squares/domain imbedding methods for Neumann problems: Applications to fluid dynamics*, in Proceedings of the 5th International Symposium on Domain Decomposition Methods for Partial Differential Equations, D. E. Keyes, T. F. Chan, G. A. Meurant, J. S. Scroggs, and R. G. Voigt, eds., SIAM, Philadelphia, 1992, pp. 451–475.

[4] M. Dryja and W. Hackbusch, *On the nonlinear domain decomposition method*, BIT, 37 (1997), pp. 296–311.

[5] M. Dryja and O. B. Widlund, *An Additive Variant of the Schwarz Alternating Method for the Case of Many Subregions*, Technical report 339, Courant Institute, New York University, New York, 1987.

[6] L. Fatone, P. Gervasio, and A. Quarteroni, *Multimodels for incompressible flows*, J. Math. Fluid Mech., 2 (2000), pp. 126–150.

[7] R. Glowinski, G. H. Golub, G. A. Meurant, and J. Periaux, Proceedings of the 1st International Symposium on Domain Decomposition Methods for Partial Differential Equations, SIAM, Philadelphia, 1988.

[8] D. K. Kaushik, D. E. Keyes, and B. F. Smith, *On the interaction of architecture and algorithm in the domain-based parallelization of an unstructured-grid incompressible flow code*, in Domain Decomposition Methods 10, J. Mandel, C. Farhat, and X.-C. Cai, eds., AMS, Providence, RI, 1998, pp. 287–295.

[9] P. L. Lions, *On the Schwarz alternating method* I, in Proceedings of the 1st International Symposium on Domain Decomposition Methods for Partial Differential Equations, R. Glowinski, G. H. Golub, G. A. Meurant, and J. Periaux, eds., SIAM, Philadelphia, 1988, pp. 1–42.

[10] S. H. Lui, *On Schwarz alternating methods for nonlinear elliptic PDEs*, SIAM J. Sci. Comput., 21 (2000), pp. 1506–1523.

[11] X.-C. Tai and M. Espedal, *Rate of convergence of some space decomposition methods for linear and nonlinear problems*, SIAM J. Numer. Anal., 35 (1998), pp. 1558–1570.

[12] R. Temam, *Navier–Stokes Equations*, North Holland, Amsterdam, 1984.

SIAM J. SCI. COMPUT.
Vol. 22, No. 6, pp. 1987–2005
© 2001 Society for Industrial and Applied Mathematics

ALGEBRAIC TWO-LEVEL PRECONDITIONERS FOR THE SCHUR COMPLEMENT METHOD*

L. M. CARVALHO†, L. GIRAUD‡, AND P. LE TALLEC§

Abstract. The solution of elliptic problems is challenging on parallel distributed memory computers since their Green's functions are global. To address this issue, we present a set of preconditioners for the Schur complement domain decomposition method. They implement a global coupling mechanism, through coarse-space components, similar to the one proposed in [Bramble, Pasciak, and Shatz, *Math. Comp.*, 47 (1986), pp. 103–134]. The definition of the coarse-space components is algebraic; they are defined using the mesh partitioning information and simple interpolation operators. These preconditioners are implemented on distributed memory computers without introducing any new global synchronization in the preconditioned conjugate gradient iteration. The numerical and parallel scalability of those preconditioners are illustrated on two-dimensional model examples that have anisotropy and/or discontinuity phenomena.

Key words. domain decomposition, two-level preconditioning, Schur complement, parallel distributed computing, elliptic partial differential equations

AMS subject classifications. 65N55, 65F10, 65F50, 65Y05

PII. S1064827598340809

1. Introduction.

In the recent years, there has been an important development of domain decomposition algorithms for solving numerically partial differential equations. Elliptic problems are challenging since their Green's functions are global: the solution at each point depends upon the data at all other points. Nowadays some methods possess optimal convergence rates for given classes of elliptic problems. It can be shown that the condition number of the associated preconditioned systems is independent of the number of subdomains and is either independent of or logarithmically dependent on the size of the subdomains. That optimality and these quasi-optimality properties are often achieved thanks to the solution of a coarse problem defined on the whole physical domain. Through the use of coarse spaces, this approach captures the global behavior of the elliptic equations.

Various domain decomposition techniques, from the eighties and nineties, have suggested different global coupling mechanisms and various combinations between them and the local preconditioners. In the framework of nonoverlapping domain decomposition techniques, we refer, for instance, to BPS (Bramble, Pasciak, and Schatz) [2], vertex space [6, 19], and some extended balancing Neumann–Neumann [13, 14, 15], as well as FETI (finite element tearing and interconnection) [8, 16], for the presentation of major two-level preconditioners.

Even though the theory for domain decomposition techniques is well developed (see [4, 18, 20] and the references therein) for regular enough elliptic two-dimensional and three-dimensional equations, there are situations, even in two dimensions, that either are not covered by the theory or have constants appearing in the theoretical

*Received by the editors June 23, 1998; accepted for publication (in revised form) October 13, 2000; published electronically March 20, 2001.
http://www.siam.org/journals/sisc/22-6/34080.html
†UERJ, Instituto de Matematica e Estatistica, Rua São Francisco de Xavier, 524, Sexto Andar, Rio de Janeiro, RJ, Brazil (luizmc@ime.uerj.br).
‡CERFACS, 42 av. Gaspard Coriolis, 31057 Toulouse Cedex, France (giraud@cerfacs.fr).
§Ecole Polytechnique, 91128 Palaiseau Cedex, France (patrick.letallec@polytechnique.fr).

bounds for the condition numbers which can be very large. Such situations might occur, for instance, in the solution of scalar heterogeneous, anisotropic elliptic problems. In that respect we consider in this paper two-level domain decomposition methods based on algebraic constructions of the coarse space for the solution of heterogeneous, anisotropic two-dimensional elliptic problems defined on structured or unstructured discretizations. Such numerical difficulties arise in the equations involved in semiconductor device simulation that was actually one of the main motivations for this work, as illustrated in one of our numerical examples. Through an experimental study of the numerical and parallel scalability of the preconditioned Schur complement method, we investigate some new coarse-space preconditioners. They are closely related to BPS [2], although we propose different coarse spaces to construct their coarse components. Furthermore, we propose a parallel implementation of the preconditioned Schur complement method that does not require any new global synchronization in the iteration loop of the preconditioned conjugate gradient solver.

The paper is organized as follows. In section 2, we formulate the problem and introduce the notation. In section 3, we describe the various coarse spaces we have considered when defining the preconditioners' coarse-space components. Section 4 provides some details on the parallel distributed implementation of the resulting domain decomposition algorithm. Computational results illustrating the numerical and parallel scalability of the preconditioners are given in section 5.

2. Preliminaries and notation. The purpose of this section is two-fold. First, we formulate a two-dimensional elliptic model problem. Then we introduce the notation that will allow us to define the coarse spaces we have considered in our preconditioners.

We consider the following 2nd order self-adjoint elliptic problem on an open polygonal domain Ω included in $\mathbb{R}^2$:

$$\left\{ \begin{array}{rcll} -\frac{\partial}{\partial x}(a(x,y)\frac{\partial v}{\partial x}) - \frac{\partial}{\partial y}(b(x,y)\frac{\partial v}{\partial y}) & = & F(x,y) & \text{in} \quad \Omega, \\ v & = & 0 & \text{on} \quad \partial\Omega, \end{array} \right. \tag{1}$$

where $a(x,y)$, $b(x,y) \in \mathbb{R}^2$ are bounded positive functions on Ω. We assume that the domain Ω is partitioned into N nonoverlapping subdomains $\Omega_1, \ldots, \Omega_N$ with boundaries $\partial\Omega_1, \ldots, \partial\Omega_N$. We discretize (1) either by finite differences or finite elements resulting in a symmetric and positive definite linear system with sparse possibly unstructured matrix

$$Au = f.$$

Let B be the set of all the indices of the discretized points which belong to the interfaces between the subdomains. Grouping the points corresponding to B in the vector u_B and the ones corresponding to the interior I of the subdomains in u_I, we get the reordered problem

$$\begin{pmatrix} A_{II} & A_{IB} \\ A_{IB}^T & A_{BB} \end{pmatrix} \begin{pmatrix} u_I \\ u_B \end{pmatrix} = \begin{pmatrix} f_I \\ f_B \end{pmatrix}. \tag{2}$$

Eliminating u_I from the second block row of (2) leads to the following reduced equation for u_B:

$$Su_B = f_B - A_{IB}^T A_{II}^{-1} f_I, \quad \text{where } S = A_{BB} - A_{IB}^T A_{II}^{-1} A_{IB} \tag{3}$$

is the Schur complement of the matrix A_{II} in A and is usually referred to as the Schur complement matrix.

To describe the preconditioners, we need to define a partition of B. Let V_j be the singleton sets that contain one index related to one cross point and let $V = \cup_j V_j$ be the set with all those indices; each cross point is represented by $\times$ in Figure 1.

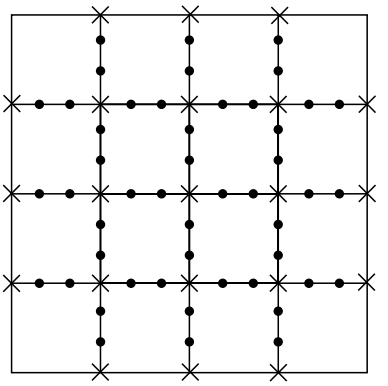


FIG. 1. *A* 4×4 *box decomposition with edge* ($\bullet$) *and vertex* ($\times$) *points.*

If $j \neq l$, $(j,l) \in \{1,2,\ldots,N\}^2$ and j and l are such that Ω_j and Ω_l are neighboring subdomains (i.e., $\partial\Omega_j$ and $\partial\Omega_l$ share at least one edge of the mesh), then we can define each edge E_i by

$$E_i = (\partial\Omega_j \cap \partial\Omega_l) - V.$$

In Figure 1, the points belonging to the m edges $(E_i)_{i=1,m}$ are represented by $\bullet$.

We can thus describe the set B as

$$B = \left(\bigcup_{i=1}^{m} E_i \right) \cup V, \tag{4}$$

which is a partition of the interface B into m edges E_i and the set V.

Here, we remark that we mix continuous curves, $\partial\Omega_i$, with sets of indices. This ambiguity can be disregarded if we consider that, in order to minimize notation, the symbols Ω_i and $\partial\Omega_i$ represent either continuous sets or discrete sets of indices associated with the grid points. In addition we will refer to as the support of a sparse vector the set of indices of its nonzero entries.

3. The preconditioners. The classical BPS preconditioner [2] can be briefly described as follows. We first define a series of projection and interpolation operators. Specifically, for each E_i we define $R_i = R_{E_i}$ as the standard pointwise restriction of nodal values on E_i. Its transpose extends grid functions in E_i by zero on the rest of the interface B. Similarly we define R_V the canonical restriction on V. Thus, we set $S_{ij} \equiv R_i S R_j^T$ and $S_V \equiv R_V S R_V^T$. Additionally, let's assume that $\Omega_1, \ldots, \Omega_N$ form the elements of a coarse grid mesh, τ^H, with mesh size H. We then define grid transfer operators between the interface and the coarse grid. R^T is an interpolation operator which corresponds to using linear interpolation between two adjacent cross points V_j, V_k (i.e., adjacent points in τ^H connected by an edge E_i) to define values on

the edge E_i. Finally, A_H is the Galerkin coarse grid operator $A_H = RAR^T$ defined on τ^H.

With these notations a very close variant of the BPS preconditioner is defined by

$$M_{BPS} = \sum_{E_i} R_i^T S_{ii}^{-1} R_i + R^T A_H^{-1} R, \tag{5}$$

as described, for instance, in this algebraic form in [4]. It can be interpreted as a generalized block Jacobi preconditioner for the Schur complement system (3) where the block diagonal preconditioning for S_V is omitted and a residual correction is used on a coarse grid. The coarse grid term $R^T A_H^{-1} R$ allows us to incorporate a global coupling between the interfaces.

This global coupling is critical for scalability. In particular, it has been shown in [2] that, when applying the original BPS technique to a uniformly elliptic operator, the preconditioned system has a condition number

$$\kappa(M_{BPS} S) = \mathcal{O}(1 + \log^2(H/h)),$$

where h is the mesh size. This implies that the condition number depends only weakly on the mesh spacing and on the number of processors. Therefore, such a preconditioner is appropriate for large systems of equations on large processor systems.

Similarly to BPS, we consider a class of preconditioners described in a generic way as

$$M = M_{local} + M_{global},$$

where

- M_{local} is a block diagonal preconditioner with one block for each edge E_i;
- M_{global} is also computed using a Galerkin formula, but involving S instead of A and using different coarse spaces and restriction operators.

In the rest of this paper, we will define various preconditioners that differ only in the definition of M_{global}. Our goal is to obtain performance similar to those of the original BPS preconditioner even in the presence of anisotropy or heterogeneity, with a simple algebraic structure and a parallel implementation strategy. In particular, for practical implementation purposes within a general purpose computer code, we do not want to refer explicitly to an underlying coarse grid, or to underlying basis functions, since these notions are always hard to identify in practice when using general grids, finite elements, or mixed finite elements.

Let U be the algebraic space of nodal vectors where the Schur complement matrix is defined, and let U_0 be a q-dimensional subspace of U. Elements of U_0 are characterized by the set of nodal values that they can achieve. This subspace will be called coarse space.

Let $R_0 : U \to U_0$ be a restriction operator which maps full vectors of U into vectors in U_0, and let $R_0^T : U_0 \to U$ be the transpose of R_0, an extension operator which extends vectors from the coarse space U_0 to full vectors in the fine space U.

The Galerkin coarse-space operator

$$A_0 = R_0 S R_0^T, \tag{6}$$

in some way, represents the Schur complement on the coarse space U_0.

The global coupling mechanism is introduced by the coarse component of the preconditioner which can thus be defined as $M_{global} = R_0^T A_0^{-1} R_0$.

The following lemma ensures the correctness of the operators with which we work.

LEMMA 1. *If the operator R_0^T is of full rank and if S is nonsingular, symmetric, and positive definite, then the matrix A_0, defined in* (6)*, is nonsingular, symmetric, and positive definite.*

The coarse-space preconditioners will differ only in the choice of the coarse space U_0 and the interpolation operator R_0^T. For convergence reasons, and similarly to the Neumann–Neumann and balancing Neumann–Neumann preconditioner [13, 14], R_0^T must be a partition of the unity in U in the sense that

$$R_0^T \mathbf{1} = \mathbf{1}, \tag{7}$$

where the symbol $\mathbf{1}$ denotes the vectors of all 1's that have different size in the right- and left-hand sides of (7).

With these notations and definitions, all the preconditioners presented in what follows can be written as

$$M = \sum_{E_i} R_i^T \tilde{S}_{ii}^{-1} R_i + R_0^T A_0^{-1} R_0, \tag{8}$$

where we usually replace S_{ii} in (5) by an approximation $\tilde{S}_{ii}$ computed using a probing technique [5, 11].

In the next section, we define various coarse spaces and restriction operators which can be used in a very general framework. The support of the basis vectors Z_k has inspired the name of the coarse spaces. Although a large number of different preconditioners can then be proposed, we restrict our study to five combinations of coarse spaces and restriction operators.

3.1. Vertex-based coarse space. The first coarse space we consider is similar to the BPS one. Each degree of freedom of U_0 is associated with one vertex V_j, and the basis vectors generating the nodal values of the elements of U_0 can be defined as follows. Let $V_k \subset V$ be a singleton set that contains the index associated with a cross point and $(E_j)_{(j \in J_k)}$ be the adjacent edges to V_k. Let m_c denote the number of vertex points; then

$$\widetilde{\mathcal{I}}_k = \bigcup_{(j \in J_k)} E_j \cup V_k$$

is the set of indices we associate with the cross point V_k to define the support of the basis vectors.

Let Z_k be a vector defined on B and $Z_k(i)$ its ith component. Then the vertex-based coarse space U_0 can be defined as

$$U_0 = \text{span}[Z_k : k = 1, \ldots, m_c], \text{ where } \quad Z_k(i) = \begin{cases} 1 & \text{if } i \in \widetilde{\mathcal{I}}_k, \\ 0 & \text{elsewhere.} \end{cases}$$

The set $\widetilde{\mathcal{I}}_k$ associated with a cross point V_k is depicted in Figure 2. The set of vectors $\mathcal{B} = \{Z_1, Z_2, \ldots, Z_{m_c}\}$ forms a basis for the subspace U_0, as these vectors span U_0 by construction and they are linearly independent.

For this coarse space, we consider three different restriction operators R_0 and their associated prolongation operator R_0^T.

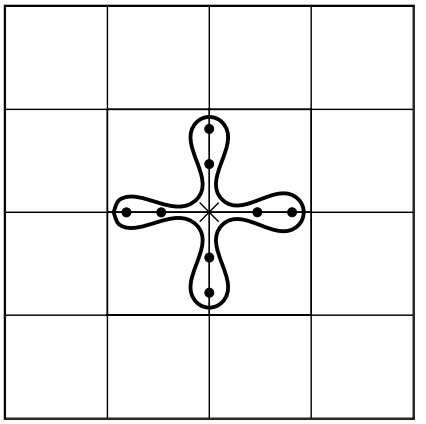


FIG. 2. *Support of one basis vector of the "vertex" coarse space.*

3.1.1. Flat restriction operator. This operator returns for each set $\widetilde{\mathcal{I}}_k$ the weighted sum of values of all the nodes in $\widetilde{\mathcal{I}}_k$. The weights are determined by the inverse of the number of sets $\widetilde{\mathcal{I}}_k,\ k \in \{1, 2, \ldots, m_c\}$, that a given node belongs to. For two-dimensional problems, the weight for the cross points is 1 and for an edge point is 1/2.

3.1.2. Linear interpolation operator. The interpolation operators in this section and the next are basically one-dimensional interpolations on the edges E_i of values defined at the cross points that are its endpoints. In this respect, let us describe them in the one-dimensional framework obtained by mapping the edge E_i on the interval (0,1) through the map $\phi(M_j) = \dfrac{\sum k \leq j |M_{k-1}M_k|}{\sum k \in E_i |M_{k-1}M_k|}$. We consider the following one-dimensional model problem:

$$\left\{ \begin{array}{rcll} -\dfrac{d}{dx}\left(a(x)\dfrac{d}{dx}u(x)\right) & = & f & \text{in (0.1)}, \\ u(x) & = & 0 & \text{at } x = 0 \text{ and } 1. \end{array} \right. \tag{9}$$

Let $H^1(0,1)$ be the standard Sobolev space on the interval (0,1) and $H_0^1(0,1)$ its subspace whose functions vanish at $x = 0$ and $x = 1$. Given a grid $x_j^h = jh, j = 0, \ldots, n$, on $(0,1)$ as the image of the original discretization of E_i, define the fine grid linear finite element space to be

$$V^h = \{v^h \in H_0^1(0,1) : v^h \text{ is linear on } [x_j^h, x_{j+1}^h], j = 0, \ldots, n-1\}$$

and denote the set of nodal basis by $\{\phi_j^h\}_{j=0}^n$.

Let $(x_i^H)_{i=1,m}$ be the set of coarse grid points defined by the vertices of the partitioning of $(0,1)$ into nonoverlapping subdomains. Now, we define the coarse subspace $V^H = span\{\phi_i^H : i = 1, \ldots, m\}$, where ϕ_i^H are the coarse grid nodal basis functions.

Since $\{\phi_i^H\}$ is a basis of V^H, which is a subspace of V^h, there exists a unique matrix R_0^T of size $n-1 \times m$ such that

$$[\phi_1^H \cdots \phi_m^H] = [\phi_1^h \cdots \phi_{n-1}^h] R_0^T,$$

which is usually referred to as the local interpolation matrix. We depict in Figure 3 an example of a coarse grid basis function that defines the linear interpolation.

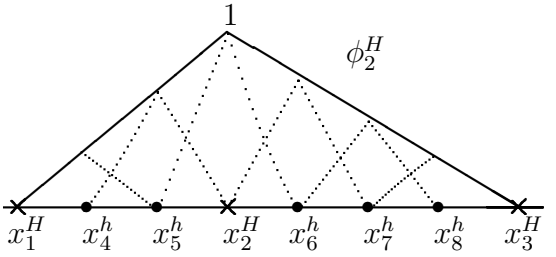


FIG. 3. *Coarse grid basis function ϕ_2^H associated with the linear interpolation.*

When we use block Jacobi as the local preconditioner combined with the vertex coarse space using this linear prolongation operator, the resulting preconditioner is equivalent to the one proposed in [2].

Let A be the matrix defined in (2). Let A_V be the Galerkin coarse grid operator associated with A defined by

$$A_V = \widetilde{R}_0 A \widetilde{R}_0^T, \tag{10}$$

where $\widetilde{R}_0$ is a restriction operator from Ω to the coarse space U_0. It has been shown [2, 20] that, in general, for elliptic problems the operator A_V is spectrally equivalent to

$$R_0 S R_0^T, \tag{11}$$

and for a few cases these coarse operators are even equal.

If we have used the approach defined by (10), the construction of the coarse space would have been reduced to some matrix-vector multiplications with A and the factorization of A_V. Nevertheless, we deal with problems for which only the spectral equivalence between (10) and (11) is ensured. For this reason, we have preferred to use the Galerkin coarse grid correction with the Schur complement matrix as described in (11) rather than the one proposed in a similar matrix form in [4] that used A_V defined by (10).

3.1.3. Operator-dependent restriction operator. The origin of the operator-dependent restriction is the operator-dependent transfer operator proposed in the framework of multigrid methods; see [23] and references therein. In [10], the authors proposed an extension of these operators for two-dimensional nonoverlapping domain decomposition methods. The general purpose of these operators is to construct from u defined on V an interpolation $\tilde{u}$ defined on B that is piecewise linear so that $a\frac{\partial \tilde{u}}{\partial x}$ and $b\frac{\partial \tilde{u}}{\partial y}$ are continuous even when either a or b in (1) are discontinuous along an edge E_i.

Similarly to the linear interpolation, we can define the operator-dependent interpolation in one-dimension through the definition of the coarse grid basis functions ϕ_i^H. In that case those basis functions are constructed by solving the following local problem in $[x_{i-1}^H, x_i^H]$:

$$\begin{cases} -\dfrac{d}{dx}(a(x)\dfrac{d}{dx}\phi_i^H) = 0 \quad \text{in } [x_{i-1}^H, x_i^H], \\ \phi_i^H(x_{i-1}^H) = 0, \phi_i^H(x_i^H) = 1. \end{cases} \tag{12}$$

In Figure 4, we depict the basis function ϕ_2^H when the function $a(x)$ is piecewise constant with some discontinuities at x_5^h and x_7^h.

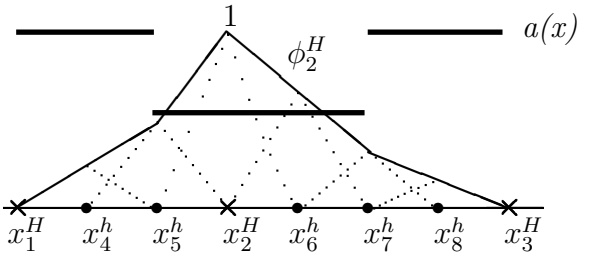


FIG. 4. *Coarse grid basis function ϕ_2^H associated with the operator-dependent interpolation.*

We omit the computational details and notice only that such operators

- can be constructed by solving one tridiagonal linear system for each E_i, which corresponds to the solution of (12). The size of the tridiagonal matrices is the number of nodes on E_i;
- reduce to a linear interpolation when $a \equiv 1$ and $b \equiv 1$;
- in one dimension reduce to the multigrid energy minimization approach [21, 22] or to the multigrid operator-dependent interpolation with harmonic averaging [10], when every other point is a coarse point (i.e., each subdomain contains only one point).

3.2. Subdomain-based coarse space. With this coarse space, we associate one degree of freedom with each subdomain. Let B be as defined in (4). Let Ω_k be a subdomain and $\partial\Omega_k$ its boundary. Then

$$\overline{\mathcal{I}}_k = \partial\Omega_k \cap B$$

is the set of indices we associate with the domain Ω_k. Figure 5 shows the elements of a certain set $\overline{\mathcal{I}}_k$.

Let Z_k be a vector defined on B and let $Z_k(i)$ be its ith component. Then the subdomain-based coarse space U_0 can be defined as

$$U_0 = \text{span}[Z_k : k = 1, \ldots, N], \text{ where } \quad Z_k(i) = \begin{cases} 1 & \text{if } i \in \overline{\mathcal{I}}_k, \\ 0 & \text{otherwise.} \end{cases}$$

Notice that for the example depicted in Figure 5, $[Z_k]$ is rank deficient. Indeed, if we consider $\tilde{v} = \sum_{i=1}^{N} \alpha_i Z_i$, where the α_i are in a checkerboard pattern, equal to -1 and $+1$, it is easy to see that $\tilde{v} = 0$.

Nevertheless, this rank deficiency can be easily removed by discarding one of the vectors of $[Z_k]$. In this particular situation, the set of vectors $\mathcal{B} = \{Z_1, Z_2, \ldots, Z_{N-1}\}$ forms a basis for the subspace U_0.

The considered restriction operator R_0 returns for each subdomain $(\Omega_i)_{i=1,N-1}$ the weighted sum of the values at all the nodes on the boundary of that subdomain. The weights are determined by the inverse of the number of subdomains in $(\Omega_i)_{i=1,N-1}$ each node belongs to. For all the nodes but the ones on $\partial\Omega_N$ (in our particular example) this weight is $1/2$ for the points on an edge and $1/4$ for the cross points. These weights can be replaced as in [13] by operator dependent weights $R_0(i,k) = a_i/(a_i + a_j)$ on the edge separating Ω_i from Ω_j, but this choice has not been tested numerically in the present work.

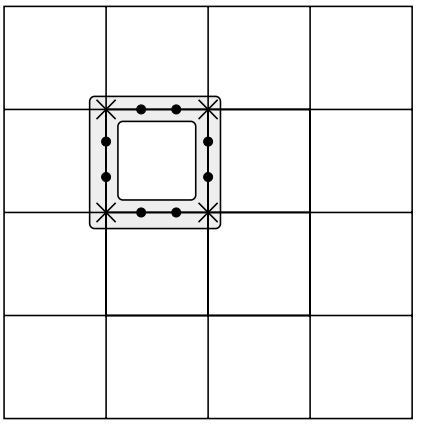


FIG. 5. *Support of one basis vector of the "subdomain" coarse space.*

Remark 1. Although used in a completely different context, this coarse space is similar to the one used in the balancing Neumann–Neumann preconditioner for Poisson-type problems [14]. We use one basis vector for each subdomain, whereas in balancing Neumann–Neumann the basis vectors are defined only for interior subdomains for solving the Dirichlet problem (1), which are the subdomains where the local Neumann problems are singular.

3.3. Edge-based coarse space. We refine the coarse space based on the subdomains, and we introduce one degree of freedom per interface between two neighboring subdomains, that is, when $\partial\Omega_i$ and $\partial\Omega_j$ share at least one edge of the mesh.

Let E_k be an edge and V_j and V_l its adjacent cross points; then

$$\hat{\mathcal{I}}_k = E_k \cup V_j \cup V_l$$

is the set of indices we associate with the edge E_k.

Let Z_k be defined on B, and let $Z_k(i)$ be its ith component. Let m_e denote the number of edges $E_i \subset B$; then the edge-based coarse space U_0 can be defined as

$$U_0 = \text{span}[Z_k : k = 1, \ldots, m_e], \qquad \text{where} \quad Z_k(i) = \begin{cases} 1 & i \in \hat{\mathcal{I}}_k, \\ 0 & \text{otherwise.} \end{cases}$$

The set $\hat{\mathcal{I}}_k$ associated with an element of the coarse space U_0 is depicted in Figure 6. The set of vectors $\mathcal{B} = \{Z_1, Z_2, \ldots, Z_{m_e}\}$ forms a basis for the subspace U_0; as before, these vectors span U_0 by construction and are linearly independent.

The considered restriction operator R_0 returns for each edge the weighted sum of the values at all the nodes on that edge. The weights are determined by the inverse of the number of edges each node belongs to. For the decomposition depicted in Figure 1, the weights for the restriction operator are 1 for points belonging to E_k and 1/4 for those belonging to V_l and V_j.

4. Parallel implementation. The independent solutions of local PDE problems expressed by the domain decomposition techniques are particularly suitable for parallel distributed computation. In a parallel distributed memory environment each

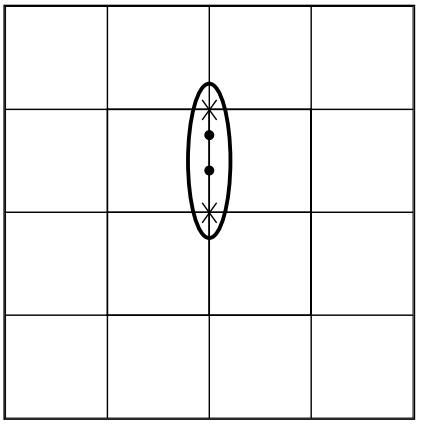


FIG. 6. *Support of one basis vector of the "edge" coarse space.*

subdomain can be assigned to a different processor. With this mapping, all but two of the basic linear algebra operations in the preconditioned conjugate gradient can be implemented either without communication or with neighbor-to-neighbor communication only. The only two steps that require global communication are the dot product computation and the solution of the coarse problem performed at each iteration. In what follows, we will describe how the construction of the coarse components of our preconditioners can be performed in parallel with only one reduction. We will also show how the coarse problem solution can be implemented without any extra global communication within the iteration loop.

4.1. Construction of the preconditioner. The linear systems associated with the coarse spaces described above are much smaller than the linear systems associated with the local Dirichlet problems, which have to be solved when computing the matrix-vector product by S. In this respect, we construct the coarse matrix A_0 once and assemble it on all the processors so that we can redundantly perform in parallel its solution at each preconditioned conjugate gradient iteration. Furthermore, we can take advantage of the structure of S and R_0 to construct A_0 in parallel. Without any communication each processor can compute the contribution of its subdomain to the entries of A_0 via matrix-vector and scalar products that involve only its local Schur complement and the vectors whose support intercepts the boundary of its subdomain. At this stage, all the processors have some nonassembled entries of A_0, and a global sum reduction (MPI_Allreduce) enables them to assemble A_0 on all the processors that can then factorize it. As the Schur complement matrix is not assembled, the most expensive part of the construction is the matrix-vector product with the local Schur complement that requires the solution of the Dirichlet problems. For each processor, the number of solutions is equal to the number of basis vector supports that intercept the boundary of the subdomain the processor is in charge of. For a box decomposition of a uniform finite elements or finite differences mesh, the number of Dirichlet problem solutions to be performed by an internal subdomain is

- four for the vertex-based coarse component,
- eight for the subdomain based coarse component (that can reduce to four for a five-point finite difference scheme as the row in A associated to the cross

points is unchanged in S),
- four for the edge-based coarse component.

4.2. Application of the preconditioner. Having made the choice of a redundant solution of the coarse component on each processor, we can further exploit this formulation to avoid introducing any new global synchronization in the preconditioned conjugate gradient iterations described below.

$$
\begin{aligned}
&x^{(0)} = 0,\ r^{(0)} = b\\
&\textbf{repeat}\\
&\quad z^{(k-1)} = Mr^{(k-1)} \qquad (13)\\
&\quad \textbf{if } k = 1 \textbf{ then}\\
&\quad\quad p^{(1)} = z^{(0)}\\
&\quad \textbf{else}\\
&\quad\quad \beta^{(k-1)} = \boxed{z^{(k-1)^T} r^{(k-1)}} / z^{(k-2)^T} r^{(k-2)}, \qquad (14)\\
&\quad\quad p^{(k)} = z^{(k-1)} + \beta^{(k-1)} p^{(k-1)} \qquad (15)\\
&\quad \textbf{endif}\\
&\quad q^{(k)} = Sp^{(k)},\\
&\quad \alpha^{(k)} = z^{(k-1)^T} r^{(k-1)} / \boxed{p^{(k)^T} q^{(k)}},\\
&\quad x^{(k)} = x^{(k-1)} + \alpha^{(k)} p^{(k)},\\
&\quad r^{(k)} = r^{(k-1)} - \alpha^{(k)} q^{(k)}\\
&\textbf{until convergence}
\end{aligned}
$$

The steps involving a potential global synchronization are boxed, while the calculation of Mr and Sp involve only neighbor-to-neighbor communication.

If we now unroll (13) in the preconditioned conjugate gradient algorithm using the general definition of the preconditioner, we have

$$
\begin{aligned}
z_k &= \left(\sum_{E_i} R_i^T \tilde{S}_{ii}^{-1} R_i + R_0^T A_0^{-1} R_0 \right) r_k\\
&= \sum_{E_i} (R_i^T \tilde{S}_{ii}^{-1} R_i r_k) + R_0^T A_0^{-1} R_0 r_k. \qquad (16)
\end{aligned}
$$

Each term of the summation in (16) is computed by one processor with possible one neighbor-to-neighbor communication. Furthermore, the numerator of (14) can also be rewritten as

$$
\begin{aligned}
(r_k, z_k) &= (r_k, Mr_k)\\
&= \left(r_k, \left(\sum_{E_i} R_i^T \tilde{S}_{ii}^{-1} R_i + R_0^T A_0^{-1} R_0 \right) r_k \right)\\
&= \sum_{E_i} (R_i r_k, \tilde{S}_{ii}^{-1} R_i r_k) + (R_0 r_k, A_0^{-1} R_0 r_k). \qquad (17)
\end{aligned}
$$

The right-hand side of (17) has two parts. The first is naturally local because it is related to the diagonal block preconditioner. The second, with the presented formulation, is global but does not require any new global reduction. $R_0 r_k$ is actually

composed of entries that are calculated in each subdomain ("interface" coarse space) or group of neighboring subdomains ("vertex" and "domain" coarse spaces). After being locally computed, the $R_0 r_k$ entries are gathered on all the processors thanks to the reduction used to assemble each local partial dot product $(R_i r_k, \tilde{S}_{ii}^{-1} R_i r_k)$. At this stage the solution $A_0^{-1} R_0 r_k$ can be performed redundantly on each processor, and β in (14) can be computed by each processor. Rewriting these steps in the iteration loop allows us to introduce the coarse component without any extra global synchronization. With this approach, we avoid a well-known bottleneck when using Krylov methods on parallel distributed memory computers.

5. Numerical experiments. We consider the solution of (1) discretized by a five-point central difference scheme on a uniform mesh using a preconditioned Schur complement method. We illustrate the numerical scalability of the proposed preconditioners on academic two-dimensional model test cases that have both anisotropy and discontinuity.

For all the experiments, the convergence is attained when the 2-norm of the residual normalized by the 2-norm of the right-hand side is less than 10^{-5}. All the computations are performed using 64-bit arithmetic. The $\tilde{S}_{ii}$ approximations in (8) are tridiagonal matrices.

5.1. Model problems. For the numerical experiments, we consider model problems that have both discontinuous and anisotropic phenomena. These difficulties arise in the equations involved in semiconductor device simulation, which was actually one of the main motivations for this study.

Figure 7 represents a unit square divided into six regions with piecewise constant functions g_j, $j = 1$ or 3. We consider the problems as having low intensity if $j = 1$ and high intensity if $j = 3$. Let a and b be the functions of the elliptic problem as described in (1). Using this notation, we can define different problems with different degrees of difficulty. In the following description, the acronyms in capital letters are used to refer to the problems in Tables 2 and 3:

- high discontinuity (HD): $a = b = g_3$,
- low discontinuity (LD): $a = b = g_1$,
- high anisotropy (HA): $a = 10^3$ and $b = 1$,
- low anisotropy (LA): $a = 10$ and $b = 1$,
- high discontinuity and high anisotropy (HDA): $a = g_3$ and $b = 1$,
- low discontinuity and low anisotropy (LDA): $a = g_1$ and $b = 1$.

The preconditioners with the coarse components are denoted in the tables:

sd: subdomain defined in section 3.2,
vl: vertex-linear defined in section 3.1 with the linear interpolation,
vo: vertex-operator-dependent defined in section 3.1.3 with the operator-dependent interpolation,
vf: vertex-flat defined in section 3.1 with the flat interpolation,
ed: edge defined in section 3.3,
no: without coarse space in this case we only use the local preconditioner (block diagonal).

5.2. Numerical behavior. We study the numerical scalability of the preconditioners by investigating the dependence of the convergence on the number and on the size of the subdomains. First, in Table 1, we give the figures for the five coarse spaces when solving the Poisson's problem. As we could expect, the behavior of all coarse-space options does not depend on the number of subdomains. There is only a

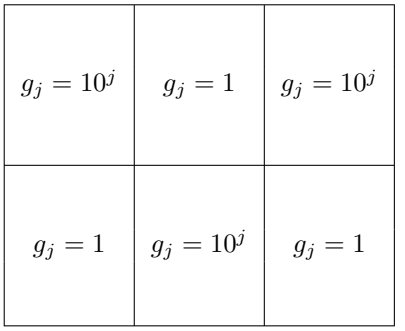


FIG. 7. *Definition of two discontinuous functions on* Ω*, the unit square of* $\mathbb{R}^2$.

TABLE 1
The number of PCG iterations varying the number of subdomains with 16×16 *points per subdomain for Poisson's problem.*

# subdomains	16	64	256	1024
no	15	28	48	90
sd	15	19	19	18
vf	15	18	18	18
vl	10	10	10	10
vo	10	10	10	10
ed	15	18	18	18

small growth in the number of iterations for subdomain, vertex-flat and edge, when the number of subdomains goes from 16 to 64.

5.2.1. Dependency of the coarse preconditioners on H. In Table 2, we report the number of preconditioned conjugate gradient iterations for each model problem. For these tests, we vary the number of subdomains while keeping constant their sizes (i.e., H variable with $\frac{H}{h}$ constant). In this table each subdomain is a 16×16 grid and the number of subdomains goes from 16 up to 1024 using a box decomposition; that is 4×4 decomposition up to 32×32 decomposition.

In the first row, we see the growth of the number of iterations of a block Jacobi preconditioner without using any coarse grid correction. Its numerical behavior is well known and is governed by the following theoretical bound for its condition number (see, for instance, [4]):

$$\text{cond}(M_{\text{bJ}} S) \leq C H^{-2}(1 + \log^2(H/h)), \tag{18}$$

where M_{bJ} denotes the block Jacobi preconditioner and C is a constant independent of H and h. The number of iterations of the block Jacobi preconditioner, reported in the first row of the tables, doubles successively, which is consistent with the theoretical upper bound in (18).

For HD and LD the behavior of all coarse alternatives is similar to the one observed for Poisson's problem in Table 1, that is, the number of iterations is independent of the number of subdomains. For LA and LDA, this similarity is still true for vertex-linear and vertex-operator-dependent; the three others exhibit a slight insignificant increase in the number of iterations.

For HA, the coarse spaces subdomain-based and vertex-flat do not improve the convergence for a number of subdomains less than 1024. For vertex-linear and vertex-

TABLE 2
The number of PCG iterations varying the number of subdomains with 16×16 *points per subdomain.*

	LA				LD				LDA			
# subdomains	16	64	256	1024	16	64	256	1024	16	64	256	1024
no	17	33	59	114	25	47	83	158	29	55	104	194
sd	18	25	27	28	19	19	19	19	22	30	33	34
vf	19	24	29	31	20	21	21	21	23	28	31	32
vl	15	17	17	17	13	13	12	12	14	16	17	17
vo	15	17	17	17	11	11	11	11	14	16	17	17
ed	19	26	27	28	20	20	18	18	21	26	27	28
	HA				HD				HDA			
# subdomains	16	64	256	1024	16	64	256	1024	16	64	256	1024
no	19	42	69	127	25	50	87	172	37	149	302	629
sd	30	64	75	86	18	19	19	19	30	64	81	83
vf	27	52	72	85	21	22	22	21	31	76	86	99
vl	26	45	66	73	16	18	18	16	21	63	81	89
vo	26	45	66	73	11	11	11	11	20	60	81	88
ed	24	43	57	69	17	19	19	18	31	62	70	77

operator the improvement is modest for 256 processors. It seems that on this example the condition numbers of the preconditioned linear systems with and without the coarse components are comparable, for instance, 2×10^2 with and 6×10^2 without the vertex-linear coarse component using 256 subdomains. More precisely, the smallest eigenvalue is not as affected by the use of the coarse component as it is for the other model problems. In the presence of high anisotropy (HA and HDA), the convergence rate of all the alternatives is comparable and depends on the number of subdomains, while an asymptotic behavior tends to appear when the number of subdomains increases.

5.2.2. Dependency of the coarse preconditioners on $\frac{H}{h}$. To study the sensitivity of the preconditioners to the size of the subdomains (i.e., $\frac{H}{h}$ variable with H constant), we report in Table 3 the experiments observed with 256 subdomains (16×16 box decomposition) when the size of the subdomains varies from 8×8 up to 32×32.

We observe that the convergence of all the preconditioners depends slightly on the size of the subdomains. Furthermore, in the anisotropic experiments (HA and HDA), this dependence is surprisingly negligible for the vertex-linear and vertex-operator-dependent alternatives. The number of iterations of those two preconditioners tends to be stable around 64 and 80 for the problems HA and HDA, respectively.

5.2.3. General comments. Summarizing the 48 experiments reported in Tables 2 and 3, operator-dependent had the best behavior on 19 experiments, edge on 11, and vertex-linear on one. Operator-dependent and vertex-linear had the same behavior on 15 tests, operator-dependent and edge on one, and vertex-linear, edge, and operator-dependent presented the same number of iteration in only one experiment. Although the closest variant to genuine BPS, that is, vertex-linear, appears to be quite robust on the examples considered in this paper, it is outperformed by operator-dependent for discontinuous problems and by edge on anisotropic problems. In addition, as the other vertex-based coarse components, BPS and its variants are no longer applicable when the domain partitioning does not exhibit a coarse mesh, as might be the case on unstructured meshes, while both domain- and edge-based coarse

TABLE 3
The number of PCG iterations varying the grid size of the subdomains from 8×8 *up to* 64×64 *using a* 16×16 *decomposition.*

	LA				LD				LDA			
Subdomain size	64	256	1024	4096	64	256	1024	4096	64	256	1024	4096
no	66	69	78	96	73	83	106	133	97	104	119	150
sd	30	27	36	42	16	19	23	30	29	33	36	45
vf	25	29	32	35	18	21	24	31	27	31	34	40
vl	15	17	19	21	12	14	15	19	15	18	19	22
vo	15	17	19	21	9	11	12	16	15	17	19	22
ed	24	27	32	34	16	18	22	29	24	27	31	36
	HA				HD				HDA			
Subdomain size	64	256	1024	4096	64	256	1024	4096	64	256	1024	4096
no	66	69	73	76	78	87	116	141	280	302	297	389
sd	65	75	76	76	17	19	23	31	65	81	87	96
vf	66	72	75	76	20	22	25	31	80	86	89	91
vl	60	64	64	63	16	18	18	23	79	81	80	77
vo	60	66	64	63	10	11	13	16	77	81	79	80
ed	51	57	59	63	16	19	22	29	63	70	79	86

components are still well defined.

On problems that are not highly anisotropic, all the coarse components give rise to preconditioners that are independent of or weakly dependent on the number of subdomains and that have a mild dependence on the size of the subdomains.

If we compare the LD and HD columns, we see that all the preconditioners are quite insensitive to discontinuity. Further, the vertex-operator-dependent alternative, specifically designed to handle interfaces that cross a discontinuity, has the best performance.

The numerical experiments tend to show that the most dominating component is not the granularity of the coarse space (the finest is not necessarily the best) but the restriction/interpolation operator R_0, as shown in the experiments with vertex. The restriction operator governs, in most cases, the quality of the coarse representation of the complete equation. The flat interpolation operator is always the worst and operator-dependent behaves the best on problems with discontinuities for which it was designed.

For discontinuous problems, LD and HD, vertex-operator-dependent is the most efficient, while for highly anisotropic problems, HA and HDA, edge is the most efficient for all cases but one. For LA and LDA operator-dependent and vertex-linear had the same behavior. The good performance of edge on anisotropic problems can be explained by the fact that we consider anisotropies aligned with the discretization, and because we use regular box decompositions, two opposite edges E_i of a subdomain are strongly coupled. The edge coarse space captures the strong coupling, while the other alternatives mix, and therefore miss, this information. This latter behavior is related to the fact that the supports of the basis vectors of all coarse spaces, excluding edge, contain at least two weakly coupled edges. So the transfer operators are not able, in this specific case, to retrieve and to spread the most appropriate information.

To conclude with the numerical experiments we consider results observed with a semiconductor simulation code, similar to the one described in [17], that solves the drift-diffusion equations using finite difference discretization. We omit the details of the drift-diffusion equation solution and mention only that it involves two nested iterations: the outer is a continuation loop, the inner a nonlinear Gauss–Seidel itera-

tion (Gummel's algorithm) that requires the solution of three linear systems at each step; we refer the reader to [3] and the references therein for further details on those equations and the numerical solution techniques, as well as for the description of the MOSFET semiconductor device considered for these experiments. The numbers of iterations displayed in Table 4 correspond to the average number of iterations for the linear system solutions involved in the Gummel's iterations for the last continuation step. The equations are discretized on a 257×257 nonuniform grid. This table shows that the use of one of the coarse-space preconditioners proposed in this paper enables us to significantly prevent the increase of the number of iterations when the number of subdomains is increased for the solution of a real life application.

TABLE 4
Average number of PCG iterations for the linear solutions involved in the MOSFET semiconductor device simulation.

# subdomains	16	64	256
no	40	70	201
vo	33	54	70

5.3. Parallel behavior. We investigate the parallel scalability of the proposed implementation of the preconditioners. For each experiment, we map one subdomain on each processor of the parallel computer. In what follows, the number of subdomains and the number of processors will be always the same. The target computer is a 128-node T3D located at CERFACS, using MPI as message passing library. The solution of the local Dirichlet problem is performed using either the sparse direct solver MA27 [7] from the Harwell Subroutine Library or a skyline [9] solver. The factorization of the tridiagonal probed matrices, used in the local part of the preconditioners (see (8)) is performed using a LAPACK [1] band solver.

To study the parallel behavior of the code, we report the maximum elapsed time (in seconds) spent by one of the processors in each of the main steps of the domain decomposition method when the number of processors is varied for solving the standard Poisson's equation. In the following tables, the rows titled "init." correspond mainly to the time for factorizing the matrix associated with the local Dirichlet problems; "setup local" is the time to construct and factorize the probing approximations $\tilde{S}_{ii}$; "setup coarse" is the time required to construct and factorize the matrix associated with the coarse problem; "iter." is the time spent in the iteration loop of the preconditioned conjugate gradient. Finally, the row "total" permits us to evaluate the parallel scalability of the complete methods (i.e., numerical behavior and parallel implementation), while "time per iter." illustrates only the scalability of the parallel implementation of the preconditioned conjugate gradient iterations. The elapsed time corresponds to a maximum, and there is some unbalance among the processors in different kernels. Therefore the reported total time differs from the sum of the time for each individual kernel.

To illustrate the extra cost introduced by the construction and the solution of the coarse problem at each iteration, we give in Tables 5, 6, and 7 the time spent in each step of the algorithm with (left column) and without (right column) the considered coarse component of the preconditioner.

We report experiments with the domain-based coarse space in Table 5. Results with the vertex-based coarse space are displayed in Table 6. For those experiments, we use MA27 to solve the local Dirichlet problems defined on 100×100 grids. That subdomain size was the largest we could use according to the 128 MB memory available

TABLE 5

Poisson-MA27: Elapsed time in each main numerical step varying the number of processors with 100×100 *points per subdomain using the domain-based coarse space.*

# procs	4		8		16		32		64		128	
init.	2.58	2.58	2.57	2.57	2.57	2.57	2.57	2.57	2.57	2.57	2.57	2.57
setup local	0.99	0.80	1.00	1.01	1.40	1.31	1.40	1.30	1.40	1.31	1.40	1.32
setup coarse	0.25	0.00	0.48	0.00	0.86	0.00	0.85	0.00	0.87	0.00	0.93	0.00
iter.	1.84	1.98	2.55	3.93	3.01	5.14	4.12	7.16	3.79	9.80	4.91	13.26
total	5.33	5.65	6.23	7.26	7.14	8.50	8.25	10.51	7.93	13.13	9.14	16.60
# iter.	16	20	22	33	26	41	35	58	32	80	40	109
time per iter.	0.12	0.12	0.12	0.12	0.12	0.13	0.12	0.12	0.12	0.12	0.12	0.12

TABLE 6

Poisson-MA27: Elapsed time in each main numerical step varying the number of processors with 100×100 *points per subdomain using the vertex-based coarse space.*

# procs	4		8		16		32		64		128	
init.	2.58	2.58	2.57	2.57	2.57	2.57	2.57	2.57	2.57	2.57	2.57	2.57
setup local	0.66	0.80	0.94	1.01	1.24	1.31	1.24	1.30	1.24	1.31	1.25	1.32
setup coarse	0.72	0.00	0.74	0.00	0.90	0.00	0.90	0.00	0.92	0.00	1.07	0.00
iter.	1.85	2.31	1.97	3.93	2.10	5.14	2.36	7.16	2.03	9.80	2.45	13.26
total	5.61	5.65	6.09	7.26	6.65	8.50	6.91	10.51	6.59	13.13	7.16	16.60
# iter.	16	20	17	33	18	41	20	58	17	80	20	109
time per iter.	0.12	0.12	0.12	0.12	0.12	0.13	0.12	0.12	0.12	0.12	0.12	0.12

TABLE 7

Poisson-Skyline: Elapsed time in each main numerical step varying the number of processors with 80×80 *points per subdomain using the vertex-based coarse space.*

# procs	4		8		16		32		64		128	
init.	2.05	2.05	2.05	2.05	2.05	2.05	2.05	2.05	2.05	2.05	2.05	2.05
setup local	0.79	0.82	1.10	1.14	1.39	1.49	1.39	1.50	1.39	1.49	1.40	1.49
setup coarse	0.49	0.00	0.83	0.00	0.98	0.00	0.98	0.00	1.00	0.00	1.14	0.00
iter.	2.06	2.60	2.26	4.37	2.43	5.83	2.66	8.07	2.31	10.41	2.80	14.38
total	5.45	5.58	6.31	7.67	6.91	9.43	7.15	11.67	6.81	14.02	7.43	17.98
# iter.	15	19	16	31	17	41	19	58	16	73	19	101
time per iter.	0.14	0.14	0.14	0.14	0.14	0.14	0.14	0.14	0.14	0.14	0.15	0.14

on each node of the target computer.

We can first observe that the numerical behavior of those preconditioners is again independent of the number of subdomains. It can be seen that the parallel implementation of the Schur complement method with only a local preconditioner scales perfectly as the time per iteration is constant and does not depend on the number of processors (i.e., 0.115 seconds on 4 processors and 0.118 on 128 nodes; both figures were rounded to 0.12 seconds when reported in Table 5 and 6).

The above scalable behavior is also observed when the coarse components, vertex or subdomain, are introduced. For instance, with the vertex-based preconditioner, the time per iteration grows from 0.116 seconds on 4 processors up to 0.122 seconds on 128 processors (again rounded to 0.12 seconds in Table 6). There are two main reasons for this scalable behavior. First, the solution of the coarse problems is negligible compared to the solution of the local Dirichlet problems. Second, the parallel implementation of the coarse components does not introduce any extra global communication.

In any case the methods scale fairly well, when the number of processors grows from 8 (to solve a problem with 80,000 unknowns) up to 128 (to solve a problem with 1.28 million unknowns). The ratios between the total elapsed time expended for running on 128 and on 8 processors are 1.18, with the vertex-based coarse preconditioner, and 1.47, with the domain-based one. That latter larger value is only due to an increase of the number of iterations.

One of the most expensive kernels of this method is the factorization of the local Dirichlet problems. Therefore, the tremendous reduction in the number of iterations induced by the use of the coarse alternatives—five times fewer iterations for the vertex-based preconditioner—is not reflected directly on a reduction of the total time. The total time is an affine function of the number of iterations with an incompressible overhead due to the Cholesky factorization at the very beginning of the domain decomposition method.

Nevertheless, if we use a less efficient local solver, which will slow down the iteration time, the gap between the solution time with and without the coarse component would increase. To illustrate this behavior, we display in Table 7 the parallel performance of the vertex-based coarse preconditioner, where we replace MA27 by a skyline solver. The size of the subdomains is smaller than the size used for testing with MA27 because the skyline solver is more memory-consuming than MA27, and 80×80 is the biggest size that fits in the memory of the computer.

6. Concluding remarks. We have presented two-level preconditioners for Schur complement domain decomposition methods in two dimensions built by combining a block diagonal preconditioner, a variant of the local component of the BPS preconditioner, with a set of new algebraic coarse-space components. They are defined using the mesh partitioning information and simple interpolation operators that follow a partition of unity principle. We have illustrated their numerical behavior on a set of two-dimensional model problems that have both anisotropy and discontinuity. Those experiments tend to show that the most dominating component is not the granularity of the coarse space (the finest is not necessarily the best) but the restriction/interpolation operator R_0. This operator governs, in most cases, the quality of the coarse representation of the complete equation. Two of the new coarse alternatives present a number of iterations equal to or less than the traditional BPS on problems which have anisotropy and/or discontinuity. One of these alternatives, edge-based, is a little more expensive. Nevertheless, as the coarse part of the preconditioner impacts neither the overall construction of the two-level preconditioner nor the cost of each iteration, these alternatives are a valid contribution on the development of this area.

The experiments have been performed on regular meshes, but there is no limitation for the implementation of the proposed two-level preconditioners on unstructured grids, whereas the possible rank deficiency, which appears in the domain-coarse alternative, may be more tricky to discard.

Those numerical methods are targeted for parallel distributed memory computers. In this respect, we have proposed a message-passing implementation that does not require any new global synchronization in the preconditioned conjugate gradient iterations, which is a well-known bottleneck for Krylov methods in distributed memory environments. We illustrated that the numerical scalability of the preconditioners combined with the parallel scalability of the implementation result in a set of parallel scalable numerical methods.

Finally, we point out that two-dimensional problems with anisotropy and discontinuity arise in device modeling. In an ongoing work, some of our new coarse pre-

conditioners have been successfully implemented in a two-dimensional unstructured mixed finite element code for the simulation of semiconductor devices [12].

REFERENCES

[1] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Oustrouchov, and D. Sorensen, *LAPACK Users' Guide*, 2nd ed., SIAM, Philadelphia, 1995.

[2] J. Bramble, J. Pasciak, and A. Schatz, *The construction of preconditioners for elliptic problems by substructuring.* I., Math. Comp., 47 (1986), pp. 103–134.

[3] L. Carvalho, *Preconditioned Schur Complement Methods in Distributed Memory Environments*, Ph.D. thesis, INPT/CERFACS, France, 1997.

[4] T. Chan and T. Mathew, *Domain decomposition algorithms*, in Acta Numer. 3, Cambridge University Press, Cambridge, UK, 1994, pp. 61–143.

[5] T. F. Chan and T. Mathew, *The interface probing technique in domain decomposition*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 212–238.

[6] M. Dryja, B. Smith, and O. Widlund, *Schwarz analysis of iterative substructuring algorithms for elliptic problems in three dimensions*, SIAM J. Numer. Anal., 31 (1994), pp. 1662–1694.

[7] I. Duff and J. Reid, *MA27—A Set of Fortran Subroutines for Solving Sparse Symmetric Sets of Linear Equations*, Tech. Report AERE R10533, Rutherford Appleton Laboratory, London, 1982.

[8] C. Farhat and F.-X. Roux, *A method of finite element tearing and interconnecting and its parallel solution algorithm*, Internat. J. Numer. Methods Engrg., 32 (1991), pp. 1205–1227.

[9] A. George and J. W. H. Liu, *Computer Solution of Large Sparse Positive Definite Systems*, Prentice-Hall Series in Computational Mathematics, Englewood Cliffs, NJ, 1981.

[10] L. Giraud, F. Guevara Vasquez, and R. S. Tuminaro, *Grid Transfer Operators for Highly Variable Coefficient Problems in Two-Level Non-Overlapping Domain Decomposition Methods*, CERFACS Tech. Report TR/PA/01/03, Toulouse, France, 2001.

[11] L. Giraud and R. S. Tuminaro, *Schur complement preconditioners for anisotropic problems*, IMA J. Numer. Anal., 19 (1999), pp. 1–17.

[12] F. Hecht and A. Marrocco, *Mixed finite element simulation of heterojunction structures including a boundary layer model for the quasi-Fermi levels*, COMPEL, 13 (1995), pp. 757–770.

[13] P. Le Tallec, *Domain decomposition methods in computational mechanics*, Comput. Mech. Adv., 1 (1994), pp. 121–220.

[14] J. Mandel, *Balancing domain decomposition*, Comm. Numer. Methods Engrg., 9 (1993), pp. 233–241.

[15] J. Mandel and M. Brezina, *Balancing domain decomposition for problems with large jumps in coefficients*, Math. Comp., 65 (1996), pp. 1387–1401.

[16] J. Mandel and R. Tezaur, *Convergence of substructuring method with Lagrange multipliers*, Numer. Math., 73 (1996), pp. 473–487.

[17] J. C. Meza and R. S. Tuminaro, *A multigrid preconditioner for the semiconductor equations*, SIAM J. Sci. Comput., 17 (1996), pp. 118–132.

[18] A. Quarteroni and A. Valli, *Domain Decomposition Methods for Partial Differential Equations*, Numer. Math. Sci. Comput., Oxford University Press, Oxford, 1999.

[19] B. Smith, *Domain Decomposition Algorithms for the Partial Differential Equations of Linear Elasticity*, Tech. Report 517, Department of Computer Science, Courant Institute, New York, 1990.

[20] B. Smith, P. Bjørstad, and W. Gropp, *Domain Decomposition, Parallel Multilevel Methods for Elliptic Partial Differential Equations*, 1st ed., Cambridge University Press, New York, 1996.

[21] W. Wan, *Scalable and Multilevel Iteratives Methods*, Ph.D. thesis, Department of Mathematics, UCLA, Los Angeles, 1998.

[22] W. L. Wan, T. F. Chan, and B. Smith, *An energy-minimizing interpolation for robust multigrid methods*, SIAM J. Sci. Comput., 21 (2000), pp. 1632–1649.

[23] P. Wesseling, *An Introduction to Multigrid Methods*, Wiley, West Sussex, UK, 1992.

SIAM J. SCI. COMPUT.
Vol. 22, No. 6, pp. 2006–2028
© 2001 Society for Industrial and Applied Mathematics

ASYMPTOTIC ANALYSIS OF THE LAMINAR VISCOUS FLOW OVER A POROUS BED*

WILLI JÄGER†, ANDRO MIKELIĆ‡, AND NICOLAS NEUSS†

Abstract. We consider the laminar viscous channel flow over a porous surface. The size of the pores is much smaller than the size of the channel, and it is important to determine the effective boundary conditions at the porous surface. We study the corresponding boundary layers, and, by a rigorous asymptotic expansion, we obtain Saffman's modification of the interface condition observed by Beavers and Joseph. The effective coefficient in the law is determined through an auxiliary boundary-layer type problem, whose computational and modeling aspects are discussed in detail. Furthermore, the approximation errors for the velocity and for the effective mass flow are given as powers of the characteristic pore size ε. Finally, we give the interface condition linking the effective pressure fields in the porous medium and in the channel, and we determine the jump of the effective pressures explicitly.

Key words. homogenization, periodic structures, Navier–Stokes, Beavers–Joseph, interface law, boundary layer, unbounded domain, finite elements, multigrid

AMS subject classifications. 35B27, 35Q30, 65N30, 74Q15, 76D05, 76M10, 76M50

PII. S1064827599360339

1. Introduction.

Finding effective boundary conditions at the surface which separates a channel flow and a porous medium is a classical problem.

Supposing a laminar incompressible and viscous flow, we find out immediately that the effective flow in a porous solid is described by Darcy's law. In the free fluid we obviously keep the Navier–Stokes system. Hence we have two completely different systems of partial differential equations. First, Darcy's law combined with the incompressibility gives a second order equation for the pressure and a first order system for the velocity. In the Navier–Stokes system, the orders of the corresponding differential operators are different, and it is not clear what kind of conditions one should impose at the interface between the free fluid and the porous part. Clearly, due to the incompressibility, the normal mass flux should be continuous. Other classically used conditions are the continuity of the pressure and, for a free fluid, the vanishing of the tangential velocity at the interface.

Let us discuss the mathematical background of the interface conditions. It is well known that Darcy's law is a statistical result giving the average of the momentum equation (the Navier–Stokes equations) over the pore structure. Its rigorous derivation involves the weak convergence in $H(\Omega, \mathrm{div})$ (respectively, the two-scale convergence) of velocities, and only the continuity of the normal velocities is preserved. Other continuity conditions at the interface are generally lost, such that further analysis is required.

Concerning other interface conditions used in engineering literature, the vanishing

*Received by the editors August 26, 1999; accepted for publication (in revised form) November 8, 2000; published electronically March 20, 2001. This work was supported by the Deutsche Forschungsgemeinschaft (DFG) through the Sonderforschungsbereich 359 (SFB 359) at the University of Heidelberg.

http://www.siam.org/journals/sisc/22-6/36033.html

†Interdisziplinäres Zentrum für wissenschaftliches Rechnen, Im Neuenheimer Feld 368, D-69120 Heidelberg, Germany (jaeger@iwr.uni-heidelberg.de, nicolas.neuss@iwr.uni-heidelberg.de).

‡UFR Mathématiques, Bât 101, Université Claude Bernard Lyon 1, 43, bd. du onze novembre, 69622 Villeurbanne Cedex, France (andro@lan.univ-lyon1.fr).

of the tangential velocity is found to be an unsatisfactory approximation, and in [2] a new condition is proposed. The condition reads

$$\nabla \vec{u}^{eff} \cdot \vec{\nu} \cdot \vec{\tau} = \alpha K^{-1/2} (\vec{u}^{eff} - \vec{v}^{f}) \cdot \vec{\tau}, \tag{1.1}$$

where $\vec{u}^{eff}$ is the effective velocity in the channel, $\vec{v}^{f}$ is the mean filtration velocity given by Darcy's law, $\vec{\tau}$ is a tangent vector to the interface, $\vec{\nu}$ is the normal into the fluid, K is the permeability of the porous medium, and the scalar α is a function of the geometry of the porous medium.

In [2], this law is derived by heuristic arguments and is justified experimentally. A theoretical attempt to derive (1.1) is undertaken in [17], and, using a statistical approach, a Brinkman type approximation in the transition layer is derived. A matching argument then allows us to obtain the formula

$$\vec{u}^{eff} \cdot \vec{\tau} = \frac{1}{\alpha} K^{1/2} \nabla \vec{u}^{eff} \cdot \vec{\nu} \cdot \vec{\tau} + \mathcal{O}(K). \tag{1.2}$$

The interested reader can also consult the lecture notes [4].

Different considerations can be found in [5] and [12]. They distinguish two cases.

(a) The pressure gradient on the side of the porous solid of the interface is normal to the interface. Consequently, we have a balanced flow on both sides of the interface. Then, using an asymptotic point of view, the following laws are obtained in [12]:

$$\vec{u}^{eff} \cdot \vec{\nu} = \vec{v}^{f} \cdot \vec{\nu}, \quad p = \text{ const.} \tag{1.3}$$

on the interface. This case describes the flows in cavities. The mathematical justification is in [9]. We shall not consider it in this paper.

(b) The pressure gradient on the side of the porous solid at the interface is not normal. This case is considered in the fundamental paper [5]. After discussing the orders of magnitude of the unknowns, it is found that on the interface the velocity of the free fluid is zero, and the pressure is continuous.

All results cited above are not mathematically rigorous. Furthermore, different approaches give different results, and two natural questions arise immediately.

(Q1) What are the correct matching conditions (i.e., conditions at the interface) between those two flow equations?

(Q2) What are the effective constants entering the matching conditions?

We are going to answer those questions in the following. In section 2, we define our problem and discuss some simple approximations. In section 3, we introduce an important auxiliary problem of boundary-layer type which we need to construct a better approximation, and section 4 gives additional results for an analogous problem on a finite strip. Then, in section 6, we show how the computation of the constants involved in the interface conditions works in practice. Finally, in section 7, the effective equations with the Beavers–Joseph type boundary condition are presented together with improved error estimates. It turns out that the difference between Darcy's pressure in the porous part and the effective channel pressure is equal to a constant multiple of the effective channel shear stress, thus contradicting [12].

2. Setting of the problem. This section deals with the equations describing fluid motion over a porous bed, under a uniform pressure gradient. We assume the condition of the experiment by Beavers and Joseph [2], i.e., a stationary laminar incompressible viscous flow.

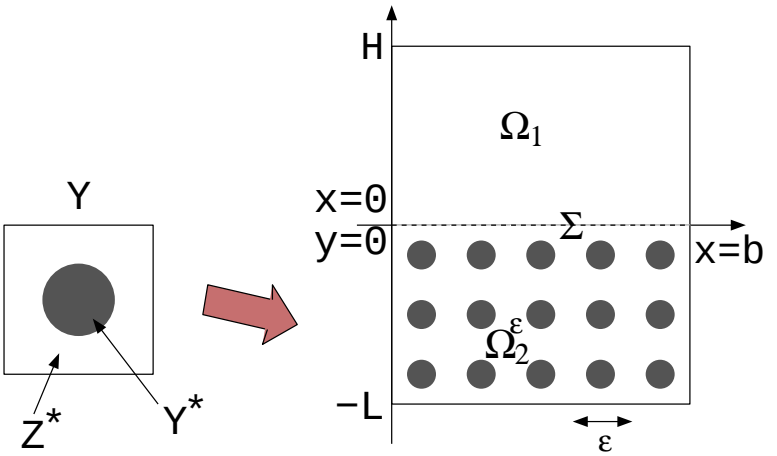


FIG. 2.1. *The flow region* Ω^ε.

For simplicity reasons, we consider a flow over a periodic porous medium with a characteristic pore size ε. The flow region $\Omega^\varepsilon \subset \Omega = (0,b) \times (-L,H)$ consists of two parts; see Figure 2.1. The upper part $\Omega_1 = (0,b) \times (0,H)$ is the channel, and the lower part Ω_2^ε is the porous medium which is obtained by putting solid obstacles of size $\varepsilon \ll b$ into the domain $\Omega_2 = (0,b) \times (-L,0)$. With Σ we denote the (permeable) interface between Ω_1 and Ω_2. More precisely, let $Y = (0,1)^2$ be made of two complementary parts, the solid part Z^* and the fluid part Y^*. It is assumed that Z^* is a smooth closed subset of Y, strictly included in Y, $Y^* \cap Z^* = \emptyset$, and $Y^* \cup Z^* = Y$. Now assume that both b and L are integer multiples of ε. Then the domain $\Omega_2 = (0,b) \times (-L,0)$ can be covered by a regular mesh of $N(\varepsilon)$ cells of size ε. Each cell $Y_k^\varepsilon = \varepsilon(Y+k)$ is divided into a fluid part $\varepsilon(Y^*+k)$ and a solid part $\varepsilon(Z^*+k)$. The fluid part Ω_2^ε of the porous medium is therefore

$$\Omega_2^\varepsilon = \Omega_2 \setminus \bigcup_{k=1}^{N(\varepsilon)} \varepsilon(Z^*+k) = \Omega_2 \cap \bigcup_{k=1}^{N(\varepsilon)} \varepsilon(Y^*+k)\,, \tag{2.1}$$

and the whole flow region is

$$\Omega^\varepsilon = \Omega_1 \cup \Sigma \cup \Omega_2^\varepsilon\,. \tag{2.2}$$

After specifying the geometry, we consider the equations determining the velocity field u^ε and the pressure field p^ε in the Beavers–Joseph experiment:

$$-\mu \triangle \vec{u}^\varepsilon + (\vec{u}^\varepsilon \nabla)\vec{u}^\varepsilon + \nabla p^\varepsilon = 0 \quad \text{in } \Omega^\varepsilon\,, \tag{2.3}$$

$$\operatorname{div} \vec{u}^\varepsilon = 0 \quad \text{in } \Omega^\varepsilon\,, \tag{2.4}$$

$$\vec{u}^\varepsilon = 0 \quad \text{on } (\partial\Omega^\varepsilon \setminus \partial\Omega) \cup (0,b) \times (\{-L\} \cup \{H\})\,, \tag{2.5}$$

$$u_2^\varepsilon = 0 \quad \text{on } (\{0\} \cup \{b\}) \times (-L,H)\,, \tag{2.6}$$

$$p^\varepsilon = p_0 \quad \text{on } \{0\} \times (-L,H)\,, \tag{2.7}$$

$$p^\varepsilon = p_b \quad \text{on } \{b\} \times (-L,H)\,. \tag{2.8}$$

For small ε, this problem is extremely difficult to solve, so one has to look for approximations. Classically, the system (2.3)–(2.8) used to be approximated by a

Poiseuille flow in Ω_1, i.e., by

$$\begin{array}{rcll} \vec{v}^0 & = & \left(\dfrac{p_b - p_0}{2b\mu} x_2(x_2 - H), 0\right) & \text{for} \quad 0 \le x_2 \le H, \\ \vec{v}^0 & = & 0 & \text{for} \quad -L \le x_2 < 0, \\ p^0 & = & \dfrac{p_b - p_0}{b} x_1 + p_0 & \text{for} \quad 0 \le x_1 \le b, \end{array} \tag{2.9}$$

and, indeed, it could be shown in [10] that (2.9) is an approximation in the following sense:

$$\int_{\Omega_p^\varepsilon} |\vec{u}^\varepsilon(x)|^2 \, dx \le C\varepsilon^3 \,, \tag{2.10}$$

$$\int_{\Omega_1} |\vec{u}^\varepsilon(x) - \vec{v}^0(x)|^2 \, dx + \int_\Sigma |\vec{u}^\varepsilon(x_1, 0)|^2 \, dx_1 \le C\varepsilon^2 \,, \tag{2.11}$$

$$\int_{\Omega_1} |p^\varepsilon(x) - p^0(x)|^2 \, dx + \int_{\Omega^\varepsilon} |\nabla \vec{u}^\varepsilon(x) - \nabla \vec{v}^0(x)|^2 \, dx \le C\varepsilon \,, \tag{2.12}$$

$$\forall \gamma > 0 : \quad b \int_0^H |u_1^\varepsilon(0, x_2) - v_1^0(x_2)| \, dx_2 \le C\varepsilon^{3/4 - \gamma} \,. \tag{2.13}$$

The above estimates indicate that in the L^2-sense $\vec{u}^\varepsilon = \vec{v}^0 + O(\varepsilon)$ in Ω_1, $\vec{u}^\varepsilon = O(\varepsilon^{3/2})$ in Ω_2, $\nabla \vec{u}^\varepsilon = \nabla \vec{v}^0 + O(\sqrt{\varepsilon})$ in Ω^ε, $p^\varepsilon = p^0 + O(\sqrt{\varepsilon})$ in Ω_1, $p^\varepsilon = O(1)$ in Ω_2^ε, $\vec{u}^\varepsilon = O(\varepsilon)$ on Σ, and the effective mass flow behaves as $-H^3 \frac{p_b - p_0}{12\mu} + O(\varepsilon^{3/4-\gamma})$.

As shown in [2] and [17], this $O(\varepsilon)$ approximation often is not good enough. Therefore, we would like to continue with the asymptotic expansion for $\vec{u}^\varepsilon$ and p^ε. Energy estimates from [10] imply that $\nabla \vec{u}^\varepsilon = \nabla \vec{v}^0 + O(\varepsilon)$ in the interior of Ω_1 but that $\nabla \vec{u}^\varepsilon = \nabla \vec{v}^0 + O(\sqrt{\varepsilon})$ globally in Ω. Further, they show that there is an oscillatory boundary layer confined to the neighborhood of Σ which can be represented in the form

$$\vec{\beta}^{bl,\varepsilon}(x) = \varepsilon \vec{\beta}^{bl}\left(\frac{x}{\varepsilon}\right) \qquad \text{and} \qquad \omega^{bl,\varepsilon}(x) = \omega^{bl}\left(\frac{x}{\varepsilon}\right), \quad x \in \Omega^\varepsilon, \tag{2.14}$$

where $\vec{\beta}^{bl}, \omega^{bl}$ are solutions to a Stokes problem on an infinite strip Z^{bl}, which we discuss in section 3.

3. The auxiliary boundary layer problem. Let $Z^{bl} = Z^+ \cup S \cup Z^-$ with $Z^+ = (0,1) \times (0, +\infty)$, $S = (0,1) \times \{0\}$, and $Z^- = (0,1) \times (-\infty, 0) \setminus \cup_{k=1}^\infty (Z^* - \{0, k\})$; see Figure 3.1. For later use, we also introduce the following notation ($k \in \mathbb{Z}$):

$$\begin{array}{c} Z_k = Z^{bl} \cap (0,1) \times (k, k+1) \,, Z_l^k = Z^{bl} \cap (0,1) \times (l, k) \,, \\ Z_{<k} = Z_{-\infty}^0 \,, Z_{>k} = Z_0^{+\infty}. \end{array} \tag{3.1}$$

Now $\{\vec{\beta}^{bl}, \omega^{bl}\}$ are the solutions to the problem

$$-\triangle_y \vec{\beta}^{bl} + \nabla_y \omega^{bl} = 0 \qquad \text{in } Z^+ \cup Z^- \,, \tag{3.2}$$

$$\operatorname{div}_y \vec{\beta}^{bl} = 0 \qquad \text{in } Z^+ \cup Z^- \,, \tag{3.3}$$

$$\left[\vec{\beta}^{bl}\right]_S (\cdot, 0) = 0 \quad \text{on } S \,, \tag{3.4}$$

$$\left[(\nabla_y \vec{\beta}^{bl} - \omega^{bl} I) e_2\right]_S (\cdot, 0) = e_1 \quad \text{on } S \,, \tag{3.5}$$

$$\vec{\beta}^{bl} = 0 \quad \text{on } \bigcup_{k=1}^\infty \{\partial Z^* - (0,k)\}, \quad \{\vec{\beta}^{bl}, \omega^{bl}\} \text{ is } y_1 - \text{periodic.} \tag{3.6}$$

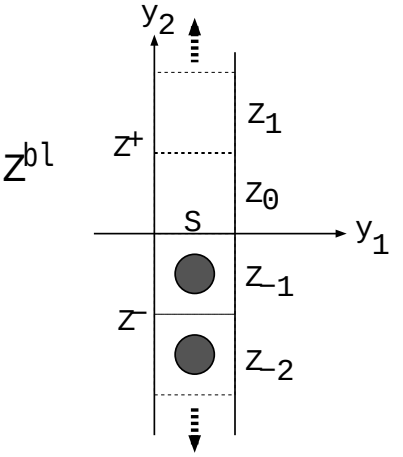


FIG. 3.1. *The boundary layer cell* Z^{bl}.

Further, in order to define ω^{bl} uniquely, we require

$$\omega^{bl}(y_1, y_2) \to 0 \quad (y_2 \to -\infty)\,. \tag{3.7}$$

A variational formulation of this problem is as follows. Find a locally square integrable vector field $\vec{\beta}^{bl} \in W$ satisfying

$$\int_{Z^{bl}} \nabla\vec{\beta}^{bl}\nabla\vec{\varphi} = -\int_S \vec{e}_1\vec{\varphi} \quad \forall\vec{\varphi} \in W, \tag{3.8}$$

where W is the function space which contains all y_1-periodic, divergence-free, locally square integrable vector fields $\vec{z}$ defined on Z^{bl}, having finite viscous energy $\int_{Z^{bl}} |\nabla\vec{z}(y_1, y_2)|^2\, dy$, and satisfying the no-slip boundary condition at the solid boundaries $\cup_{k=1}^{\infty}(\partial Z^* - \{0, k\})$.

In [9, Proposition 3.22, pp. 462–463] the following result is proved.

THEOREM 3.1. *The problem* (3.8) *has a unique solution* $\vec{\beta}^{bl} \in W$. *It is locally infinitely differentiable outside of the interface* S. *Furthermore, a pressure field* ω^{bl} *exists, which is unique up to an additive constant and locally infinitely differentiable outside of the interface* S, *such that* (3.2) *holds. In a neighborhood of* S, $\vec{\beta}^{bl} - ((y_2 - y_2^2/2)e^{-y_2}H(y_2), 0) \in W^{2,q}$ *and* $\omega^{bl} \in W^{1,q}$ *for all* $q \in [1, +\infty)$.

The following lemma states some simple properties of $\{\vec{\beta}^{bl}, \omega^{bl}\}$:

LEMMA 3.2. *Extend* $\vec{\beta}^{bl}$ *with* $\vec{\beta}^{bl} = 0$ *on* $\cup_{k=1}^{l}(Z^* - (0, k))$. *Then* $\{\vec{\beta}^{bl}, \omega^{bl}\}$ *satisfies*

$$\int_0^1 \beta_2^{bl}(y_1, a)\, dy_1 = 0 \quad \forall a \in \mathbb{R}, \tag{3.9}$$

$$\int_0^1 \omega^{bl}(y_1, a)\, dy_1 = \int_0^1 \omega^{bl}(y_1, b)\, dy_1 \quad \forall a, b \geq 0, \tag{3.10}$$

$$\int_0^1 \beta_1^{bl}(y_1, a)\, dy_1 = \int_0^1 \beta_1^{bl}(y_1, b)\, dy_1 \quad \forall a, b \geq 0, \tag{3.11}$$

$$\int_0^1 \beta_1^{bl}(y_1, 0)\, dy_1 = -\int_{Z^{bl}} |\nabla\vec{\beta}^{bl}(y)|^2\, dy\,. \tag{3.12}$$

Proof. See [11]. □

We expect that the problem (3.2)–(3.7) represents a boundary layer. This means that changes of the velocity and the pressure fields are concentrated around the interface S and vanish very rapidly with increasing distance from S. In linear elasticity, results of this type are called Saint-Venant's principle. Saint-Venant's principle is also valid in our case, and we want to discuss this in more detail.

We start our considerations with Z^+ (the part of Z^{bl} filled by the fluid). Here, one can obtain sharp decay estimates by using results for the decay of solutions of general elliptic equations (see Theorem 10.1 from [13], which is an application of Tartar's lemma).

THEOREM 3.3. *Let*

$$\xi^{bl} = \operatorname{curl} \vec{\beta}^{bl} = \frac{\partial \beta_1^{bl}}{\partial y_2} - \frac{\partial \beta_2^{bl}}{\partial y_1}\,. \tag{3.13}$$

Then, for every $a > 0$ and $\alpha \in \mathbb{N}^2$, we have

$$\left|D^\alpha \xi^{bl}(y_1, y_2)\right| \le C(\alpha, a) e^{-2\pi y_2} \quad \forall y_1 \in (0,1), y_2 > a\,. \tag{3.14}$$

Proof. See [11]. □

With the help of this estimate for $\xi^{bl} = \operatorname{curl} \vec{\beta}^{bl}$, we can prove exponentially fast stabilization of $\vec{\beta}^{bl}$.

COROLLARY 3.4. *Let*

$$C_1^{bl} := \int_0^1 \beta_1^{bl}(y_1, 0)\, dy_1\,. \tag{3.15}$$

Then for every $\alpha \in \mathbb{N}^2$, $a > 0$, and $\delta < 2\pi$ we have

$$\left|\vec{\beta}^{bl}(y_1, y_2) - (C_1^{bl}, 0)\right| + \left|D^\alpha \vec{\beta}^{bl}(y_1, y_2)\right| \le C(a, \delta, \alpha) e^{-\delta y_2} \quad \forall y_2 > a\,. \tag{3.16}$$

Proof. See [11]. □

For the pressure field ω^{bl}, we have the following.

COROLLARY 3.5. *For*

$$C_\omega^{bl} := \int_0^1 \omega^{bl}(y_1, 0)\, dy_1 \tag{3.17}$$

we have

$$\left|\omega^{bl}(y_1, y_2) - C_\omega^{bl}\right| \le C e^{-2\pi y_2} \quad \forall y_2 > 0\,. \tag{3.18}$$

Proof. See [11]. □

Now we turn our attention to the porous part Z^-. Due to the presence of the solid obstacles, our estimates are much less precise in this case.

LEMMA 3.6. *Let the distance between the solid obstacles and the boundary of the unit cell Y be bigger than or equal to $2d_*$. Let $\vec{\beta}^{bl}$ be the solution of* (3.2)–(3.7)*, and let $l \in \mathbb{Z}, l < -1$. We introduce a function $\vec{\beta}^{lower}$ by*

$$\vec{\beta}^{lower}(y) = \vec{\beta}^{bl} \quad for \quad y_2 \ge l, \tag{3.19}$$

$$\vec{\beta}^{lower}(y) = 0 \quad for \quad y_2 \le l - d_*, \tag{3.20}$$

$$\beta_1^{lower}(y) = \frac{-y_2+l-d_*}{d_*^2}\Bigg[(-y_2+l-d_*)\left(\beta_1^{bl}(y_1,y_2)+\frac{2}{d_*}\int_0^{y_1}\beta_2^{bl}(\xi,2l-y_2)\,d\xi\right)$$

$$+2\int_{y_2}^{l}\beta_1^{bl}(y,\eta)\,d\eta+2\int_0^{y_1}\beta_2^{bl}(\xi,l)d\xi+\frac{4}{d_*}\int_l^{2l-y_2}\int_0^{y_1}\beta_2^{bl}(\xi,\eta)\,d\xi\,d\eta\Bigg], \tag{3.21}$$

$$\beta_2^{lower}(y) = \frac{(y_2-l-d_*)^2}{d_*^2}\left(\beta_2^{bl}(y_1,y_2)+\frac{2}{d_*}\int_l^{2l-y_2}\beta_2^{bl}(y_1,\eta)\,d\eta\right) \tag{3.22}$$

$$for\quad y\in[0,1]\times(l-d_*,l)\,.$$

Then $\vec{\beta}^{lower}\in W\cap H^1(Z^{bl})^2$, *and with* $Z_{l,l-1}:=Z_l\cup(0,1)\times\{l\}\cup Z_{l-1}$, *we have*

$$\|\nabla\vec{\beta}^{lower}\|^2_{L^2(Z_l)^4}\le C^{lower}\|\nabla\vec{\beta}^{bl}\|^2_{L^2(Z_{l,l-1})^4}\,, \tag{3.23}$$

where the constant C^{lower} *satisfies*

$$C^{lower}\le\frac{28}{d_*^3}+C_P\frac{392}{d_*^4} \tag{3.24}$$

with C_P *denoting the Poincaré constant appearing in*

$$\left\|\beta_j^{bl}\right\|^2_{L^2(Z_{l,l-1})}\le C_P\left\|\nabla\beta_j^{bl}\right\|^2_{L^2(Z_{l,l-1})^2}. \tag{3.25}$$

If we further assume that the ball $B_\rho(1/2,1/2)$ *with radius* ρ *and center* $(1/2,1/2)$ *is contained in* Z^*, *an easy calculation in polar coordinates yields the estimate*

$$C_P\le\frac{1}{4}\ln\frac{1}{\rho\sqrt{2}}, \tag{3.26}$$

which (together with $d_*\le 1$*) results in the estimate*

$$\|\nabla\vec{\beta}^{lower}\|^2_{L^2(Z_l)^4}\le\frac{14}{\rho d_*^4}\left(2d_*+7\ln\frac{1}{\rho\sqrt{2}}\right)\|\nabla\vec{\beta}^{bl}\|^2_{L^2(Z_l\cup Z_{l-1})^4}\,. \tag{3.27}$$

Proof. This is a tedious but straightforward calculation; for details see [11]. ∎

With the help of Lemma 3.6, we are able to deduce the exponential decay of $\nabla\vec{\beta}^{bl}$.

PROPOSITION 3.7. *Assume that* $l\in\mathbb{Z}$ *with* $l<0$. *Define* $Z_{<l}$ *as in* (3.1). *Then the solution* $\vec{\beta}^{bl}$ *of problem* (3.8) *fulfills*

$$\|\nabla\vec{\beta}^{bl}\|_{L^2(Z_{<l})}\le\|\nabla\vec{\beta}^{bl}\|_{L^2(Z^-)^4}e^{-\gamma^{lower}|l|}\,, \tag{3.28}$$

where

$$\gamma^{lower}=\frac{1}{2}\ln\left(\frac{C^{lower}+1}{C^{lower}}\right). \tag{3.29}$$

Proof. If we test (3.8) with some function $\vec{\beta}^{bl}-\vec{\beta}^{lower}$, where $\vec{\beta}^{lower}$ is the function constructed in the previous lemma, this results in the estimate

$$\|\nabla\vec{\beta}^{bl}\|^2_{L^2(Z_{<l})^4}\le C^{lower}\|\nabla\vec{\beta}^{bl}\|^2_{L^2(Z_{l,l-1})^4}\,. \tag{3.30}$$

Using the hole-filling technique as in [9, Lemma 2.4] (we add $C_{lower}\|\nabla\vec{\beta}^{bl}\|^2_{L^2(Z_{<l-2})}$ to both sides of (3.30) and evaluate the resulting recursion), we obtain exponential decay with rate γ^{lower} given by (3.29). □

Applying local regularity results, one immediately obtains the following corollary.

COROLLARY 3.8. *For any $a < 0$, $\alpha \in \mathbb{N}$, the solution $\{\vec{\beta}^{bl}, \omega^{bl}\}$ decays exponentially for $y_2 \to -\infty$, i.e.,*

$$|D^\alpha \vec{\beta}^{bl}(y_1, y_2)| + |D^\alpha \omega^{bl}(y_1, y_2)| \le C(a, \alpha) e^{-\gamma^{lower}|y_2|} \tag{3.31}$$

for all $y_2 < a < 0$.

The above results imply that $\vec{\beta}^{bl}$ is a boundary-layer type velocity field and ω^{bl} is a boundary-layer type pressure. Only the constants C_1^{bl} and C_ω^{bl} from (3.15) and (3.17) will enter the effective flow equations in the channel. They contain the information about the geometry of the porous bed.

REMARK 3.9. *As we shall see from the numerical examples, in general $C_\omega^{bl} \neq 0$. However, if the geometry of Z^- is axisymmetric with respect to reflections around the axis $y_1 = 1/2$, i.e., $\partial Z^*(y_1, y_2) = \partial Z^*(1 - y_1, y_2)$, then $C_\omega^{bl} = 0$. This result is obtained by the following simple argument.*

Let Z^- be axisymmetric around the axis $y_1 = 1/2$. Let $\vec{\beta}^{bl}$ be a solution for (3.2)–(3.7). *Then $(\beta_1^{bl}(1-y_1, y_2),\ -\beta_2^{bl}(1-y_1, y_2),\ -\omega^{bl}(1-y_1, y_2))$ is also a solution. By uniqueness, it must be equal to $(\vec{\beta}^{bl}, \omega^{bl})$ and we conclude that $\omega^{bl}(1 - y_1, y_2) = -\omega^{bl}(y_1, y_2)$. Therefore,* (3.10) *implies $C_\omega^{bl} = 0$.*

4. Approximation of the boundary layer problem on a finite domain. In this section we propose a scheme for computing the actual values of C_1^{bl} and C_ω^{bl} for cases where the geometry of the porous medium is known. Since problem (3.2)–(3.7), respectively, (3.8) is defined on an infinite domain Z^{bl}, the first step is to approximate its solution with solutions of problems defined on finite domains of the form $Z_l^k := Z^{bl} \cap (0, 1) \times (l, k)$, where $l < 0 < k$. Let, therefore, $\{\vec{\beta}_{k,l}^{bl}, \omega_{k,l}^{bl}\}$ be the solution of

$$-\triangle_y \vec{\beta}_{k,l}^{bl} + \nabla_y \omega_{k,l}^{bl} = 0 \quad \text{in } Z_l^0 \cup Z_0^k\,, \tag{4.1}$$

$$\operatorname{div}_y \vec{\beta}_{k,l}^{bl} = 0 \quad \text{in } Z_l^0 \cup Z_0^k\,, \tag{4.2}$$

$$\left[\vec{\beta}_{k,l}^{bl}\right]_S(\cdot, 0) = 0 \quad \text{on } S\,, \tag{4.3}$$

$$\left[\{\nabla_y \vec{\beta}_{k,l}^{bl} - \omega_{k,l}^{bl} I\} e_2\right]_S(\cdot, 0) = e_1 \quad \text{on } S\,, \tag{4.4}$$

$$\vec{\beta}_{k,l}^{bl} = 0 \quad \text{on } \bigcup_{n=1}^{\infty} \{\partial Z^* - (0, n)\}, \quad \{\vec{\beta}_{k,l}^{bl}, \omega_{k,l}^{bl}\} \text{ is } y_1 - \text{periodic}\,, \tag{4.5}$$

with the following additional boundary conditions motivated by Corollary 3.4 and Proposition 3.7:

$$\vec{\beta}_{k,l}^{bl}(y_1, l) = (0, 0)\,, \quad (\beta_{k,l}^{bl})_2(y_1, k) = \frac{\partial (\beta_{k,l}^{bl})_1}{\partial y_2}(y_1, k) = 0 \quad \forall y_1 \in [0, 1]\,. \tag{4.6}$$

Further, to define the pressure field $\omega_{k,l}^{bl}$ uniquely, we require

$$\int_{y_2 = l} \omega_{k,l}^{bl}\, dy = 0\,. \tag{4.7}$$

Let us define

$$C^{bl}_{1,k,l} := \int_0^1 (\beta^{bl}_{k,l})_1(y_1, 0)\, dy_1 \tag{4.8}$$

and

$$C^{bl}_{\omega,k,l} := \int_0^1 \omega^{bl}_{k,l}(y_1, 0)\, dy_1 = \int_0^1 \omega^{bl}_{k,l}(y_1, a)\, dy_1 \quad \forall\, 0 < a \le k\,. \tag{4.9}$$

Then, analogous to the estimates on $(\vec{\beta}^{bl}, \omega^{bl})$ of the previous section, one can prove the following properties for the solution $(\vec{\beta}^{bl}_{k,l}, \omega^{bl}_{k,l})$ of (4.1)–(4.7).

LEMMA 4.1. *For every $0 < a < k$, $0 < \delta < 2\pi$, and $\alpha \in \mathbb{N}^2$, there are constants $C(a, \delta, \alpha)$ such that for all $y \in Z^k_a$ we have*

$$|\vec{\beta}^{bl}_{k,l}(y_1, y_2) - (C^{bl}_{1,k,l}, 0)| + |D^\alpha \vec{\beta}^{bl}_{k,l}(y_1, y_2)| \le C(a, \alpha, \delta) e^{-\delta y_2}\,, \tag{4.10}$$

and

$$\left|\omega^{bl}_{k,l}(y_1, y_2) - C^{bl}_{\omega,k,l}\right| \le C e^{-2\pi y_2}\,. \tag{4.11}$$

For $m \in \mathbb{Z}, l < m < -1$, we have

$$\|\vec{\beta}^{bl}_{k,l}\|_{L^2(Z^0_l \setminus \overline{Z^0_m})^4} \le C e^{-\gamma^{lower}|m|} \tag{4.12}$$

with γ^{lower} from (3.29). *Furthermore, for $y \in Z^0_l$*

$$|\vec{\beta}^{bl}_{k,l}(y)| + |\nabla \vec{\beta}^{bl}_{k,l}(y)| + |\omega^{bl}_{k,l}(y_1, y_2)| \le C e^{-\gamma^{lower}|y_2|}\,. \tag{4.13}$$

From these estimates we obtain the following proposition.

PROPOSITION 4.2. *Let $\{\vec{\beta}^{bl}_{k,l}, \omega^{bl}_{k,l}\}$ be the solution of problem* (4.1)–(4.7). *Then, for every $\delta < 2\pi$, a constant C exists such that*

$$\|\nabla \vec{\beta}^{bl}_{k,l} - \nabla \vec{\beta}^{bl}\|_{L^2(Z^k_l)} \le C(e^{-\delta k} + e^{-\gamma^{lower}|l|})\,. \tag{4.14}$$

Proof. Let $\xi := \vec{\beta}^{bl} - \vec{\beta}^{bl}_{k,l}$ and $\omega := \omega^{bl} - \omega^{bl}_{k,l}$. Then (ξ, ω) is y_1-periodic, vanishes on $\cup_{k=1}^{l}(\partial Z^* - (0, k))$, and solves the Stokes equation in Z^k_l. Set

$$\hat{\xi} := \begin{cases} \xi\,, & l + 1 \le y_2, \\ \xi^{lower}\,, & l + 1 - d_* \le y_2 < l + 1, \\ 0\,, & l < y_2 < l + 1 - d_*, \end{cases} \tag{4.15}$$

where ξ^{lower} is derived from ξ in the same way as $\vec{\beta}^{lower}$ was derived from $\vec{\beta}^{bl}$ in Lemma 3.6. Testing the momentum equation with $\hat{\xi}$, we obtain

$$\int_{Z^k_{l+1}} |\nabla \xi|^2 + \int_{l+1-d_*}^{l+1} \int_0^1 \nabla \xi \nabla \xi^{lower} = \int_0^1 (\nabla \xi - \omega I)\vec{e}_2 \xi\, dy_1\, |_{y_2 = k}\,. \tag{4.16}$$

Now note that ω may be replaced by $\omega - c$ for an arbitrary $c \in \mathbb{R}$ due to $\int_{y_2 = \text{const}} \xi_2 = 0$. Then we can apply the exponential stabilization results for $\{\vec{\beta}^{bl}, \omega^{bl}\}$ and $\{\vec{\beta}^{bl}_{k,l}, \omega^{bl}_{k,l}\}$ from (3.16), (3.18), (4.10), and (4.11) to obtain (4.14). ∎

The approximation error between C_1^{bl} and $C_{1,k,l}^{bl}$ can then be estimated as follows.

COROLLARY 4.3. *For every $\delta < 2\pi$, there is a constant C such that*

$$\left|C_1^{bl} - C_{1,k,l}^{bl}\right| \le C(e^{-2\pi k} + e^{-\gamma^{lower}|l|})\,. \tag{4.17}$$

Proof. Note that

$$\left|C_1^{bl} - C_{1,k,l}^{bl}\right| = \left|\int_{Z_0} \beta_1^{bl} - (\beta_{k,l}^{bl})_1\right| \le \left\|\beta_1^{bl} - (\beta_{k,l}^{bl})_1\right\|_{L^2(Z_0)}, \tag{4.18}$$

which can be estimated as desired by using Poincaré's inequality on $Z_0 \cup Z_{-1}$ together with (4.14). □

In order to obtain estimates for the pressure difference $\omega_{k,l}^{bl} - \omega^{bl}$, we need the following result.

LEMMA 4.4. *For each $F \in L^2(Z_l^k)$ satisfying $\int_{Z_l^k} F = 0$, there is a function $\vec{\varphi} \in H^1(Z_l^k)$ vanishing on the boundaries $y_2 = k$, $y_2 = l$, and $\cup_{k=1}^{l}(\partial Z^* - (0,k))$ and satisfying* $\operatorname{div}\vec{\varphi} = F$ *together with the stability estimate*

$$\|\nabla\vec{\varphi}\|_{L^2(Z_l^k)} \le C(k+|l|)\|F\|_{L^2(Z_l^k)}\,. \tag{4.19}$$

More generally, if $F \in H^r(Z_l^k)$ for some integer $r \ge 0$, then $\vec{\varphi}$ can be chosen such that

$$\|\vec{\varphi}\|_{H^r(Z_l^k)} \le C(k+|l|)\|F\|_{H^{r-1}(Z_l^k)}\,. \tag{4.20}$$

Proof. The proof is similar to the proof of Lemma 3.4 in [9] and can be found in [11]. □

With the help of this lemma we obtain the following proposition.

PROPOSITION 4.5. *Let ω^{bl} and $\omega_{k,l}^{bl}$ be the pressure fields determined by* (3.2)–(3.7)*, respectively,* (4.1)–(4.7)*. Then, for every $\delta < 2\pi$, there is a constant C such that*

$$\left\|\omega^{bl} - \omega_{k,l}^{bl}\right\|_{L^2(Z_l^k)} \le C(k+|l|)(e^{-\delta k} + e^{-\gamma^{lower}|l|})\,. \tag{4.21}$$

For the difference $C_\omega^{bl} - C_{\omega,k,l}^{bl}$ we have the better estimate

$$\left|C_\omega^{bl} - C_{\omega,k,l}^{bl}\right| \le C\sqrt{|l|}(e^{-\delta k} + e^{-\gamma^{lower}|l|})\,. \tag{4.22}$$

Proof. Set $\bar{\omega} := \frac{1}{|Z_l^k|}\int_{Z_l^k}(\omega^{bl} - \omega_{k,l}^{bl})$, and let $F := \omega^{bl} - \omega_{k,l}^{bl} - \bar{\omega}$. Since $\int_{Z_l^k} F = 0$, Lemma 4.4 yields a function $\vec{\varphi}$ which we can use as a test function in the difference of momentum equations

$$-\triangle(\vec{\beta}^{bl} - \vec{\beta}_{k,l}^{bl}) + \nabla(\omega^{bl} - \omega_{k,l}^{bl}) = 0\,. \tag{4.23}$$

Inserting $\bar{\omega}$ in the pressure term, testing with $\vec{\varphi}$, and doing a partial integration, we obtain

$$\int_{Z_l^k} \nabla(\vec{\beta}^{bl} - \vec{\beta}_{k,l}^{bl})\nabla\vec{\varphi}\,dx = \int_{Z_l^k} F^2\,dx\,. \tag{4.24}$$

If we now use the stability estimate (4.19) together with (4.14), this yields

$$\left\|\omega_{k,l}^{bl} - \omega^{bl} - \bar{\omega}\right\|_{L^2(Z_l^k)} \le C(k+|l|)(e^{-2\pi k} + e^{-\gamma^{lower}|l|})\,. \tag{4.25}$$

Finally, to estimate $\bar{\omega}$, let $r_j := \int_{Z_j} \omega(y)$. Then we have $|\bar{\omega}| \le |r_l| + \sum_{j=l}^{-1} |r_{j+1} - r_j|$. By Theorem 3.7 of [9], we also have

$$|r_{j+1} - r_j| \le C \|\nabla(\vec{\beta}^{bl} - \vec{\beta}^{bl}_{k,l})\|_{L^2(Z_j^{j+2})}\,, \tag{4.26}$$

which can be used together with (4.13) to estimate

$$|\bar{\omega}| \le C e^{-\gamma^{lower}|l|} + C\sqrt{|l|}\|\nabla(\vec{\beta}^{bl} - \vec{\beta}^{bl}_{k,l})\|_{L^2(Z_l^1)}\,. \tag{4.27}$$

Using the triangle inequality in (4.25) together with (4.27) and (4.14), we obtain (4.21).

In order to get (4.22), we note that

$$\left|C^{bl}_{\omega} - C^{bl}_{\omega,k,l}\right| = |r_0| \le |r_l| + \sum_{j=l}^{-1} |r_{j+1} - r_j|\,, \tag{4.28}$$

such that the estimate follows directly from (4.27) and (4.14). ∎

We complete this section with a regularity result, which we will need in the following.

PROPOSITION 4.6. *Let $f \in L^2(Z_l^k)^2$, and let $\{\xi, \pi\}$ be the y_1-periodic solution of the nonhomogeneous Stokes system with right-hand side f in Z_l^k, boundary conditions*

$$\xi = 0 \quad \text{on } \{y_2 = l\} \cup \bigcup_{k=1}^{l} (\partial Z^* - (0,k))\,, \quad \xi_2 = \frac{\partial \xi_1}{\partial y_2} = 0 \quad \text{on } \{y_2 = k\}\,, \tag{4.29}$$

and pressure normalization $\int_{y_2=l} \pi \, dy = 0$. Then we have the estimate

$$\begin{aligned} \|\nabla\xi\|_{L^2(Z_l^k)} + \|\xi_2\|_{L^2(Z_l^k)} + \left\| \frac{\xi_1}{1 + \max\{y_2, 0\}} \right\|_{L^2(Z_l^k)} + \left\|D^2\xi\right\|_{L^2(Z_l^k)} + \|\nabla\pi\|_{L^2(Z_l^k)} \\ \le C(\|f_2\|_{L^2(Z_l^k)} + \|f_1(1 + \max\{y_2, 0\})\|_{L^2(Z_l^k)}) \end{aligned} \tag{4.30}$$

with a constant C independent of k and l.

Proof. The proof is rather technical but standard. For details, see [11]. ∎

COROLLARY 4.7. *Let $\{\vec{\beta}^{bl}_{k,l}, \omega^{bl}_{k,l}\}$ be the solution to* (4.1)–(4.7), *and let*

$$\tilde{\beta} := \vec{\beta}^{bl}_{k,l} - \vec{e_1} \max\left\{ y_2 - \frac{y_2^2}{2}, 0 \right\} \frac{(k-1)e^{-y_2} + e^{-k}(1 - 2k + \frac{k^2}{2})}{k - 1 + e^{-k}(1 - 2k + \frac{k^2}{2})}\,. \tag{4.31}$$

Then

$$\|D^2\tilde{\beta}\|_{L^2(Z_l^k)} + \left\|\nabla\omega^{bl}_{k,l}\right\|_{L^2(Z_l^k)} \le C\,, \tag{4.32}$$

where C can be chosen independent from k and l.

5. Discretization. Now we turn our attention to the discretization of problem (4.1)–(4.7). Essentially, we use a stabilized finite element discretization in the sense of [8], [3]. Unfortunately, in both of these papers only polygonal domains were considered, so the direct application of these results to our domain Z_l^k is unsatisfying because of the curved boundaries $\cup_{k=1}^{l}(\partial Z^* - (0,k))$. We resolve this difficulty by using generalized domain partitions given by nonlinear mappings of the elements while

essentially keeping the approximation results known for usual domain partitions with linear mappings; see [19], [20], [1], [15]. While this is very convenient theoretically, the practical implementation will usually be too complicated. Therefore, it is important that the use of a simpler polygonal approximation of the domain can be interpreted as a perturbation of this approach; see the discussion at the end of this section.

Let Ω be a Lipschitz domain, and let $\mathcal{T}_h$ be a partition of Ω in subsets e, called elements, where each element e is the image of a reference element $\hat{e}$ under a mapping $\Phi_e : \hat{e} \to e$, where $\hat{e}$ is either the reference triangle $\hat{T} := \{(x_1, x_2) \mid x_1 \geq 0, x_2 \geq 0, x_1 + x_2 \leq 1\}$ or the reference quadrangle $\hat{Q} := [0,1]^2$.

We require the following properties of the partition $\mathcal{T}_h$.

1. $\cup_{e \in \mathcal{T}_h} e = \bar{\Omega}$.
2. For two elements $e \neq e'$ we have $\mathrm{int}(e) \cap e' = \emptyset$.
3. A side of an element e (which is defined as the image of a side of $\hat{e}$) is either a subset of $\partial\Omega$ or the side of exactly one other element $e' \neq e$. In the second case, we require that the mapping $\Phi_{e'}^{-1} \circ \Phi_e$ restricted to the corresponding side of $\hat{e}$ is linear.
4. The $\Phi_e : \hat{e} \to e$ are bi-Lipschitzian mappings with
$$\|\nabla \Phi_e\|_\infty \leq C_1 \operatorname{diam}(e)\,, \quad \left\|\nabla \Phi_e^{-1}\right\|_\infty \leq C_1 \operatorname{diam}(e)^{-1} \tag{5.1}$$
for some constant $C_1 > 0$.
5. $\Phi_e \in W^{2,\infty}$, with $\|D^2 \Phi_e\|_\infty \leq C_2 \operatorname{diam}(e)^2$.
6. For simplicity reasons, we consider only quasi-uniform partitions, i.e., a constant $C_3 > 0$ exists such that for $h := \max_{e \in \mathcal{T}_h} \operatorname{diam}(e)$ we have $\operatorname{diam}(e) \geq C_3 h$ for all $e \in \mathcal{T}_h$.

Arbitrarily fine triangulations of this kind exist, which can be shown by modifying triangulations of polygonal approximations of the domain Ω; see [19]. The quality of the domain partition $\mathcal{T}_h$ is determined by the constants C_1, C_2, C_3. Uniform refinement of such a partition results in a new partition which still has the above properties (especially property 5).

Next let
$$\mathcal{FE}(\hat{e}) = \begin{cases} P_1 & \text{if} \quad e = \hat{T}, \\ Q_1 & \text{if} \quad e = \hat{Q}, \end{cases} \tag{5.2}$$
where P_1 is the space of linear polynomials, and Q_1 is the span of P_1 and the polynomial $x_1 x_2$. With these finite elements, we can define the space
$$\mathcal{S}_h := \{\psi \in H^1(\Omega) \mid \ \forall e : \psi \circ \Phi_e \in \mathcal{FE}(\hat{e})\}\,. \tag{5.3}$$

The following approximation result is then the substitute for approximation results on triangulations with linear element mappings.

THEOREM 5.1. *Let Ω, $\mathcal{T}_h$, and $\mathcal{S}_h$ be as above. Then, for $u \in H^1(\Omega)$, a $u_h \in \mathcal{S}_h$ exists such that*
$$\|u - u_h\|_{L^2(\Omega)} + h\|\nabla u_h\|_{L^2(\Omega)} \leq Ch\|\nabla u\|_{L^2(\Omega)}\,. \tag{5.4}$$
Second, for $u \in H^2(\Omega)$, a $u_h \in \mathcal{S}_h$ exists such that
$$\|u - u_h\|_{L^2(\Omega)} + h\|\nabla(u - u_h)\|_{L^2(\Omega)} \leq Ch^2(\|\nabla u\|_{L^2(\Omega)} + \left\|D^2 u\right\|_{L^2(\Omega)})\,. \tag{5.5}$$
Additionally, if u has zero boundary values on some component of $\partial\Omega$, then u_h can also be chosen such that it has zero boundary values on that boundary component.

In all cases, the constants depend only on the smoothness of the domain and the quality of the domain partition $\mathcal{T}_h$.

Proof. See [19], [20] for the case of (deformed) triangular elements, and see [15], where the case of (deformed) quadrilaterals is also handled by using a generalized interpolation operator of Clément type. ∎

REMARK 5.2. *In contrast to standard interpolation results for triangulations with linear element mappings, the term $\|\nabla u\|_{L^2(\Omega)}$ appears on the right-hand side of* (5.5). *This is due to the fact that linear functions are no more interpolated exactly (in contrast to constant functions). Note that this estimate is possible only because of the Φ_e approximating linear mappings in the limit* $\operatorname{diam}(e) \to 0$ *(see property* 5 *required for the partition $\mathcal{T}_h$).*

Now, let $\Omega = Z_l^k$, and let $\mathcal{T}_h$ be a domain partition of Z_l^k (fitting across the lateral boundaries, where periodic boundary conditions are prescribed for problem (3.2)–(3.7)). Then define $\mathcal{S}_h$ analogously to (5.3) as

$$\mathcal{S}_h := \{\psi \in H^1(Z_l^k) \mid \psi \text{ is } y_1 - \text{periodic } \wedge\ \forall e : \psi \circ \Phi_e \in \mathcal{FE}(\hat{e})\}\,, \tag{5.6}$$

and ansatz spaces for velocity and pressure

$$\vec{H}_h := \Bigg\{ \vec{\varphi}_h \in \mathcal{S}_h^2 \mid \vec{\varphi}_h = 0 \text{ on } \{y_2 = l\} \cup \bigcup_{k=1}^{l} (\partial Z^* - (0,k)),$$

$$(\vec{\varphi}_h)_2 = 0 \text{ on } \{y_2 = k\} \Bigg\}, \tag{5.7}$$

$$L_h := \left\{ \psi_h \in \mathcal{S}_h \mid \int_{y_2=l} \psi_h \, dy = 0 \right\}. \tag{5.8}$$

We now search for $(\vec{\beta}_{k,l,h}^{bl}, \omega_{k,l,h}^{bl}) \in \vec{H}_h \times L_h$, which is the solution to

$$\int_{Z_l^k} \nabla \vec{\beta}_{k,l,h}^{bl} \nabla \vec{\varphi}_{k,l,h} \, dy + \int_{Z_l^k} \nabla \omega_{k,l,h}^{bl} \vec{\varphi}_{k,l,h} \, dy = -\int_S \vec{\varphi}_{k,l,h} \cdot \vec{e}_1\,, \tag{5.9}$$

$$\int_{Z_l^k} \operatorname{div} \vec{\beta}_{k,l,h}^{bl} \chi_{k,l,h} + \alpha \sum_{Q \in \mathcal{T}_h} h_Q^2 \int_Q \nabla \omega_{k,l,h}^{bl} \nabla \chi_{k,l,h} \, dy = 0 \tag{5.10}$$

for all $(\vec{\varphi}_{k,l,h}, \chi_{k,l,h}) \in \vec{H}_h \times L_h$. The second term in (5.10) must be included for the above pair of ansatz spaces to stabilize the discretization; see [8], [3]. The constant α can be any positive number. The following error estimates then hold.

PROPOSITION 5.3. *Let $\mathcal{T}_h, \vec{H}_h, L_h$ be defined as above. Assume that the interior boundary $S = \{y_2 = 0\}$ is aligned with the sides of the elements of $\mathcal{T}_h$, let $(\vec{\beta}_{k,l}^{bl}, \omega_{k,l}^{bl})$ be the solution of problem* (4.1)–(4.7)*, and let $(\vec{\beta}_{k,l,h}^{bl}, \omega_{k,l,h}^{bl}) \in \vec{H}_h \times L_h$ be the solution of* (5.9)–(5.10). *Furthermore, let*

$$R(\vec{\beta}_{k,l}^{bl}, \omega_{k,l}^{bl}) = \|\nabla \vec{\beta}_{k,l}^{bl}\|_{L^2(Z_l^k)} + \|D^2 \vec{\beta}_{k,l}^{bl}\|_{L^2(Z_0^k)} + \|D^2 \vec{\beta}_{k,l}^{bl}\|_{L^2(Z_l^0)} + \|\nabla \omega_{k,l}^{bl}\|_{L^2(Z_l^k)}\,.$$

Then we have an error estimate in the viscous energy norm of the form

$$\|\nabla(\vec{\beta}_{k,l,h}^{bl} - \vec{\beta}_{k,l}^{bl})\|_{L^2(Z_l^k)} \le ChR(\vec{\beta}_{k,l}^{bl}, \omega_{k,l}^{bl})\,, \tag{5.11}$$

together with a stability estimate for the pressure gradient of the form

$$\left(\sum_Q h_Q^2 \big\|\nabla(\omega_{k,l,h}^{bl} - \omega_{k,l}^{bl})\big\|_{L^2(Q)}^2 \right)^{\frac{1}{2}} \le ChR(\vec{\beta}_{k,l}^{bl}, \omega_{k,l}^{bl})\,. \tag{5.12}$$

We further have the L^2-error estimate for the pressure

$$\left\| \omega^{bl}_{k,l,h} - \left(\omega^{bl}_{k,l} - \frac{1}{\left|Z^k_l\right|} \int_{Z^k_l} \omega^{bl}_{k,l}\, dy \right) \right\|_{L^2(Z^k_l)} \leq C(k+|l|)hR(\vec{\beta}^{bl}_{k,l}, \omega^{bl}_{k,l}) \tag{5.13}$$

and the velocity

$$\|\vec{\beta}^{bl}_{k,l,h} - \vec{\beta}^{bl}_{k,l}\|_{L^2(Z^k_l)} \leq C(k+1)h^2 R(\vec{\beta}^{bl}_{k,l}, \omega^{bl}_{k,l}) \tag{5.14}$$

with a constant C which is independent of k and l.

Proof. The proof can be done as in [8], [3], or [7] with the following modifications. First, on our more general domain partitions, the approximation results from Theorem 5.1 replacing the standard ones. Second, [8], [3], [7] handle only the Dirichlet case. However, one can easily check that their proofs can be transferred with almost no changes because the slip boundary condition still allows for partial integration with vanishing boundary terms. For (5.13), the proof uses the assertion of Lemma 4.4 such that the factor $k+|l|$ needs to be included in the estimate. For (5.14), one needs the H^2-regularity estimate from Proposition 4.6 which introduces the factor $k+1$. ∎

We now describe the definition of the discrete approximations to $C^{bl}_{1,k,l}$ and $C^{bl}_{\omega,k,l}$. First, we set

$$C^{bl}_{1,k,l,h} := \int_S (\beta^{bl}_{k,l,h})_1 \,. \tag{5.15}$$

Second, in order to obtain good error estimates for the numerical approximation of $C^{bl}_{\omega,k,l}$, we define $C^{bl}_{\omega,k,l,h}$ as a smoothly weighted average of the pressure field in the following way. Let $\tilde{\lambda} : [0,1] \to [0,1]$ be a Lipschitz-continuous function satisfying $\lambda(0) = 0$, $\lambda(1) = 1$, and $\int_0^1 \lambda(x)\, dx = 1$. Then define $\tilde{\lambda} : Z^k_l \to \mathbb{R}$ as

$$\tilde{\lambda}(y_1, y_2) = \begin{cases} \lambda(y_2 - (k-1)) & \text{for} \quad y_2 \geq k-1, \\ -\left(\int_{Z_{-1}} \lambda(-y_2)\, dy \right)^{-1} \lambda(l+1-y_2) & \text{for} \quad y_2 \leq l+1, \\ 0 & \text{elsewhere.} \end{cases} \tag{5.16}$$

It is obvious that $\int_{Z_k} \tilde{\lambda}(y)\, dy = -\int_{Z_l} \tilde{\lambda}(y)\, dy = 1$, such that we may approximate $C^{bl}_{\omega,k,l}$ by

$$C^{bl}_{\omega,k,l,h} := \int_{Z^k_l} \tilde{\lambda}(y)\, \omega^{bl}_{k,l,h}(y)\, dy \,. \tag{5.17}$$

Then we can prove the following proposition.

PROPOSITION 5.4. *For $C^{bl}_{1,k,l,h}, C^{bl}_{\omega,k,l,h}$ given by* (5.15), (5.17) *we have the estimates*

$$\left| C^{bl}_{1,k,l,h} - C^{bl}_{1,k,l} \right| \leq C(k+1)h^2 \,, \tag{5.18}$$

$$\left| C^{bl}_{\omega,k,l,h} - C^{bl}_{\omega,k,l} \right| \leq C(k+1)(k+|l|)h^2 + Ce^{-\gamma^{lower}|l|} \,. \tag{5.19}$$

Proof. In order to show (5.18), we use the definitions (4.8) and (5.15) in the momentum equations (4.1) and (5.9) to obtain

$$C^{bl}_{1,k,l,h} - C^{bl}_{1,k,l} = \int_S (\vec{\beta}^{bl}_{k,l,h} - \vec{\beta}^{bl}_{k,l}) \vec{e_1}\, ds$$
$$= \int_{Z^k_l} \nabla \vec{\beta}^{bl}_{k,l} \nabla (\vec{\beta}^{bl}_{k,l} - \vec{\beta}^{bl}_{k,l,h})\, dx + \int_{Z^k_l} \nabla \omega^{bl}_{k,l} (\vec{\beta}^{bl}_{k,l} - \vec{\beta}^{bl}_{k,l,h})\, dx$$
$$= \int_{Z^k_l} |\nabla (\vec{\beta}^{bl}_{k,l} - \vec{\beta}^{bl}_{k,l,h})|^2\, dx - \int_{Z^k_l} \nabla (\omega^{bl}_{k,l} - \omega^{bl}_{k,l,h}) \vec{\beta}^{bl}_{k,l,h}\, dx + \int_{Z^k_l} \nabla \omega^{bl}_{k,l} (\vec{\beta}^{bl}_{k,l} - \vec{\beta}^{bl}_{k,l,h})\, dx\,.$$

Here, the first term is of order $O(h^2)$ by (5.11), the second term can be estimated as

$$\int_{Z^k_l} \nabla (\omega^{bl}_{k,l} - \omega^{bl}_{k,l,h}) \vec{\beta}^{bl}_{k,l,h}\, dx = \int_{Z^k_l} \nabla (\omega^{bl}_{k,l} - \omega^{bl}_{k,l,h}) (\vec{\beta}^{bl}_{k,l,h} - \vec{\beta}^{bl}_{k,l})\, dx = O((k+1)h^2)$$

by (5.14), and, if one applies (5.10), the third term can be estimated as

$$\int_{Z^k_l} \nabla \omega^{bl}_{k,l} (\vec{\beta}^{bl}_{k,l} - \vec{\beta}^{bl}_{k,l,h})\, dx = \int_{Z^k_l} \nabla (\omega^{bl}_{k,l} - \omega^{bl}_{k,l,h}) (\vec{\beta}^{bl}_{k,l} - \vec{\beta}^{bl}_{k,l,h}) + \alpha \sum_{Q \in \mathcal{T}_h} h_Q^2 \int_Q \left| \nabla \omega^{bl}_{k,l,h} \right|^2.$$

Next, if we use (5.14) and (5.12), we can see that the resulting terms are of order $O((k+1)h^2)$ and $O(h^2)$, respectively. Thus we have shown (5.18).

In order to show the estimate (5.19), we first note that

$$C^{bl}_{\omega,k,l} - C^{bl}_{\omega,k,l,h} = \int_{Z^k_l} (\omega^{bl}_{k,l}(y) - \omega^{bl}_{k,l,h}(y)) \tilde{\lambda}(y)\, dy + O(e^{-\gamma^{lower}|l|}) \tag{5.20}$$

by (4.13), (4.9), and (5.17). Next let $\vec{\varphi} \in H^2(Z^k_l)$ be the vector field given by Lemma 4.4, which satisfies $\operatorname{div} \vec{\varphi} = \tilde{\lambda}$, $\vec{\varphi} = 0$ on ∂Z^k_l, and

$$\left\| D^2 \vec{\varphi} \right\|_{L^2(Z^k_l)} \leq C(k + |l|)\,. \tag{5.21}$$

Furthermore, Theorem 5.1 yields the existence of an interpolating vector field $\vec{\varphi}_h \in \vec{H}_h$ which fulfills

$$\|\vec{\varphi} - \vec{\varphi}_h\|_{L^2(Z^k_l)} + h \|\nabla (\vec{\varphi} - \vec{\varphi}_h)\|_{L^2(Z^k_l)} \leq C h^2 \left\| D^2 \vec{\varphi} \right\|_{L^2(Z^k_l)}. \tag{5.22}$$

By using this, we can estimate the first term from the right-hand side of (5.20) as follows:

$$\int_{Z^k_l} (\omega^{bl}_{k,l} - \omega^{bl}_{k,l,h})\, \tilde{\lambda}\, dy = \int_{Z^k_l} (\omega^{bl}_{k,l} - \omega^{bl}_{k,l,h}) \operatorname{div} \vec{\varphi}\, dy = \int_{Z^k_l} \nabla (\omega^{bl}_{k,l} - \omega^{bl}_{k,l,h})\, \vec{\varphi}\, dy$$
$$= \int_{Z^k_l} \nabla (\omega^{bl}_{k,l} - \omega^{bl}_{k,l,h})\, (\vec{\varphi} - \vec{\varphi}_h)\, dy - \int_{Z^k_l} \nabla (\vec{\beta}^{bl}_{k,l} - \vec{\beta}^{bl}_{k,l,h})\, \nabla \vec{\varphi}_h\, dy\,.$$

Here, the first term is of order $C(k+|l|)h^2$ by (5.12), (5.22), and (5.21). The second term can be estimated by replacing first $\vec{\varphi}_h$ by $\vec{\varphi}$, which can be done up to an error of order $C(k+|l|)h^2$ due to (5.11), (5.22), and (5.21). For the rest, a partial integration yields

$$\int_{Z^k_l} \nabla (\vec{\beta}^{bl}_{k,l} - \vec{\beta}^{bl}_{k,l,h}) \nabla \vec{\varphi}\, dy = \int_{Z^k_l} (\vec{\beta}^{bl}_{k,l} - \vec{\beta}^{bl}_{k,l,h}) \Delta \vec{\varphi}\, dy \leq C(k+1)(k+|l|)h^2 \tag{5.23}$$

if one uses (5.14) and (5.21). Thus, (5.19) is also proved. ∎

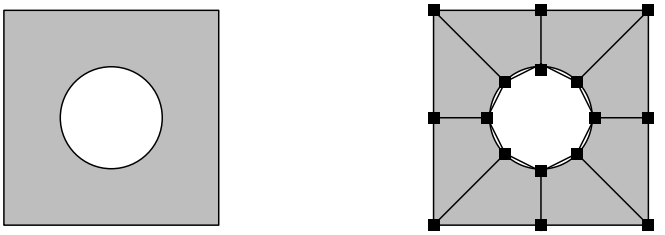


FIG. 6.1. *Symmetric cell and its coarsest grid.*

TABLE 6.1
Results for the symmetric cell ($k = l = 2$).

j	$C^{bl}_{1,k,l,h}(j)$	$C^{bl}_{1,k,l,h}(j) - C^{bl}_{1,k,l,h}(j-1)$
0	-0.6083663	—
1	-0.6134767	$-5.11041129 \cdot 10^{-3}$
2	-0.6096270	$3.8496901 \cdot 10^{-3}$
3	-0.6081967	$1.4302327 \cdot 10^{-3}$
4	-0.6077899	$4.0679752 \cdot 10^{-4}$
5	-0.6076814	$1.0854489 \cdot 10^{-4}$
∞	$-0.6076417 \pm 5.0 \cdot 10^{-8}$	

As mentioned at the beginning of this section, the practical implementation of domain partitions with nonlinear element mappings is rather complicated. It is usually easier to approximate a domain Ω with curved boundaries by polygonal domains Ω_h, which then can be partitioned in triangles and/or quadrangles (see, for example, [18]). However, it can be shown that, on a discrete level, this simpler approach is equivalent to our theoretically more convenient formulation up to a quadrature error of optimal order; see [16].

6. Numerical results. We are now ready to demonstrate our method for computing the constants C^{bl}_1 and C^{bl}_ω on specific examples. First, we consider the symmetric geometry shown on the left part of Figure 6.1. The boundary ∂Z^* is a circle with radius 0.25 and center at $(0.5, 0.5)$. On the right-hand side of Figure 6.1, the initial polygonal approximation and the initial grid $\mathcal{T}_{h_0}$ with quadrangle elements (see section 5) is depicted. $\mathcal{T}_{h_0}$ is then uniformly refined, yielding further grids $\mathcal{T}_{h_1}, \ldots, \mathcal{T}_{h_J}$. By using the discretization described in the previous section, we obtain for every $\mathcal{T}_{h_i}, i = 1, \ldots, J$, a system of linear equations which must be solved to obtain the discrete solution $\{\vec{\beta}^{bl}_{k,l,h}, \omega^{bl}_{k,l,h}\}$.

Since the arising linear systems are very large, we have applied the multigrid method, which is known to be of optimal complexity for a large range of problems; see [6]. However, due to the pressure stabilization and the polygonal approximation of the smooth boundary $\cup^l_{k=1}(\partial Z^* - (0, k))$, we are not in a Galerkin setting. In this case, it is known that the simple multigrid V-cycle (one coarse-grid correction between pre- and postsmoothing) does not have to converge independently of the number of levels. We therefore used the W-cycle (two coarse-grid corrections between smoothing). Both pre- and postsmoothing are done by two steps of a block incomplete decomposition, where each block contains the unknowns of one grid node (the corners of the elements). And, indeed, our numerical observations confirm that this method is robust with respect to the number of levels and variations of the parameters k, l; see also Table 6.5 below.

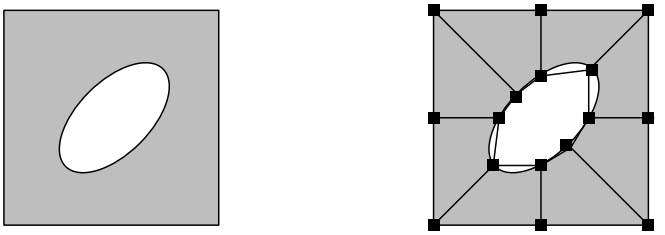


FIG. 6.2. *Unsymmetric cell and coarsest grid.*

TABLE 6.2
Results for the unsymmetric cell ($k = l = 2$).

j	$C^{bl}_{1,k,l,h}(j)$	$C^{bl}_{1,k,l,h}(j) - C^{bl}_{1,k,l,h}(j-1)$	$C^{bl}_{\omega,k,l,h}(j)$	$C^{bl}_{\omega,k,l,h}(j) - C^{bl}_{\omega,k,l,h}(j-1)$
0	-0.552980	—	-0.192913	—
1	-0.542858	$1.012 \cdot 10^{-2}$	-0.437585	$-2.446 \cdot 10^{-1}$
2	-0.529998	$1.286 \cdot 10^{-2}$	-0.459770	$-2.205 \cdot 10^{-2}$
3	-0.525892	$4.106 \cdot 10^{-3}$	-0.449916	$9.880 \cdot 10^{-3}$
4	-0.524860	$1.032 \cdot 10^{-3}$	-0.447415	$2.507 \cdot 10^{-3}$
5	-0.524602	$2.584 \cdot 10^{-4}$	-0.446836	$5.798 \cdot 10^{-4}$
∞	$-0.524501 \pm 4.9 \cdot 10^{-6}$		$-0.4464 \pm 1.4 \cdot 10^{-4}$	

In Table 6.1, the results for a computation with $k = l = 2$ are shown. Starting from the coarsest level, which contains 20 elements, we refine five times, which yields a grid with 20480 elements (61440 unknowns). On each level $j = 0, \dots, 5$, we solve the discretized equation and compute the approximations $C^{bl}_{1,k,l,h}$ and $C^{bl}_{\omega,k,l,h}$, given by (5.15) and (5.17) (with the choice $\lambda(x) = x$ in (5.16)). The value given for $j = \infty$ is computed by polynomial extrapolation. We see that both $C^{bl}_{1,k,l,h}$ and $C^{bl}_{\omega,k,l,h}$ converge to limit values with rate $O(h^2)$, which we expected from Proposition 5.3. As shown in Remark 3.9, C^{bl}_{ω} must be zero. Since our grid is symmetric, the solution of the discrete problem also has the same symmetry property, such that the approximations $C^{bl}_{\omega,k,l,h}$ are zero up to machine accuracy.

TABLE 6.3
$C^{bl}_{1,k,l}$ (extrapolated) for varying k, l.

$l \setminus k$	1	2	3
1	-0.524500	-0.524500	-0.524500
2	-0.524500	-0.524501	-0.524501
3	-0.524500	-0.524501	-0.524501

Let us now assume that the porous part is generated by the unsymmetric cell from Figure 6.2, where the boundary curve is given by the ellipse

$$[0, 2\pi] \ni \phi \mapsto (0.5, 0.5) + \frac{0.25}{1.0 + 0.3\cos(2\phi + 1.5708)}(\cos\phi, \sin\phi)\,. \tag{6.1}$$

For this domain we obtain the results shown in Table 6.2. Both $C^{bl}_{1,k,l,h}$ and $C^{bl}_{\omega,k,l,h}$ converge again with order $O(h^2)$, which is in accordance with Proposition 5.3. The error arising from the cutting of the domain is not noticeable any more already for $k = l = 2$. This is shown in Tables 6.3 and 6.4, where the results for varying k and l are given. Note that even the values for $k = l = 1$ are accurate up to the extrapolation error.

TABLE 6.4
$C^{bl}_{\omega,k,l}$ *(extrapolated) for varying* k, l.

$l \setminus k$	1	2	3
1	-0.3750	-0.3750	-0.3750
2	-0.4464	-0.4464	-0.4464
3	-0.4464	-0.4464	-0.4464

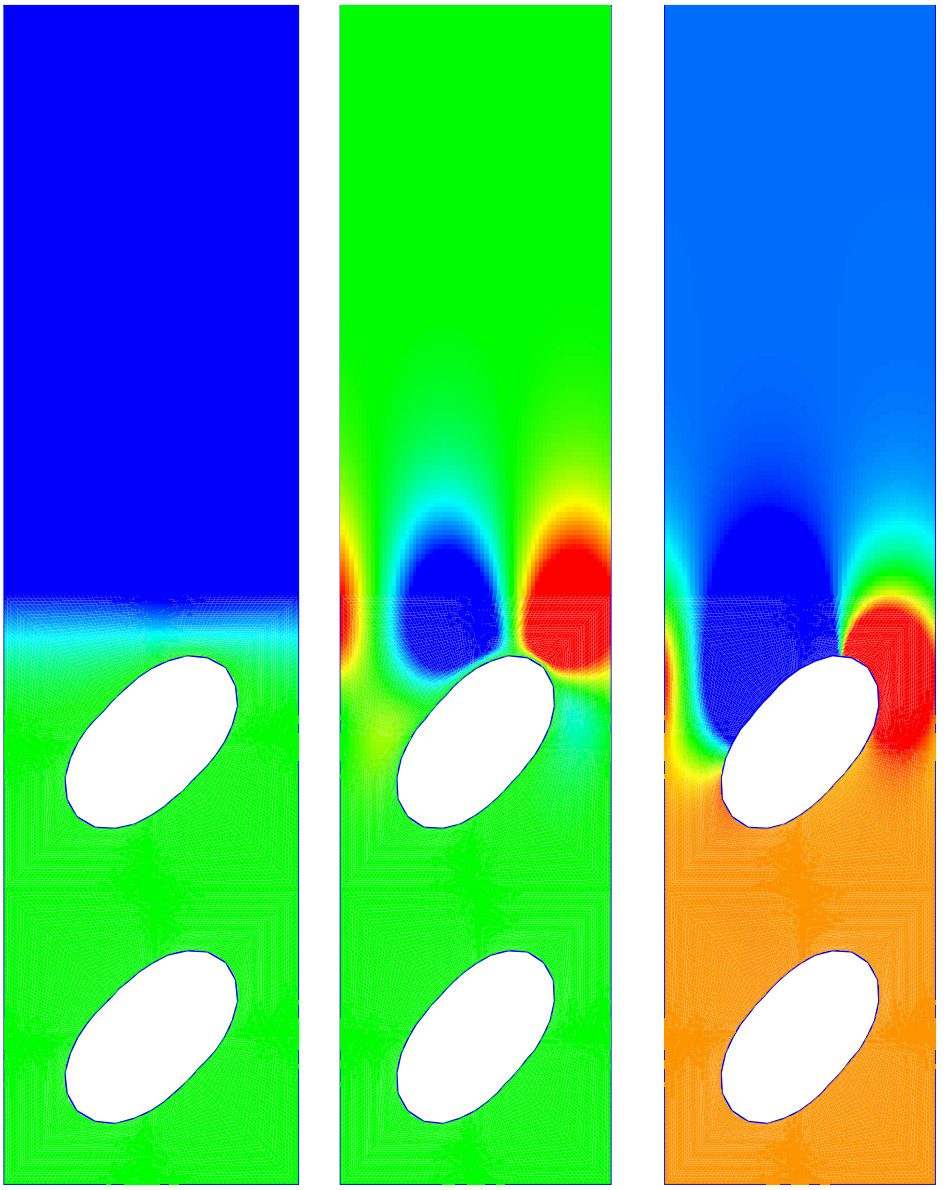


FIG. 6.3. *Detail from a picture showing* β_1^{bl}, β_2^{bl}, *and* ω^{bl}.

TABLE 6.5
Multigrid convergence rates.

$k,l \setminus j$	1	2	3	4	5
$k,l=1$	0.093	0.144	0.193	0.168	0.176
$k,l=2$	0.209	0.152	0.194	0.168	0.176
$k,l=3$	0.247	0.168	0.194	0.168	0.176

Figure 6.3 shows the three solution components β_1^{bl}, β_2^{bl}, and ω^{bl}. From here and from the values of C_ω^{bl} in Table 6.2, it is obvious that a pressure jump occurs inside the boundary layer.

Finally, Table 6.5 shows the convergence rates of our multigrid iteration. As we expected from the discussion above, it is perfectly robust with respect to the number of levels. So far, we did not observe a significant dependency on k or l, even if one might expect some deterioration, since the error estimates from Proposition 5.3 are needed in the usual W-cycle convergence theory. The reason might be that sharper error estimates are possible in suitably weighted norms.

7. The effective equations. It turned out that $\vec{\beta}^{bl}$ stabilizes exponentially fast to a constant vector $(C_1^{bl},0)$ for $y_2 \to +\infty$ (and to 0 for $y_2 \to -\infty$), which translates into $\vec{\beta}^{bl,\varepsilon}$ tending to $\varepsilon(C_1^{bl},0)$ for $x_2 \to +\infty$. This forces us to consider the corresponding counter-flow in the channel, which is described by the following two-dimensional (2D) Oseen–Couette system in Ω_1:

$$-\mu\triangle\vec{d} + \nabla g + v_1^0 \frac{\partial \vec{d}}{\partial x_1} + d_2 \frac{\partial v_1^0}{\partial x_2}\vec{e}_1 = 0 \quad \text{in } \Omega_1\,, \tag{7.1}$$

$$\operatorname{div} \vec{d} = 0 \quad \text{in } \Omega_1\,, \tag{7.2}$$

$$\vec{d} = \vec{e}_1 \text{ on } \Sigma, \qquad \vec{d} = 0 \text{ on } (0,b)\times\{H\}, \tag{7.3}$$

$$d_2 = 0\,,\ g = 0 \text{ on } (\{0\}\cup\{b\})\times(0,H)\,. \tag{7.4}$$

If we assume that $\mathrm{Re} := \frac{1}{8\mu^2}|p_b - p_0|H^2$ is not too big, the problem (7.1)–(7.4) has a unique solution in the form of 2D Couette flow $\vec{d} = (1-\frac{x_2}{H})\vec{e}_1$ and $g = 0$.

Following the ideas from [9], we write down the correction of order ε for the velocity. Essentially, it corresponds to the elimination of the tangential component of the normal stress at Σ, caused by the approximation by Poiseuille's flow and its contribution to the energy estimate. The correction reads

$$\vec{u}^\varepsilon = \vec{v}^0 H(x_2) - \varepsilon\vec{\beta}^{bl}\left(\frac{x}{\varepsilon}\right)\frac{\partial v_1^0}{\partial x_2} + \varepsilon C_1^{bl}\left(\frac{\partial v_1^0}{\partial x_2}\vec{e}_1 + \vec{d}\right)H(x_2) + O(\varepsilon^2). \tag{7.5}$$

The correction of the velocity field by the oscillatory boundary layer velocity $\vec{\beta}^{bl,\varepsilon}$ involves the introduction of the boundary layer pressure field $\omega^{bl,\varepsilon}$. Hence, as usual for flow problems, it is necessary to correct the pressure field simultaneously. The corresponding pressure correction reads

$$p^\varepsilon = p^0 H(x_2) + p^{1,\varepsilon} H(-x_2) - \big(\omega^{bl,\varepsilon} - H(x_2)C_\omega^{bl}\big)\mu\frac{\partial v_1^0}{\partial x_2}(0) + o(1). \tag{7.6}$$

Here, $p^{1,\varepsilon}$ is an appropriate regularization of the effective pressure p in the porous bed, defined by

$$\operatorname{div}\big(K\nabla p\big) \quad = \quad 0 \qquad \text{in } \Omega_2, \tag{7.7}$$

$$p(x_1,-0) = p^0(x_1) + \mu C_\omega^{bl}\frac{\partial v_1^0}{\partial x_2}(0) \quad \text{on } \Sigma, \tag{7.8}$$

$$(K\nabla p)\cdot e_2 = 0 \quad \text{on } (0,b)\times\{-L\}, \tag{7.9}$$

$$p = p_0 \quad \text{on } \{0\}\times(-L,0), \quad \text{and} \quad p = p_b \quad \text{on } \{b\}\times(-L,0), \tag{7.10}$$

where the permeability tensor K is defined as $K_{ij} = \int_{Y^*} w_i^j(y)\,dy$. Here, the w_i^j are solutions of the auxiliary problems

$$-\triangle_y \vec{w}^j + \nabla_y \pi^j = \vec{e}_j \quad \text{in } Y^*, \tag{7.11}$$

$$\operatorname{div}_y \vec{w}^j = 0 \quad \text{in } Y^*, \tag{7.12}$$

$$\vec{w}^j = 0 \quad \text{on } \partial Z^*, \quad \{\vec{w}^j, \pi^j\} \text{ is } Z\text{-periodic}, \quad \int_{Y^*}\pi^j = 0\,. \tag{7.13}$$

The pressure field p is a C^∞-function outside the corners. However, due to the discontinuities of the traces at $(0,0)$ and at $(b,0)$, its gradient is not square integrable in Ω_2. We must regularize the values at the upper corners, and $p^{1,\varepsilon}$ is such a regularization, satisfying $p = p^{1,\varepsilon} + O(\sqrt{\varepsilon})$.

Let us now introduce the difference $\mathcal{U}_0^\varepsilon$ between the velocity field $\vec{u}^\varepsilon$ and its expansion up to order $O(\varepsilon)$, i.e., we set

$$\vec{\mathcal{U}}_0^\varepsilon(x) = \vec{u}^\varepsilon - \vec{v}^0 + \vec{\beta}^{bl,\varepsilon}\frac{\partial v_1^0}{\partial x_2}(0) - \varepsilon C_1^{bl}\frac{\partial v_1^0}{\partial x_2}(0)H(x_2)\frac{x_2}{H}\vec{e}_1. \tag{7.14}$$

Then, after constructing the corresponding outer boundary layer, it was proved in [10] that

$$\int_0^b\int_{-L}^0 |\vec{\mathcal{U}}_0^\varepsilon(x)|^2\,dx + \varepsilon^2\int_0^b\int_{-L}^H |\nabla\vec{\mathcal{U}}_0^\varepsilon(x)|^2\,dx \le C\varepsilon^4|\log\varepsilon|^2\,, \tag{7.15}$$

$$\int_0^b |\vec{\mathcal{U}}_0^\varepsilon(x_1,0)|^2\,dx_1 + \int_0^b\int_0^H |\vec{\mathcal{U}}_0^\varepsilon(x)|^2\,dx \le C\varepsilon^3|\log\varepsilon|^2\,. \tag{7.16}$$

It should be noted that the presence of the logarithmic term is a consequence of the corner singularities in the effective pressure. It was proved in [9] that in the absence of the boundary singularities, the above estimates hold without the $\log\varepsilon$ term. Therefore, in the interior of the domain Ω, the expansion (7.5) is of order $O(\varepsilon^{3/2})$. Globally, it is of order $O(\varepsilon^{3/2}|\log\varepsilon|)$.

The estimates (7.15)–(7.16) are sufficient for calculating the next order correction at the interface Σ. The estimate (7.16) gives us a possibility of approximating the velocity values at Σ by an oscillatory velocity field. Computationally, it is not very useful.

In view of the problem setting in [9], the Beavers–Joseph law corresponds to taking into the account the next order corrections for the velocity. In fact, we formally get on the interface Σ

$$\frac{\partial u_1^\varepsilon}{\partial x_2} = \frac{\partial v_1^0}{\partial x_2}(0)\left(1 - \frac{\partial\beta_1^{bl}}{\partial y_2}\left(\frac{x}{\varepsilon}\right)\right) + \frac{\partial\mathcal{U}_{01}^\varepsilon}{\partial x_2} + O(\varepsilon)\,, \tag{7.17}$$

$$\frac{1}{\varepsilon}u_1^\varepsilon = -\beta_1^{bl}\left(\frac{x}{\varepsilon}\right)\frac{\partial v_1^0}{\partial x_2}(0) + O(\sqrt{\varepsilon}|\log\varepsilon|)\,, \tag{7.18}$$

$$\begin{aligned} u_1^\varepsilon + \varepsilon C_1^{bl}\frac{\partial u_1^\varepsilon}{\partial x_2} = \mathcal{U}_{01}^\varepsilon + \varepsilon C_1^{bl}\frac{\partial\mathcal{U}_{01}^\varepsilon}{\partial x_2} + \frac{\partial v_1^0}{\partial x_2}(0)\left(\varepsilon C_1^{bl} - \beta_1^{bl}\left(\frac{x}{\varepsilon}\right)\right) \\ -\varepsilon C_1^{bl}\frac{\partial v_1^0}{\partial x_2}(0)\frac{\partial\beta_1^{bl}}{\partial y_2}\left(\frac{x}{\varepsilon}\right) + O(\varepsilon^2)\,. \end{aligned} \tag{7.19}$$

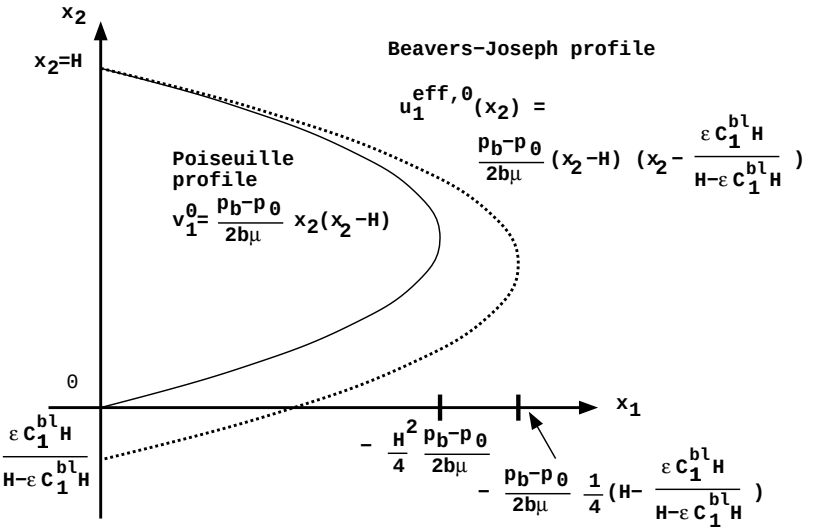


FIG. 7.1. *Beavers–Joseph versus Poiseuille profile.*

Integrals of the absolute value of the right-hand side are not small, since our estimates do not give any pointwise control of ∇u on Σ. Nevertheless, the right-hand side of (7.19) is small in the appropriate Sobolev norm of a negative order. The precise estimates are in [10]. Hence we get the effective law

$$u_1^\varepsilon + \varepsilon C_1^{bl} \frac{\partial u_1^\varepsilon}{\partial x_2} = 0 \quad \text{on } \Sigma, \tag{7.20}$$

which is exactly Saffman's modification of the Beavers–Joseph law from (1.2) with $\frac{1}{\alpha} K^{1/2} = -\varepsilon C_1^{bl}$.

Let us introduce the effective flow equations in Ω_1 through the following boundary value problem. Find a velocity field $\vec{u}^{\,eff}$ and a pressure field p^{eff} such that

$$-\mu \triangle \vec{u}^{\,eff} + (\vec{u}^{\,eff} \nabla)\vec{u}^{\,eff} + \nabla p^{eff} = 0 \quad \text{in } \Omega_1, \tag{7.21}$$

$$\operatorname{div} \vec{u}^{\,eff} = 0 \quad \text{in } \Omega_1, \tag{7.22}$$

$$\vec{u}^{\,eff} = 0 \quad \text{on } (0,b) \times \{H\}, \tag{7.23}$$

$$u_2^{eff} = 0 \quad \text{on } (\{0\} \cup \{b\}) \times (0,H), \tag{7.24}$$

$$p^{eff} = p_0 \quad \text{on } \{0\} \times (0,H) \text{ and } p^{eff} = p_b \quad \text{on } \{b\} \times (0,H), \tag{7.25}$$

$$u_2^{eff} = 0 \text{ and } u_1^{eff} + \varepsilon C_1^{bl} \frac{\partial u_1^{eff}}{\partial x_2} = 0 \text{ on } \Sigma. \tag{7.26}$$

Under the same assumption of laminarity as for problem (7.1)–(7.4), the problem (7.21)–(7.26) has a unique solution $\{\vec{u}^{\,eff}, p^{eff}\}$ with $p^{eff} = p^0$ and

$$\vec{u}^{\,eff} = \left(\frac{p_b - p_0}{2b\mu} \left(x_2 - \frac{\varepsilon C_1^{bl} H}{H - \varepsilon C_1^{bl}} \right) (x_2 - H), 0 \right) \quad \text{for } 0 \le x_2 \le H \tag{7.27}$$

(see Figure 7.1). The effective mass flow rate through the channel is

$$M^{eff} = b \int_0^H u_1^{eff}(x_2)\, dx_2 = -\frac{p_b - p_0}{12\mu} H^3 \frac{H - 4\varepsilon C_1^{bl}}{H - \varepsilon C_1^{bl}}\,. \tag{7.28}$$

By using the theory of very weak solutions for the Stokes system, the following approximation properties of $\{\vec{u}^{eff}, p^{eff}, M^{eff}\}$ are proved in [10]:

$$\int_0^H \int_0^b \left|p^\varepsilon - p^{eff}\right|^2 dx + \varepsilon \int_0^H \int_0^b \left|\nabla(\vec{u}^\varepsilon - \vec{u}^{eff})\right| dx \le C\varepsilon^2 |\log \varepsilon|^2, \tag{7.29}$$

$$\int_0^H \int_0^b \left|\vec{u}^\varepsilon - \vec{u}^{eff}\right|^2 dx + \int_0^H \int_0^b \left|M^\varepsilon - M^{eff}\right|^2 dx \le C\varepsilon^3 |\log \varepsilon|^2. \tag{7.30}$$

The estimates (7.29)–(7.30) justify Saffman's modification of the law by Beavers and Joseph. Furthermore, we are able to calculate their proportionality factor; it is equal to $-K^{1/2}/C_1^{bl}$. We note that (7.26) is not the only possible interface law. If one replaces the Beavers–Joseph law by $u_1^{eff} = \varepsilon C_1^{bl} \frac{\partial v_1^0}{\partial x_2}(0)$, the estimates (7.29)–(7.30) remain valid. However, such a condition involves the knowledge of the zeroth order approximation v_1^0 and is not really an interface condition.

At this stage, we can consider the approximation to the channel flow as satisfactory, but we still must determine the filtration velocity and the pressure in the porous medium. We have already mentioned the influence of corner singularities on the solution of the problem (7.8)–(7.10). In order to avoid the discussion of such effects, we limit ourselves to the behavior in the interior of the porous medium. The inertia effects are negligible, and we can use the theory from [9, Theorem 3]. In the interior, all boundary layer effects are exponentially small, and we have

$$\vec{u}^\varepsilon = -\varepsilon^2 \sum_{j=1}^{2} \vec{w}^j \left(\frac{x}{\varepsilon}\right) \frac{\partial p(x)}{\partial x_j} + O(\varepsilon^3), \tag{7.31}$$

where $\vec{w}^j$, $j = 1, 2$, are defined by (7.11). Hence the filtration velocity is given through the Darcy law $\vec{v}^f(x) = -K\nabla p(x)$ in Ω_2.

Far away from the corners, the pressure field p approximates p^ε at order $O(\sqrt{\varepsilon})$. It can be determined only after we have found the effective pressure field in the channel and the stabilization constant C_ω^{bl} giving the pressure difference at infinities for the auxiliary problem. As shown in section 6, this stabilization constant is generally different from zero, and we must use the interface law

$$p^{eff}(x_1, -0) = p^{eff}(x_1, +0) + \mu C_\omega^{bl} \frac{\partial v_1^0}{\partial x_2}(0) \qquad \text{on } \Sigma. \tag{7.32}$$

This shows that, contrary to intuition, *the effective pressure in the system channel flow/porous medium is not always continuous*. Thus the continuity assumption for the effective pressure from [12] is not correct in general.

Finally, let us note that we have some freedom in choosing the position of the interface. It could be set at any distance of order $O(\varepsilon)$ from the solid obstacles. Intuitively, the law of Beavers and Joseph should be independent of such a choice. This invariance result can be established rigorously, and we refer for details to [14], where it was proved that a perturbation of order $O(\varepsilon)$ in the position of the interface leads to a change of the constant C_1^{bl}, but $\vec{u}^{eff}$ changes only at order $O(\varepsilon^2)$.

REFERENCES

[1] I. Babuska, G. Caloz, and J. E. Osborn, *Special finite element methods for a class of second order elliptic problems with rough coefficients*, SIAM J. Numer. Anal., 31 (1994), pp. 945–981.

[2] G. S. Beavers and D. D. Joseph, *Boundary conditions at a naturally permeable wall*, J. Fluid Mech., 30 (1967), pp. 197–207.
[3] F. Brezzi and J. Pitkäranta, *On the stabilization of finite element approximations of the Stokes equation*, in Efficient Solution of Elliptic Systems, Notes Numer. Fluid Mech. 10, Viewig, Braunschweig, 1984, pp. 11–19.
[4] R. E. Caflisch and J. Rubinstein, *Lectures on the Mathematical Theory of Multiphase Flow*, Lecture notes, Courant Institute of Mathematical Sciences, New York, 1984.
[5] H. I. Ene and E. Sanchez-Palencia, *Equations et phénomènes de surface pour l'écoulement dans un modèle de milieu poreux*, J. Mécan., 14 (1975), pp. 73–108.
[6] W. Hackbusch, *Multigrid Methods and Applications*, Springer Ser. Comput. Math. 4, Springer–Verlag, Berlin, 1985.
[7] J. Harig, *Eine robuste und effiziente Finite Elemente Methode zur Lösung der inkompressiblen 3-D Navier-Stokes-Gleichungen auf Vektorrechnern*, Ph.D. thesis, Universität Heidelberg, Heidelberg, Germany, 1991.
[8] T. J. R. Hughes, L. P. Franca, and M. Balestra, *A new finite element formulation for computational fluid dynamics:* V. *Circumventing the Babuška-Brezzi condition*, Comp. Methods Appl. Mech. Engrg., 59 (1986), pp. 85–99.
[9] W. Jäger and A. Mikelić, *On the boundary conditions at the contact interface between a porous medium and a free fluid*, Ann. Scuola Norm. Sup. Pisa Cl. Sci., 23 (1996), pp. 403–465.
[10] W. Jäger and A. Mikelić, *On the interface boundary condition of Beavers, Joseph, and Saffman*, SIAM J. Appl. Math., 60 (2000), pp. 1111–1127.
[11] W. Jäger, A. Mikelić, and N. Neuss, *Asymptotic Analysis of the Laminar Viscous Flow over a Porous Bed*, Tech. rep., Universität Heidelberg, Heidelberg, Germany, 1999. IWR-Preprint 1999-33, http://www.iwr.uni-heidelberg.de/sfb.
[12] T. Levy and E. Sanchez-Palencia, *On boundary conditions for fluid flow in porous media*, Internat. J. Engrg. Sci., 13 (1975), pp. 923–940.
[13] J.-L. Lions, *Some Methods in the Mathematical Analysis of Systems and Their Control*, Gordon and Breach, New York, 1981.
[14] A. Mikelić, *Homogenization theory and applications to filtration through porous media*, in Filtration in Porous Media and Industrial Applications, Lecture Notes in Math. 1734, M. S. Espedal, A. Fasano, and A. Mikelić, eds., Springer-Verlag, Berlin, 2000, pp. 127–214.
[15] N. Neuss, *Homogenisierung und Mehrgitter*, Ph.D. thesis, Universität Heidelberg, Heidelberg, Germany, 1995. Also as ICA-Report N96/7, ICA Stuttgart, Germany, 1996.
[16] N. Neuss and C. Wieners, *Criteria for the Approximation Property for Multigrid Methods in Nonnested Spaces*, Tech. rep. SFB 359, Universität Heidelberg, Heidelberg, Germany, 2000; also available online from http://www.iwr.uni-heidelberg.de/sfb.
[17] P. G. Saffman, *On the boundary condition at the interface of a porous medium*, Stud. Appl. Math., 1 (1971), pp. 93–101.
[18] R. Verfürth, *Finite element approximation of incompressible Navier-Stokes equations with slip boundary condition*, Numer. Math., 50 (1987), pp. 697–721.
[19] M. Zlamal, *Curved elements in the finite element method.* I, SIAM J. Numer. Anal., 10 (1973), pp. 229–240.
[20] M. Zlamal, *Curved elements in the finite element method.* II, SIAM J. Numer. Anal., 11 (1974), pp. 347–362.

SIAM J. SCI. COMPUT.
Vol. 22, No. 6, pp. 2029–2050
© 2001 Society for Industrial and Applied Mathematics

SPECIAL BILINEAR QUADRILATERAL ELEMENTS FOR LOCALLY REFINED FINITE ELEMENT GRIDS*

WEIGANG WANG†

Abstract. A set of special bilinear quadrilateral elements are developed to handle irregular nodes on the interface between different refinement levels in 1-irregular meshes for the adaptive finite element method. With this alternative approach, irregular nodes are eliminated, and at the same time the C^0 continuity at interelements is strictly ensured. The new elements simplify the refinement procedure and circumvent many inconveniences in local refinement with linear quadrilateral elements. Solutions to problems of a heat conduction, a reaction-diffusion, and incompressible and viscous flows past a circular cylinder are respectively presented to test the effectiveness of special elements. They demonstrate good accuracy and a potential applicability to more complex problems.

Key words. special linear quadrilateral finite elements, local refinement, irregular node, Navier–Stokes equations

AMS subject classifications. 65N30, 65M50

PII. S1064827599358911

1. Introduction.

Adaptive finite element methods (AFEM) with quadrilateral elements pose certain special difficulties in handling the elements close to the interface between different refinement levels. Specifically, the local refinement procedure results in irregular nodes ("hanging nodes," "virtual nodes," or "slave nodes" in some literature) at the interface. Improper handling of these irregular nodes causes a discontinuous solution at the interface, thereby degrading the accuracy of the whole discretized system.

Different approaches have been proposed to handle these irregular nodes. A short review of those methods can be found in [20]. There exist three different approaches, i.e., converting irregular nodes to regular ones by dividing some additional elements [13, 15, 17], constructing additional constraint equations [6], and reconstructing interpolation functions [1, 9, 16]. The approach of dividing additional elements often results in a discretized system with quadrilateral elements or a combination of quadrilateral and triangular elements. In both situations, the resulting systems are full of elements with undesirable aspect ratios which often decrease the accuracy locally or globally. The additionally divided elements often cause inconvenience (from a programming point of view) in the further refinement process. The method with additional constraint equations is often a complicated procedure and is difficult to use directly in existing FEM codes. The approach of reconstructing interpolation functions is often carried out under certain restrictive conditions which are not directly extended to complex cases.

However, the different approaches share a common goal, namely, to ensure the continuity of unknowns (represented by interpolation functions) across the interelements, whatever elements are used. In addition to this so-called C^0 continuity, another important requirement is that the methods should be easy to use and should be able to handle complex cases.

*Received by the editors July 20, 1999; accepted for publication (in revised form) September 20, 2000; published electronically March 20, 2001.
http://www.siam.org/journals/sisc/22-6/35891.html
†Institut für Thermo- und Fluiddynamik, Ruhr-Universität Bochum, D-44780 Bochum, Germany (wang@lstm.ruhr-uni-bochum.de).

In the following context, we use the notions "interelements" and "interface between elements" interchangeably unless stated otherwise. In fact, both indicate the interior border between elements.

The interpolation functions for the FEM are mostly constructed in a piecewise manner, the C^0 continuity at interelements being inevitably determined (not entirely, though) by the behavior of the interpolation functions applied there. In the present paper, we propose an alternative concept for the local refinement with bilinear quadrilateral elements in 1-irregular meshes. Specifically, a set of special elements and their interpolation functions are constructed in order to keep the C^0 continuity at the interelements and to simplify the refinement process.

These specially designed elements possess some unique features. They *geometrically* do not satisfy one of the basic requirements for a finite element, i.e., $\cap\Omega^e = \emptyset$, where Ω^e, $e = 1, 2, \ldots, n$ are the finite elements in a bounded region. They, in fact, have geometric overlaps if we consider the geometric border of an element as defined by the element's own nodes. However, considering the contribution of every element to the weighted residual equations, the classic definition is strictly satisfied. This *forward* (*delayed* might be more precise) enforcement of the basic requirement simplifies the local refinement procedure with linear quadrilateral elements. In fact, these special elements with their corresponding special interpolation functions make the C^0 continuity at the interelements a relatively easier task and circumvent many inconveniences in the local refinement process.

Similar to normal bilinear elements, these special elements are self-contained. The special elements simply ignore the existing irregular nodes and internally build up on their own (by the interpolation functions). Both the aforementioned features allow the existing FEM software to easily adopt the concept with minimum effort involved. The special elements also can be used in the cases with distorted elements without losing the C^0 continuity at the interelement with irregular nodes. Furthermore, it is possible to extend the concept to the discretization for triangular elements and for 3-dimensional cases.

The concept of constructing special bilinear quadrilateral elements is, in fact, a generalization of a similar idea proposed in a companion paper about special *quadratic* quadrilateral elements [20]. Interested readers are referred to that paper for additional details. The *forward* enforcement of the basic requirement for a finite element is used in both situations. However, the interface location, the geometric parameters, and the corresponding C^0 continuity condition for bilinear quadrilateral elements are more complex and need more rigorous verifications, which the emphasis of the present paper is focused on.

2. Local refinement with special elements. Shown in Figure 2.1 (a) is a typical situation where the elements at two different levels of refinement meet at interface *amb*. A big element 1 and two small elements 2 and 3 are drawn separately for clarity in Figure 2.1 (b). Element 1 has four regular corner nodes and has no more nodes on side *ab*, while elements 2 and 3 both have three regular corner nodes and another corner node m somewhere on interface *ab*. From a discretization point of view, node m is an unusual node because it only depends on elements 2 and 3 but never directly on element 1 even though node m falls geometrically into element 1's domain. Node m is the so-called irregular node. A similar situation also appears in the unstructured meshes for finite difference or finite volume methods.

In the following, we introduce a new concept to handle the problem caused by the irregular nodes in local refinement with bilinear quadrilateral elements. The

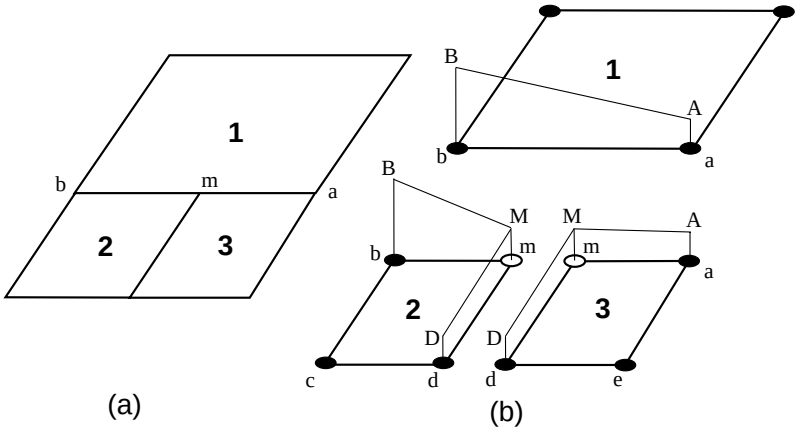


FIG. 2.1. *Bilinear quadrilateral elements in transient region of a* 1*-irregular mesh. The numbers indicate elements, the lower-case letters denote the nodes, and the upper-case letters denote the nodal values, while thick lines mark the element borders and thin lines show unknown's value represented by interpolation functions; the regular nodes are marked by solid circles and the irregular nodes by void circles.*

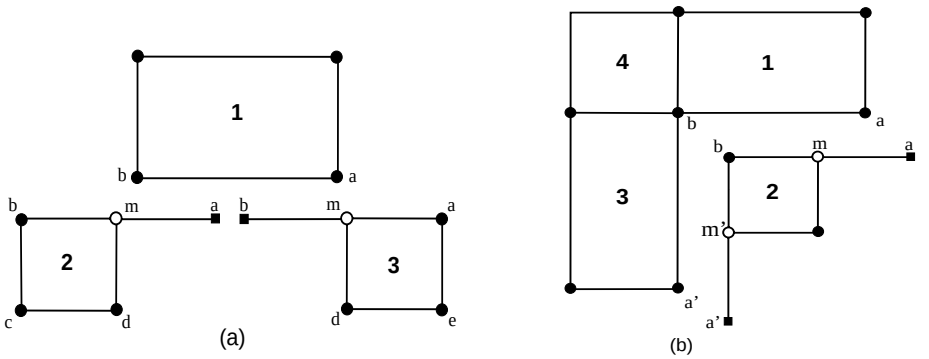


FIG. 2.2. *Special bilinear quadrilateral elements of the first* (2) *and the second* (3) *kind in* (a)*, and the third kind* (2) *in* (b)*. A solid square indicates an irregular node's replacement node.*

basic idea is to maintain the C^0 continuity at interfaces by eliminating the irregular nodes completely. For the small element with an irregular node on the interface, e.g., element 2 in Figure 2.2 (a), this can be readily achieved by *creating a new interpolation function without the irregular node but with a corresponding extra regular node*. Specifically, we construct special bilinear interpolation functions for element 2 based on nodes b, c, d, and a. Node a is a replacement for irregular node m and is taken from the big element at the other side of the interface ab (this replacement node can also be considered as taken from neighboring small element 3); see Figure 2.2 (a). Similarly, the special bilinear interpolation functions for element 3 can be constructed based on nodes d, e, a, and b. Subsequently, elements 2 and 3 are called special elements of the first and the second kind, respectively.

In reality, there may appear another kind of special element. This element appears normally at corner-like interfaces between different refinement levels. Element 2, shown in Figure 2.2 (b), is the special element of this third kind. Comparing special element 2 of the third kind in Figure 2.2 (b) with those of the first and second kind in Figure 2.2 (a), we can clearly recognize that the special element of the third kind is, in fact, a geometrical combination of the first and the second kind.

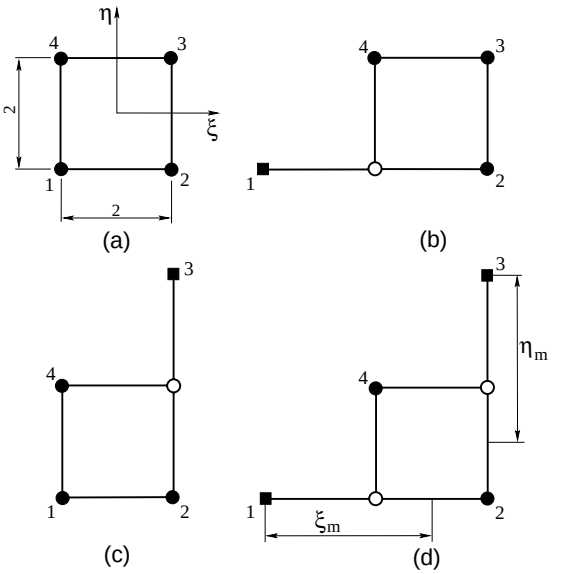


FIG. 2.3. *Normal* (a) *and special bilinear quadrilateral elements of the first* (b)*, the second* (c)*, and the third* (d) *kind in natural coordinates. The numbers beside solid circles and squares denote the node numbers.*

Analyzing special elements 2 and 3 in Figure 2.2 (a), we can see that each has four nodes, and thus the bilinear interpolation functions can be constructed in quite a normal way. Geometrically, elements 2 and 3 are mirror images of each other. Based on the concept explained above, irregular node m does not appear in either element 2 or element 3, and therefore the geometrical border for a special element is not directly defined by its four nodes. If we would define the border of the special elements only by their nodes, an overlapping area ($\triangle abd$) would appear between special elements 2 and 3.

Next we define an interface in this overlapping triangle, so that the overlapping area will be counted once and only once into the finite element equations, and at the same time the condition of the C^0 continuity is strictly held at such an interface. This can be achieved by *forwarding* the enforcement of the basic requirement for subdivision with finite elements to a later stage, i.e., at the stage of constructing interpolation functions. In this way, the C^0 continuity and the basic requirement for the finite elements will both be strictly satisfied. This *forward* enforcement of the basic requirement for an element forms the very base of the interpolation functions constructed below.

Collecting the normal as well as the special elements for bilinear quadrilateral elements, the four different elements are shown in Figure 2.3. In the following, we first list the interpolation functions for the special elements. A more rigorous analysis and an explanation about the C^0 continuity will be given in the next section.

The special interpolation functions for the special elements and the normal elements can be constructed in a systematic way, and the following general form can be obtained:

$$
\begin{aligned}
\Phi_1 &= \frac{(1-\xi)(1-\eta)}{2(1-\xi_m)},\\
\Phi_2 &= -\frac{1+\xi_m-\eta_m+3\xi_m\eta_m-(1+\xi_m+3\eta_m-\xi_m\eta_m)\xi}{4(1-\xi_m)(1+\eta_m)}\\
&\quad -\frac{(1-3\xi_m-\eta_m-\xi_m\eta_m)\eta+(3-\xi_m+\eta_m+\xi_m\eta_m)\xi\eta}{4(1-\xi_m)(1+\eta_m)},\\
\Phi_3 &= \frac{(1+\xi)(1+\eta)}{2(1+\eta_m)},\\
\Phi_4 &= \frac{(1-\xi)(1+\eta)}{4},
\end{aligned}
\tag{2.1}
$$

where parameters ξ_m and η_m indicate the different kinds of bilinear elements used and their values also depend on the degree of irregularity of the elements. Parameters ξ_m and η_m are, respectively, the two coordinates of irregular nodes' replacement nodes; see Figures 2.3 (a) and (d).

As an example, we take a case with rectangular elements. The normal and special rectangular elements are generated only by dividing another rectangular element equally into four small elements and vice versa. In the case of $\xi_m = -1$ and $\eta_m = 1$, the interpolation functions of a normal 4-node element (see Figure 2.3 (a)) can be obtained. If we set $\xi_m = -3$ and $\eta_m = 1$, a special element of the first kind results; see Figure 2.3 (b). The interpolation functions for the elements of the second and the third kind can be derived in a similar way by choosing different values of ξ_m and η_m. Some possible values of ξ_m and η_m for simple cases are listed in Table 2.1 for the normal and the three special elements.

TABLE 2.1
Parameters ξ_m and η_m for normal and special bilinear quadrilateral elements in simple cases.

Element type	In Figure 2.3	ξ_m	η_m
Normal	(a)	-1	1
1st	(b)	-3	1
2nd	(c)	-1	3
3rd	(d)	-3	3

By a simple analysis based on the property of bilinear functions, the C^0 continuity at the interface between one big and two small elements (e.g., *ab* in Figure 2.1) is strictly ensured. However, the C^0 continuity at the interface between the first and the second special elements is not so obvious. In fact, where the interface should be located depends on parameters ξ_m and η_m. More geometrical implications of the two parameters will be explained in the next section. Here we only state that, for the cases with rectangular elements where we generate the new elements for the next refinement level by bisecting equally the big element's sides, the parameters in Table 2.1 will always result in the right interfaces between elements as well as the corresponding C^0 continuity. Furthermore, the interpolation functions in (2.1) with the parameters in Table 2.1 can accommodate also slightly distorted quadrilateral elements without deteriorating the accuracy of the system significantly.

In some situations, the elements at a finer refinement level might not be created by halving the elements at the coarse refinement level, for instance, in the case of a nonrectangular domain, at places close to a moving interface in multiphase flows, in a moving boundary, or some other cases where general quadrilateral elements are desired. The C^0 continuity condition at the interface between two corresponding spe-

cial elements might not be satisfied for such general cases by choosing parameters ξ_m and η_m from Table 2.1.

In the next section, we demonstrate, in a more rigorous manner, the C^0 continuity at the interface between the two related special elements and the corresponding conditions. The proper values for parameters ξ_m and η_m will be derived thereafter for more general cases.

3. C^0 continuity of special elements. Since the special elements of the third kind can be obtained virtually by combining the first and the second kind, for the most part of the present section, we describe in detail the C^0 continuity analysis of the special elements of the first and the second kind because they possess the essential feature; later in this section the special element of the third kind will be used to derive the parameters ξ_m and η_m since other special elements can be derived from the third kind.

Shown in Figure 3.1 (a) are the two special elements in their surrounding area, while the enlarged version of elements 2 and 3 is illustrated in Figure 3.1 (b). The interface between the bigger element 1 and the two smaller elements 2 and 3 connects a coarse and a fine refinement level in this part of the 1-irregular mesh. As can

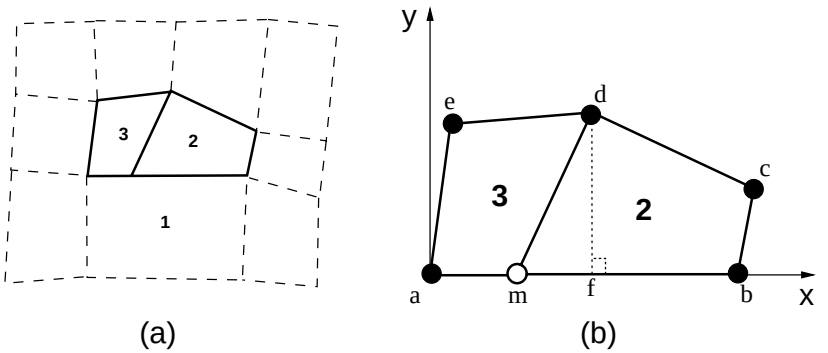


FIG. 3.1. *Special bilinear quadrilateral elements of the first and the second kind. The numbers indicate the elements, and the letters indicate the nodes; solid lines indicate the elements of the first and the second kind, and dashed lines indicate the surrounding elements; the dotted line represents the projecting path for node d to its projecting point f on interface ab.*

be seen, both elements 2 and 3 are quite arbitrary in shape in order to cover general cases. Without losing generality, we have moved the two elements to a new coordinate system by rotating and shifting only. Specifically, node a is moved to the origin, and interface ab is rotated to coincide with x-axis; see Figure 3.1 (b).

Let the nodal values for elements 2 and 3 be u_a, u_b, u_c, u_d, and u_e. We construct two bilinear interpolation functions using nodal coordinates and nodal values for elements 2 and 3, respectively. These interpolation functions form two bilinear surfaces over the two elements in 2-dimensional space. The behaviors of the two interpolation functions over the triangular area $\triangle abd$ (where all potential interfaces between elements 2 and 3 could be located) will be the focus of the following analysis.

The bilinear interpolation function ψ_{ij} for each node j in element i can be constructed in the general form

$$\psi_{ij} = \alpha_{ij} + \beta_{ij}x + \gamma_{ij}y + \delta_{ij}xy, \tag{3.1}$$

where x and y are the coordinates, and α_{ij}, β_{ij}, γ_{ij}, and δ_{ij} represent the polynomial coefficients determined by corresponding nodal coordinates. As explained in section 2,

we construct the special elements and the special interpolation functions by choosing nodes $a, b, c,$ and d for element 2 and nodes $a, b, d,$ and e for element 3. The expressions for the interpolation functions are quite lengthy. In the following, only the interpolation functions for element 3 are listed:

$$\psi_{3a} = 1 - \frac{x}{x_b} - \frac{x\,y(-x_b y_d + x_e y_d + x_b y_e - x_d y_e)}{x_b(x_d - x_e)y_d y_e} - \frac{y(x_b x_d y_d - x_d x_e y_d - x_b x_e y_e + x_d x_e y_e)}{x_b(x_d - x_e)y_d y_e}, \tag{3.2}$$

$$\psi_{3b} = \frac{x\,x_e y\,y_d - x_d x_e y\,y_d - x\,x_d y y_e + x_d x_e y\,y_e + x\,x_d y_d y_e - x\,x_e y_d y_e}{x_b(x_d - x_e)y_d y_e}, \tag{3.3}$$

$$\psi_{3d} = \frac{(x - x_e)y}{(x_d - x_e)y_d}, \tag{3.4}$$

$$\psi_{3e} = \frac{(x - x_d)y}{(x_e - x_d)y_e}. \tag{3.5}$$

The element-wise interpolation functions for these two elements therefore can be expressed in the following form:

$$\Psi_2 = \sum_{j=a,b,c,d} \psi_{2j} u_j, \tag{3.6}$$

$$\Psi_3 = \sum_{j=a,b,d,e} \psi_{3j} u_j, \tag{3.7}$$

where u_j denotes the nodal value at node j, and Ψ_2 and Ψ_3 indicate the interpolation functions for elements 2 and 3, respectively.

To prove the C^0 continuity, we analyze the behaviors of Ψ_2 and Ψ_3 at interface dm, a segment of straight line starting from node d and ending at irregular node m which can be anywhere on interface ab. Interface dm can be generally expressed as

$$x = x_m + \frac{x_d - x_m}{y_d} y. \tag{3.8}$$

If we substitute (3.6) and (3.7) with (3.8) and extract the essential terms, the difference between the two interpolation functions at interface dm can be written as

$$\Psi_3 - \Psi_2 = \frac{(x_d - x_m)y(y - y_d)}{x_b(x_c - x_d)(x_d - x_e)y_c y_d^2 y_e} F(u_j, x_j), \tag{3.9}$$

where $F(u_j, x_j)$ is a lengthy sum of many terms like $u_b x_b x_c y_c y_d, u_e x_b x_c y_c y_d$, and so on.

From (3.9), we can now make several important conclusions and obtain the answers to all uncertain aspects raised earlier:

1. At interface dm, the difference expressed by the two interpolation functions from special elements 2 and 3 is not always zero; in other words, the C^0 continuity does not always hold at any interface lying inside the overlapping area ($\triangle abd$); the discontinuity, however, never takes place at the interface's two ends of $y = y_a (= 0)$ and $y = y_d$; furthermore, a gap or an overlap between the two elements also occurs in case of the discontinuity if elements 2 and 3 are isoparametric ones.

2. The C^0 continuity is strictly satisfied (that is, $\Psi_3 - \Psi_2 = 0$) if interface dm is perpendicular to the interface between the normal element and the two special elements, specifically, $x_m = x_d (= x_f)$, for the case in Figure 3.1 (b).

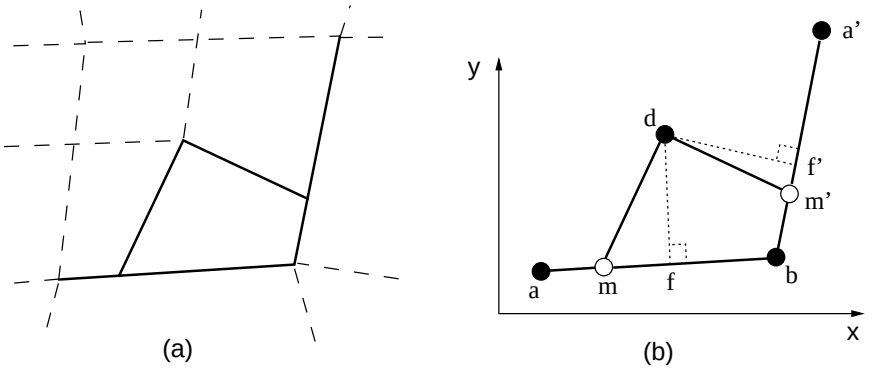


FIG. 3.2. *Special bilinear element of the third kind in the transient region. The solid lines indicate the element of the third kind, and the dashed lines indicate the surrounding elements; the dotted lines represent the projecting path for node d.*

3. Under some extreme circumstances, for instance, $x_b = x_a$, $x_c = x_d$, $x_d = x_e$, $y_c = y_b$, $y_d = y_m$ or $y_e = y_a$, (3.9) might become singular (it also depends on $F(u_j, x_j)$, see (3.9); in fact, one or more quadrilateral elements would degrade into a triangle, a line, or even a single point if such unusual element distortions occur; furthermore, another case with distorted elements might occur when node d's projecting point f falls outside interface ab; for these highly distorted elements, the C^0 continuity cannot be ensured.

The essential task in applying the special elements now reduces to that of keeping the interface between the two special elements (always in pairs for 1-irregular meshes) at the finer refinement level perpendicular to the corresponding interface connecting the finer and the coarser refinement levels. The implicit prerequisite is naturally that the projecting point m must fall in between on the related interface. If not, the elements involved must have been severely distorted.

Since the information at the irregular node is replaced by some linear relations through the corresponding regular nodes, the irregular node will not appear in the finite element equations at all. Therefore, the *right* interface between the corresponding special elements will be implicitly merged into the special interpolation functions. Only after achieving that, the special elements can satisfy the basic requirement for the finite elements:

$$\cup\overline{\Omega^e} = \Omega \text{ and } \Omega^e = \emptyset,$$

where Ω^e, $e = 1, 2, \ldots, n$ are element subdomains of a discretization of a bounded region Ω in $\mathbb{R}^d$ where n is the number of elements and d is the number of space dimensions.

In the following, we use the condition for the *right* interface to derive parameters ξ_m and η_m for (2.1). Shown in Figure 3.2 are a bilinear element of the third kind and its nearby region in a part of the 1-irregular mesh close to an interface where two different refinement levels meet. Based on the aforementioned perpendicularity condition for the corresponding two interfaces, the projecting point of node d to interface ab can be derived as

$$x_f = -\frac{((x_a - x_b)(x_a - x_b) + (y_a - y_b)^2)(-x_b y_a + x_a y_b)}{((x_a - x_b)(x_a - x_b) + (y_a - y_b)^2)(y_a - y_b)} + \frac{(-x_a + x_b)(x_a - x_b)(x_b y_a - x_a y_b)}{((x_a - x_b)(x_a - x_b) + (y_a - y_b)^2)(y_a - y_b)} \tag{3.10}$$

$$
\begin{aligned}
y_f = &-\frac{(-x_a+x_b)(y_a-y_b)(-x_ax_d+x_bx_d-y_ay_d+y_by_d)}{((x_a-x_b)(x_a-x_b)+(y_a-y_b)^2)(y_a-y_b)},\\
&-\frac{(x_a-x_b)(x_by_a-x_ay_b)}{(x_a-x_b)(x_a-x_b)+(y_a-y_b)^2}\\
&-\frac{(y_a-y_b)(-x_ax_d+x_bx_d-y_ay_d+y_by_d)}{(x_a-x_b)(x_a-x_b)+(y_a-y_b)^2}.
\end{aligned}
\tag{3.11}
$$

According to the coordinates of projecting point f and nodes a and b, the ratio of af to fb can be obtained as

$$
\frac{af}{fb}=\frac{(x_a-x_b)(x_a-x_d)+(y_a-y_b)(y_a-y_d)}{(x_a-x_b)(-x_b+x_d)+(y_a-y_b)(-y_b+y_d)}.
\tag{3.12}
$$

Considering the natural coordinate system and node numbers used in Figure 2.3, parameter ξ_m therefore can be written as

$$
\begin{aligned}
\xi_m &= -1-2\left|\frac{af}{fb}\right|\\
&= -1-2\left|\frac{(x_1-x_2)(x_1-x_4)+(y_1-y_2)(y_1-y_4)}{(x_1-x_2)(x_2-x_4)+(y_1-y_2)(y_2-y_4)}\right|,
\end{aligned}
\tag{3.13}
$$

where x_j and y_j denote the physical coordinates and j is the corresponding node number used in interpolation functions in natural coordinates (see Figure 2.3 as well as (2.1)). The similar procedure can be adopted to obtain parameter η_m, which can be subsequently written as

$$
\begin{aligned}
\eta_m &= 1+2\left|\frac{a'f'}{f'b}\right|\\
&= 1+2\left|\frac{(x_3-x_2)(x_3-x_4)+(y_3-y_2)(y_3-y_4)}{(x_3-x_2)(x_2-x_4)+(y_3-y_2)(y_2-y_4)}\right|.
\end{aligned}
\tag{3.14}
$$

It can be seen from (3.13) and (3.14) that parameters ξ_m and η_m depend only on the geometrical information of the elements. Furthermore, special elements appear always in pairs in the 1-irregular mesh, for instance, one first kind with one second kind, one first kind with one third kind, and so on. Therefore, parameter ξ_m of a special element of the first kind and parameter η_m of its counterpart (e.g., the second kind) are not independent but are related to each other by the relation of $\eta_m = 1 - 4/(\xi_m + 1)$. Note that this relation is correct only for special elements of the first and second kind (or the first and third kind), where $\xi_m \neq -1$. There exist similar relations between other special elements as well.

Though the C^0 continuity exists at the interface, the first-order derivative across the interface is normally discontinuous, which is a general feature of non-Hermite elements. However, the magnitude of this discontinuity indicates, indirectly at least, part of the discretization error in the system [20]. The error indicator based on this discontinuity will be used to perform the local refinement for the test cases in section 4.

The special elements can be quite systematically incorporated into AFEM. Element dividing in 1-irregular meshes can start in general from normal elements since special elements appear only at the finer-element side of the interface between different refinement levels. Element merging, on the other hand, can start from special as well as normal elements. However, the merging process often alters some neighboring

elements into special elements, and therefore some additional operation is needed to make some new special elements in the surrounding area. The remaining part of the refinement procedure, for instance, the interpolation for the new nodes, the update of data structure, and so on, is more or less the same as any other refinement procedure in AFEM.

Since special elements are self-contained as any normal bilinear quadrilateral element and do not require any information about their irregular nodes, they can be used conveniently and flexibly in a wide range of implementations. For instance, they can be adopted into other currently available FEM codes with minimal modification.

If we only consider the actual domain of a special element on which the local weighted residual equations later actually depend, the special element has kept approximately the original geometric aspect ratio. Though it does not indicate a perfect subdivision with exactly the same aspect ratio because the replacement node to an irregular one is often away from the actual domain, the special elements do keep, to certain extent, this ratio in an aforementioned way. This feature is always desirable, and it potentially improves the accuracy in the region close to the interface between different refinement levels.

Compared to the normal linear quadrilateral elements, only a small amount of extra computing effort is required to calculate parameters ξ_m and η_m from (3.13) and (3.14) for the interpolation functions in (2.1). Other possible extra resources might include a small amount of memory to store the element type, or this information can be determined based on the neighbors of an element, which in any case is needed for an unstructured mesh in AFEM.

4. Numerical test cases. In this section, the numerical test cases are presented. Specifically, the three cases are a heat transfer problem with a Laplace equation, a model combustion problem, and a viscous flow past a circular cylinder with incompressible Navier–Stokes equations. All test cases have been calculated with a research code FESTAL v.2.1 (Finite Element System for Turbulent and Laminar Combustion) which is still in development by the author. Standard Galerkin (SG) and streamline upwind/Petrov–Galerkin (SUPG) approaches have been implemented. In the following, only the SG approach is used. We skip the formulation of the FEM, and the interested readers should refer to FEM literature for details (for example, [5, 18]).

For Navier–Stokes equations, the method of successive substitution [18] or the SIMPLE algorithm [14] is used to solve the nonlinear system. The AFEM with the special elements introduced in section 2 is used for the local refinement of all test cases. Furthermore, use is made of the error indicator [20] based on the discontinuity of the first-order derivative at interelements. Later in this section we introduce another indicator based on the flow acceleration to capture the localized phenomena such as those in the vortex region.

First, we solve a 2-dimensional heat conduction problem [10] which can be stated as

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} = 0, \tag{4.1}$$

subject to the boundary conditions

$$\begin{aligned} T &= T_1 && \text{at } x = \pm W/2 \quad \text{or} \quad y = 0, \\ T &= T_m \cos\frac{\pi x}{W} + T_1 && \text{at } y = H, \end{aligned}$$

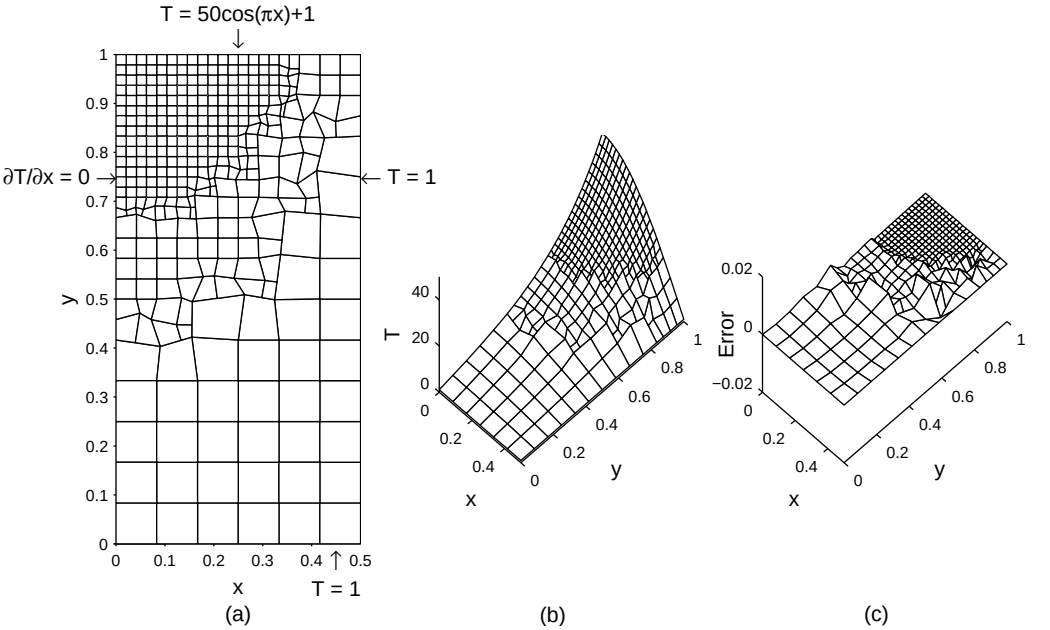


FIG. 4.1. *Mesh and boundary conditions* (a)*, solution* (b) *and its estimated numerical error* (c) *are calculated with special and normal elements for a heat conduction problem. The distortion of the elements close to the interface of different refinement levels are arbitrarily introduced. The* 1*-irregular mesh consists of* 363 *nodes in* 342 *elements.*

where T denotes the nondimensional temperature, T_1 and T_m represent some constant temperature, and W and H are the width and height of the rectangular domain, respectively. The analytical solution is

$$T = T_m \frac{\sinh(\pi y/W)}{\sinh(\pi H/W)} \cos\frac{\pi x}{W} + T_1, \tag{4.2}$$

which makes the direct calculation of numerical error possible. It can be seen that the solution of the problem is highly nonlinear. Therefore, it is a suitable test problem to discretize and to solve with special bilinear elements. We choose $T_1 = 1$ and $T_m = 50$ in a unit square ($W = H = 1$) first and make use of its symmetry feature. The discretization is carried out in a domain of 0.5×1 with a homogeneous Neumann boundary condition applied at $x = 0$; see Figure 4.1 (a). In this case, all three kinds of special elements are deployed in the area where the discontinuity of the first-order derivative at the interelements is large. Shown in Figure 4.1 are the mesh with special and normal elements (a), the numerical solution (b), and the numerical error (c). The arbitrarily distorted elements close to the interface separating different refinement levels are introduced to test the robustness and the reliability of the special elements.

The numerical error is calculated as the difference between the numerical result and the analytical solution from (4.2). Obviously, the error in the region full of normal elements is small, which indicates also that the discontinuity of the first-order derivative at the interelements is not big enough to allow further refinement. Close to the interface between different refinement levels, a bigger error appears. The error largely results from the changing aspect ratio of distorted elements, the discretization error inherited by using bilinear elements, the error caused by discontinuity of the first-order (or higher-order) derivative of the global interpolation functions, and the truncation error from the computer.

However, after considering all of these factors, we find that the maximum value for relative error in the whole domain is 0.67%, while the average error norm is 4.8×10^{-5}. Quite an accurate numerical solution has been obtained from the mesh with such distorted elements. The seemingly rough mesh surface shown in Figure 4.1 (b) is actually the visual effect of the interior borders of distorted elements, and the surface is, in fact, quite smooth.

The next test case is a model combustion problem [1] with a one-step reaction ($A \to B$):

$$\frac{\partial T}{\partial t} = \frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} + D(1+\alpha - T)\exp\left(-\frac{\delta}{T}\right), \tag{4.3}$$

where α denotes the heat release rate, D denotes the Damköhler number, δ denotes the activation energy, and R denotes the reaction rate. Following [1], we choose $\alpha = 1$, $\delta = 20$, and $R = 5$. The initial condition for this case can be written as

$$T = 2, \quad x = 0 \text{ and } y = 0, \tag{4.4}$$

$$T = 1, \quad \text{otherwise.} \tag{4.5}$$

The initial condition at the origin, in fact, produces a constant "hot spot" and serves as an ignition source. The following boundary conditions are employed:

$$T = 2, \quad x = 0, \text{ and } y = 0, \tag{4.6}$$

$$\frac{\partial T}{\partial x} = 0, \quad x = 0, \ y \neq 0, \text{ and } y \neq 1, \tag{4.7}$$

$$\frac{\partial T}{\partial y} = 0, \quad y = 0, \ x \neq 0, \text{ and } x \neq 1, \tag{4.8}$$

$$T = 1, \quad x = 1, \text{ or } y = 0. \tag{4.9}$$

Physically, (4.3)–(4.9) describe an ignition process and a flame propagation after the ignition. Reactant A is converted into product B of high temperature during the combustion process. This problem has a nonlinear source term, which also makes a good test case. Though the analytical solution is not available, it is however a suitable case to test how the refinement strategy effectively works. Specifically, it tests how the local refinement with special elements closely follows the moving flame front in an unsteady phenomenon.

For this case, the element refinement is carried out by bisecting equally the big element's sides and the element merging by combining four small neighboring elements together if the error indicators (the discontinuities of first-order derivatives at inter-elements) fall outside the specified limits (or criteria).

The initial mesh is constructed with all three kinds of special bilinear elements introduced in section 2 as well as normal elements. The finest mesh around the origin is used to simulate better the ignition source at $t = 0$; see Figure 4.2 (a). There are 97 elements with 106 nodes in the mesh. The biggest element is 0.25×0.25, the smallest element is only 0.00098×0.00098 in size, and there are many special elements with different sizes in between. The use of special elements makes it practically possible to concentrate very few but extremely fine elements around the ignition source and to use coarse elements elsewhere.

Shown in Figure 4.2 is a sequence of numerical results from the initial condition until the flame reaches the boundaries opposite the hot spot. The 3-dimensional mesh

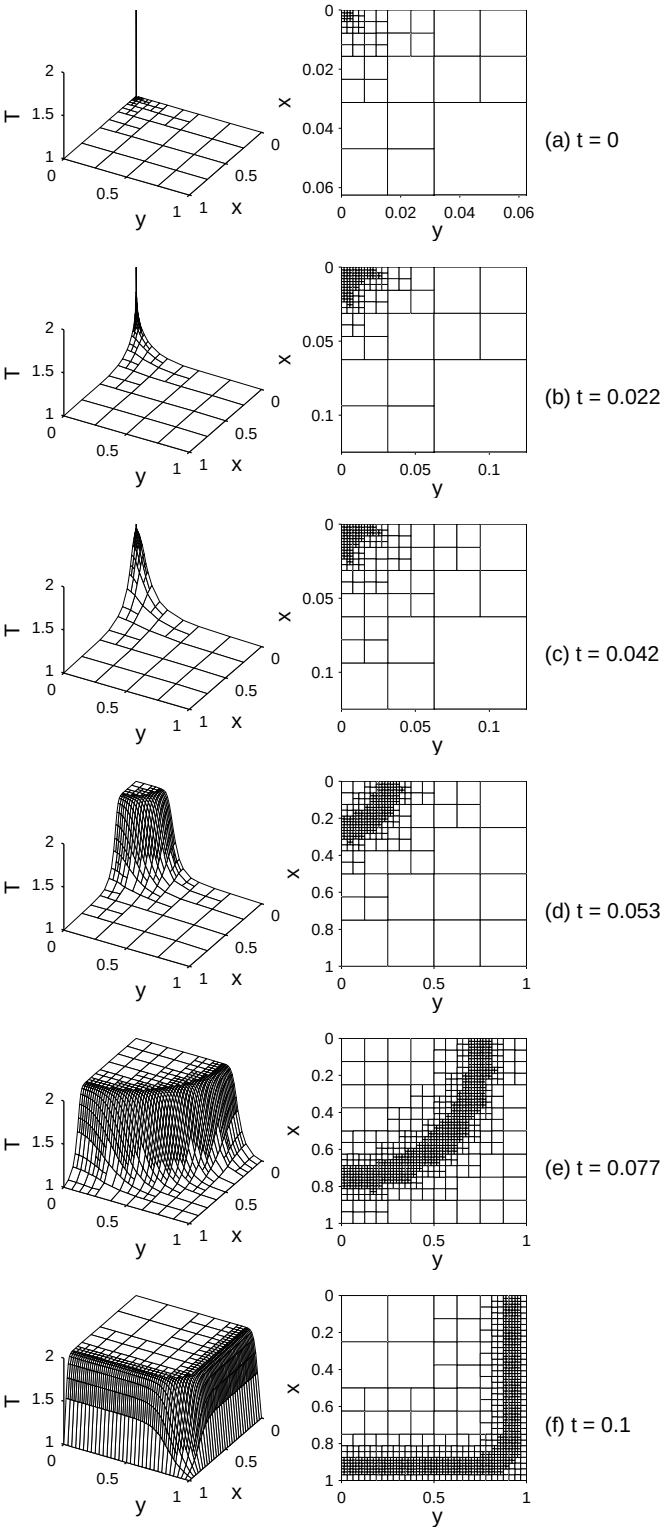


FIG. 4.2. *Numerical results with special linear elements on a sequence of adapted meshes for a model combustion problem. Mesh properties (number of nodes/number of elements):* (a) 106/97, (b) 188/181, (c) 203/196, (d) 304/313, (e) 900/949, *and* (f) 904/937.

surfaces at the left show numerical solutions at different times, and the 2-dimensional meshes at the right illustrate the element distribution close to the reaction zone.

The reactant is truly ignited at about $t = 0.042$; see Figure 4.2 (c). Before the ignition and during this ignition delay period, the reactant is actually undergoing a heat accumulating process around the hot spot (Figures 4.2 (a) and (b)). During this period, the ignition source induces a highly nonlinear feature around it, and correspondingly the elements are deployed at seven to eight different refinement levels in the whole domain with the finest close to the ignition source. After the ignition, the flame front begins to propagate outwards, and the finer elements start to be taken out from the region close to the ignition source until the flame front arrives at the boundaries, as shown in Figure 4.2 (c)–(f).

The fine elements generated from the local refinement process follow quite well with the flame front and leave the hot product behind with coarse elements. The finest elements are deployed exactly near the flame front where they are needed, and the special elements in the transient region give a smooth transition from the slowly to rapidly changing dynamic and chemical phenomena, or vice versa. As mentioned earlier, the error indicator based on the discontinuity of the first-order derivatives at the interelements have been used quite effectively throughout the self-adaptive refinement procedure. Before and after the ignition, the criteria for the error indicator were adjusted during the run time. Specifically, a higher upper limit was used before the ignition, and a lower upper limit was used after the ignition because the physical process shows a much stronger nonlinear feature before and during the ignition process than during the flame propagating process.

The last test case is a viscous flow past a circular cylinder (with a diameter of unity) described with incompressible Navier–Stokes equations of primitive variables. The dimensionless form of the governing equation can be written as

$$\begin{aligned} &\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0, \\ &u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} = -\frac{\partial p}{\partial x} + \frac{1}{Re}\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right), \\ &u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} = -\frac{\partial p}{\partial y} + \frac{1}{Re}\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right), \end{aligned} \tag{4.10}$$

where u and v denote the components of the velocity vector, p is the pressure, and Re is the Reynolds number. To discretize these equations, we take quadratic elements [20] for velocities and bilinear elements for pressure to satisfy the Babuska–Brezzi condition [2, 3]. To treat the convection terms, the method of successive substitution is used, which is quite suitable for such cases with moderate Reynolds numbers. For convection-dominated flows with higher Reynolds numbers, some upwind schemes [4, 12, 11] have to be considered.

Stable numerical solutions can be obtained for relatively higher Reynolds numbers, though experiments show that the vortex in the wake of a circular cylinder becomes oscillating at about $\mathrm{Re} \geq 40$ and the so-called Kármán vortex street develops. In the experiment, the influences of the possibly asymmetric cylinder, the improper alignment of the flow field, the potential 3-dimensional nature, and the disturbance in the laboratory all contribute to the earlier onset of the flow transition from steady to unsteady. In 2-dimensional numerical simulation, however, the absence of these influences delays the flow transition to unstable vortex shedding.

The incompressible viscous flows with Reynolds numbers of 10, 20, 40, and 100 have been calculated with special elements. In the following, the results of pressure are the focus for the analysis because only pressure is discretized on bilinear quadrilateral elements.

The calculation is started initially from an unstructured mesh with 553 nodes in 168 elements that covers half of the symmetric physical domain. Shown in Figure 4.3 (a) is the initial mesh with the boundary conditions. The initial mesh is constructed based on some knowledge of the flow; of 168 general quadrilateral elements, finer elements are distributed near the cylinder and aligned with the flow; a constant velocity is specified for the incoming flow, nonslip condition is used for the velocity on the surface of the cylinder, and the boundary condition for pressure is simply specified on a single boundary node at the top-right corner.

The local refinement is carried out at every few iteration steps. Though the elements are general quadrilaterals, the element refinement is still performed by halving the big element's sides, and the element merging is performed by combining four small neighboring elements.

In addition to the discontinuity of the first-order derivative, the error indicator also takes into account the contribution from the velocity direction changes within the element, in order to capture the vortex in the wake. Specifically, the element refinement also takes place in the region where the velocity vectors have a tendency to turn rapidly. We construct the following indicator, $e_{a,i}$, to measure such a tendency:

$$e_{a,i} = \frac{1}{\pi} \max \left\{ [\arctan(A_j) - \arctan(A_k)] : \ j,k \in \{\text{nodes in element } i\} \right\}, \tag{4.11}$$

$$A_l = \left(\frac{\partial v}{\partial x} u_l + \frac{\partial v}{\partial y} v_l \right) \Bigg/ \left(\frac{\partial u}{\partial x} u_l + \frac{\partial u}{\partial y} v_l \right), \qquad l = j,k. \tag{4.12}$$

For a steady state case, A_l equals $\frac{dv}{dt} / \frac{du}{dt}$, and therefore $\arctan(A_l)$ represents the acceleration direction of the flow at the location of node l. Indicator $e_{a,i}$ is a sensitive parameter to detect the possible direction change of the flow, for instance, in the vortex or the recirculation zone. The quantity $[\arctan(A_j) - \arctan(A_k)]$ is defined as the difference of flow acceleration directions at nodes j and k, which should be positive and less than π. Obviously, $e_{a,i}$ vanishes in the region where the flow goes straight.

The indicator for every individual dependent variable (unknown) is summed up without weighting (or with weighting to emphasize the influence of certain variables), and a single indicator is formed on element basis. By specifying a lower and an upper limit (criterion), refining or coarsening of elements is performed if the error indicators for the elements fall outside the specified range. In the self-adaptive procedure, special and normal elements are removed or added accordingly. Moreover, for different Reynolds numbers, the computational domain is extended in the y-direction until the pressure on the cylinder surface no longer depends on the domain size.

Shown in Figure 4.3 (b) and (c) is the final mesh for the converged solution with a Reynolds number of 20. In the adaptive process 375 elements with 1176 nodes are reasonably well distributed across the whole domain; see Figure 4.3 (b). Very fine elements are deployed in the region close to the front stagnation point and near the cylinder, less finer elements are deployed in the area further away from the cylinder and in the wake of the cylinder, and very coarse elements are distributed in the upper and down streams and in the far field. In the whole domain the smallest element is

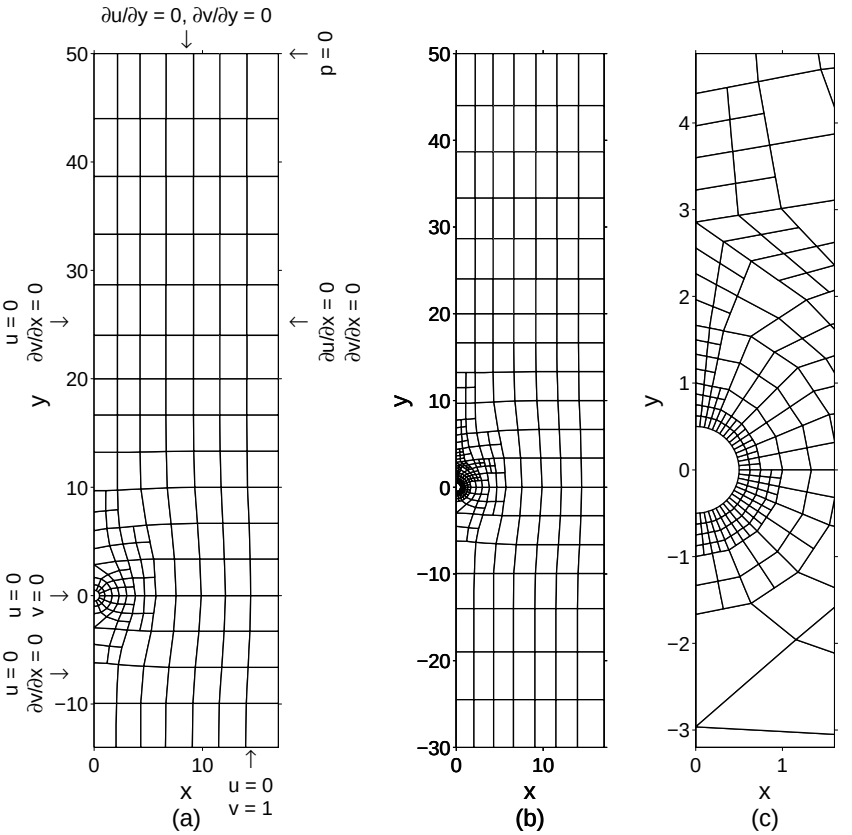


FIG. 4.3. *Initial mesh (*553 *nodes in* 168 *elements) and boundary conditions* (a) *for the flows past a circular cylinder, the final mesh* (b) *with* 1176 *nodes in* 375 *elements, and the closeup* (c) *of the mesh in the region around the cylinder for a case with a Reynolds number of* 20.

TABLE 4.1
Mesh properties and flow characteristics for the viscous flows past a circular cylinder with various Reynolds numbers.

Re	Domain	Element number	Node number	Separation point[°]	Vortex center location	Reattachment point
10	17×74	344	1085	163	0.108, 0.617	0.744
20	17×80	375	1176	143	0.209, 0.849	1.419
40	17×92	395	1238	133	0.289, 1.193	2.758
100	17×112	428	1341	123	0.402, 2.585	6.339

about 0.049×0.049, and the biggest one is 3×6 in size. Meshes for the other three cases have similar features. Table 4.1 shows some properties of the final meshes for the four cases. Since the elements are "economically" distributed, a relatively small number of discretized points (nodes) are required to accurately predict the detailed feature of each flow. For a typical case with a Reynolds number of 100, the mesh with special elements consists of 1341 nodes, while the numerical solutions were based on meshes of 3721 and 7410 discretized points in [7] and [8], respectively.

The pressure distribution on the surface of a body in a flow field is important for the prediction of the drag and the lift coefficients. Illustrated in Figure 4.4 are the pressure distributions obtained by the semianalytical method of series truncation [19],

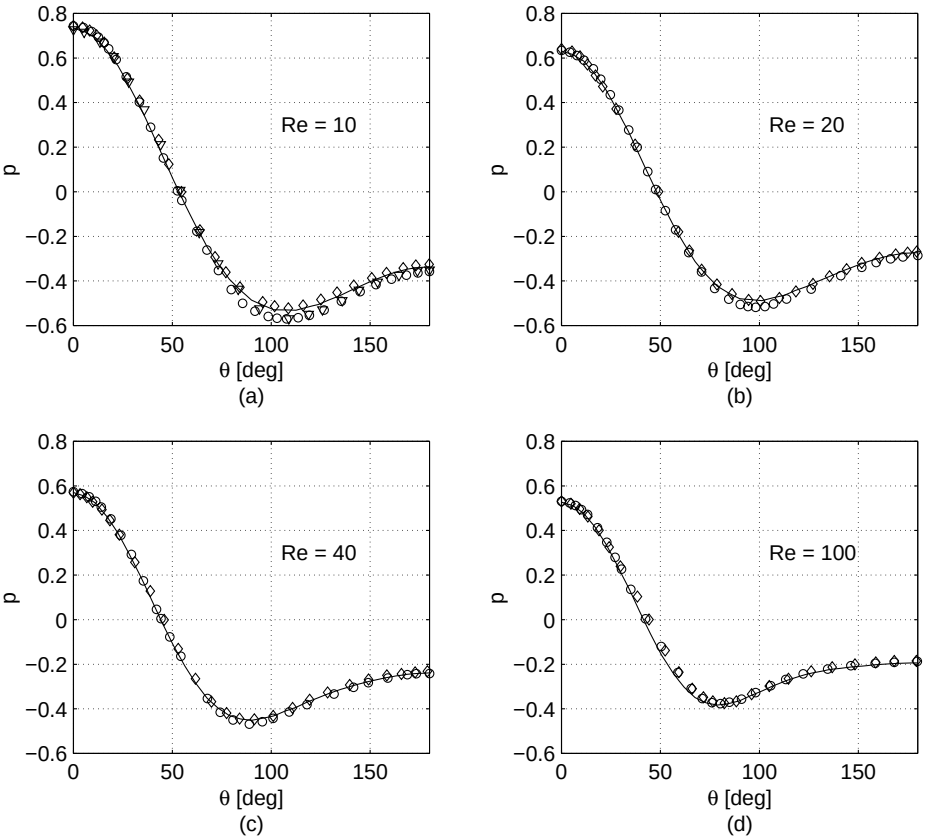


FIG. 4.4. *Pressure on the cylinder surface for the cases with Reynolds numbers of* 10 (a), 20 (b), 40 (c), *and* 100 (d). *The front stagnation point is at* $\theta = 0$, *and pressure is nondimensionalized by* ρv^2. *Triangles: semi-analytical results by Underwood* [19]; *circles: numerical results by Dennis* [7]; *diamonds: numerical results by Fornberg* [8]; *lines: numerical solution from the present work.*

TABLE 4.2
Pressure at front and rear stagnation points for the viscous flows past a circular cylinder with various Reynolds numbers.

Re	p	Underwood [19]	Dennis [7]	Fornberg [8]	Present work
10	p_f	0.728	0.744	0.743	0.739
	p_r	-0.350	-0.357	-0.326	-0.341
20	p_f		0.634	0.640	0.631
	p_r		-0.285	-0.27	-0.273
40	p_f		0.573	0.571	0.570
	p_r		-0.242	-0.23	-0.239
100	p_f		0.531	0.523	0.527
	p_r		-0.188	-0.17	-0.194

by the finite difference discretization of streamfunction-vorticity formulation [7, 8], and by the discretization of primitive variables with special quadratic elements [20] and special bilinear elements introduced in section 2. The pressures at the front and rear stagnation points, p_f and p_r, respectively, are also listed in Table 4.2. It can be seen that quite good agreements have been achieved.

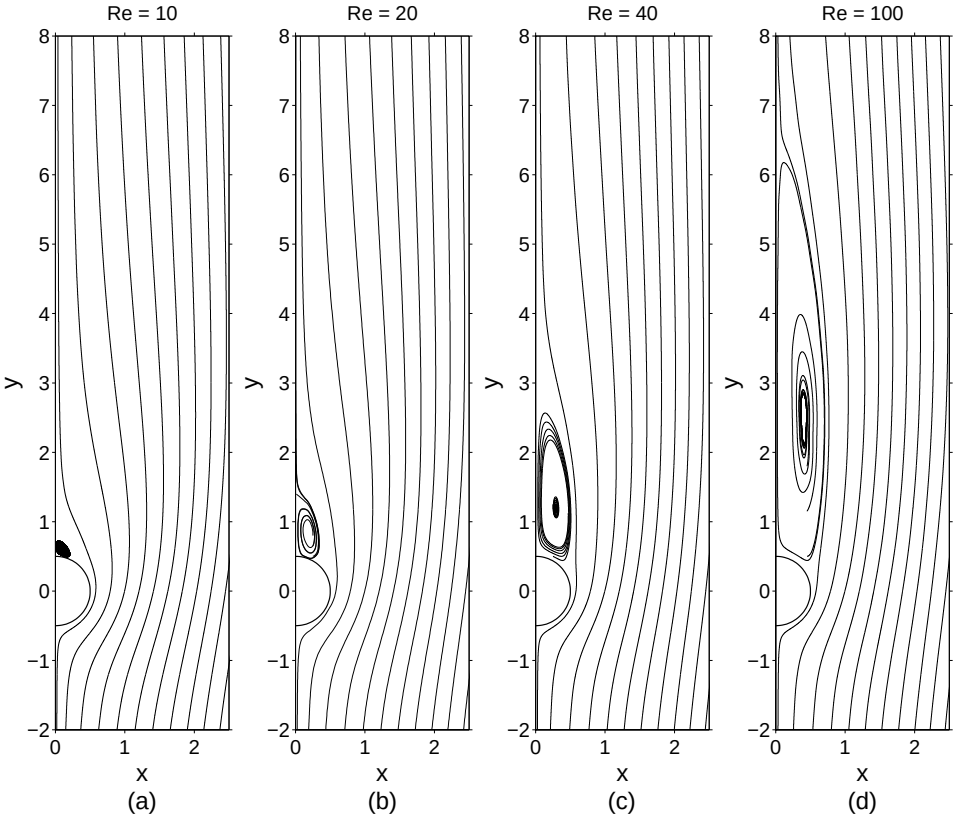


FIG. 4.5. *Streamlines for flows past a circular cylinder with Reynolds numbers of* 10 (a), 20 (b), 40 (c), *and* 100 (d). *The streamlines in the vortex regions are truncated for clarity.*

In the following, velocity-related results are presented only for completeness because the velocity is discretized on special quadratic elements. Shown in Figures 4.5 and 4.6 are the streamline plots and velocity fields in the vicinity of the cylinder for four different Reynolds numbers, respectively. The streamlines have been obtained from the postprocessing of velocity.

The separation points, the vortex centers, and the reattachment points for these four cases are listed in Table 4.1. The separation point is measured anticlockwise from the front stagnation point and is given in degree; the location of vortex center is described as a coordinate pair, while the reattachment point is indicated by its y-coordinate.

The vortex behind the cylinder starts to develop at $\mathrm{Re} = 7.334$, which is estimated as the reattachment point reaches the second node at the symmetric axis $(x = 0, y = 0.5625)$ above the cylinder. With an increasing Reynolds number, the streamline indicates clearly that the vortex grows with the reattachment point moving downstream, and the separation point moves back toward the front stagnation point; see Table 4.1 and Figure 4.5.

Velocity fields show that the flow decelerates toward the front stagnation point and accelerates over the cylinder, and the vortex develops in the wake for a Reynolds number ranging from 10 to 100. The velocity magnitude in the vortex zone is rather small even though the vectors are scaled up three times to make the vortex structure

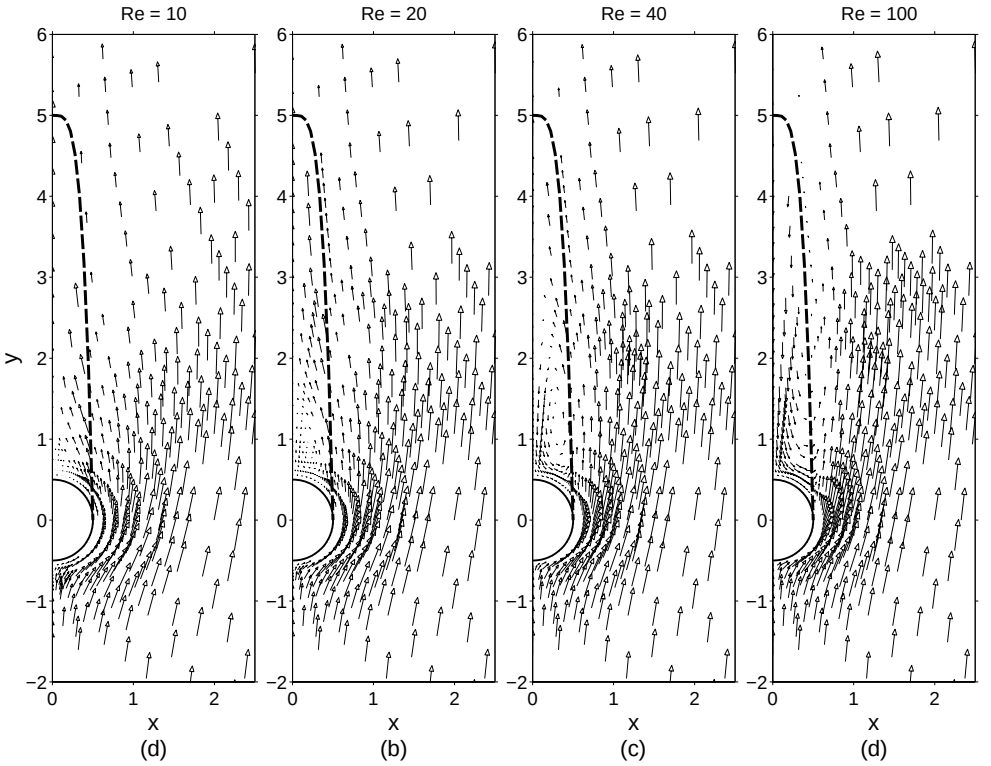


FIG. 4.6. *Velocity fields for viscous flows past a circular cylinder with Reynolds numbers of* 10 (a), 20 (b), 40 (c), *and* 100 (d). *For clarity, some velocity vectors have been taken out in regions with fine elements, and the vectors confined in the region marked with thick dashed lines are scaled up with a factor of* 3.

visible; see Figure 4.6. The error indicator based on the direction change of flow acceleration plays an important role in capturing the vortex structure in the flow field with such small velocities.

5. Final remarks. Special linear quadrilateral elements possess many useful features for the AFEM. These elements are C^0 continuous at interfaces, self-contained, accurate, capable of handling distortion, and easy to implement. The test cases have shown that they can be used effectively in solving such problems as heat transfer, diffusion, diffusion-reaction, pure fluid dynamic, and simple combustion cases.

With the error indicators based on the discontinuity of the first-order derivatives at interelements and the acceleration direction of the flow in an element, the AFEM with special linear quadrilateral elements captures well the localized physical phenomena in the domain of interest.

The implementation of special elements is quite a straightforward procedure. Only a small amount of additional computer storage and computing time is required in terms of the bookkeeping list or the discretization.

Appendix. Special triangular elements.

The concept of special elements can be extended to triangular elements. Shown in Figure A.1 is a typical situation where different triangular elements at different refinement levels meet.

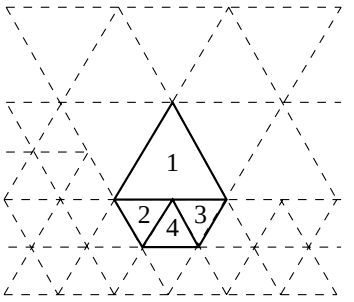


FIG. A.1. *Triangular elements in a transient region.*

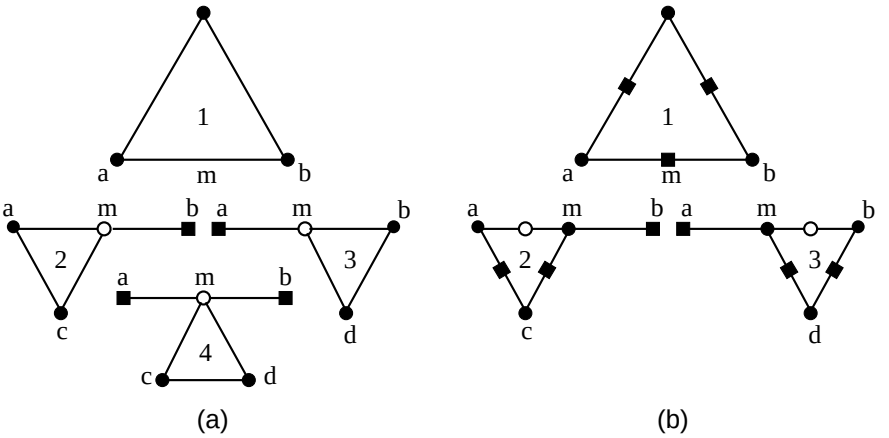


FIG. A.2. *Special linear* (a) *and quadratic* (b) *triangular elements. The numbers indicate elements, and the lower-case letters represent nodes; the solid circles indicate regular corner nodes, the void circles indicate irregular nodes, and the solid squares indicate middle nodes or irregular nodes' replacement nodes.*

In the following, we consider a case of dividing a triangular element into four smaller triangles and generating one irregular node only at one side of a small triangle. The triangular element with more than one side at the interface between different refinement levels can always be transformed or reconstructed so that only one side faces at such an interface.

For the case with linear triangular elements, three different special triangular elements can be constructed to satisfy the C^0 continuity conditions at interelements; see Figure A.2 (a). Special element 2 with nodes a, b, and c and special element 3 with nodes d, b, and a can be created. Because element 4 has only *one* apex (irregular node m) on interface ab, special element 4 has to be constructed uniquely and consistently by taking the information from four regular nodes (a, b, d, and c) instead of three. Elements 2, 3, and 4 are the special linear triangular elements of the first, second, and third kind, respectively.

Since the nodal values of a linear triangular element are always on a plane with straight sides, irregular node m can be placed anywhere on interface ab, and the C^0 continuity between elements 1, 2, 3, and 4 is always kept. However, to place this irregular node at the middle of interface ab can better maintain the aspect ratio during the refinement process and is most desired because of the accuracy consideration.

It can be seen that each of the special elements 2 and 3 has three nodes, and

their linear interpolation functions can be therefore constructed. The interpolation functions for element 4 require information of four nodes (see Figure A.2 (a)), and therefore the interpolation function cannot be constructed in a self-contained manner. This might not be convenient from a programming point of view, but it avoids dividing element 1, as in the procedure with a red-green refinement approach.

Similar to the case for quadrilateral elements, the information about node m will be merged consistently into the interpolation functions of the three special elements, which results in the C^0 continuity between elements 1, 2, 3, and 4.

For the case with quadratic triangular elements, only two special elements are required, that is, element 2 of the first kind, and element 3 of the second kind; see Figure A.2 (b). Element 1 is a normal six-node triangular element with a single middle node, node m, on interface ab. Element 2 has three corner nodes, two middle nodes, and an irregular node on interface am, and so does element 3.

In order to keep the C^0 continuity across interface am, we choose node b to replace the irregular node and to form special element 2. The property of the interpolation functions guarantees the C^0 continuity at interface am. A similar procedure can be applied to element 3, and the C^0 continuity will be satisfied at interface mb. It can be seen that the two special elements are self-contained for the case with quadratic triangular elements, which is not the case for special linear triangular elements.

Acknowledgments. The author wishes to thank Professor B. Rogg for the helpful discussion and Dipl.-Ing. B. Michaelis for the valuable suggestions and for the assistance in verifying some important details. The author is grateful to Dr. K. N. Lakshmisha for carefully reading the manuscript and making many useful suggestions. The author is also greatly indebted to emeritus Professor K. Gersten for his inspiring advice. Furthermore, the constructive suggestions and comments of the anonymous referees are greatly acknowledged.

REFERENCES

[1] S. Adjerid and J. E. Flaherty, *A Local Refinement Finite Element Method for Two-Dimensional Parabolic Systems*, Tech. rep. 86-7, Department of Computer Science, Rensselaer Polytechnic Institute, Troy, NY, 1986.

[2] I. Babuska and A. K. Aziz, *Survey lectures on the mathematical foundations of the finite element method*, in Mathematical Foundations of the Finite Element Method with Applications to Partial Differential Equations, A. K. Aziz, ed., Academic Press, New York, 1972, pp. 1–359.

[3] F. Brezzi, *On the existence, uniqueness and approximation of saddle-point problems arising from Lagrangian multipliers*, Rev. Francaise Automat. Informat. Recherche Operationnelle Sér. Rouge, 8 (1974), pp. 129–151.

[4] A. Brooks and T. Hughes, *Stream-line upwind/Petrov-Galerkin formulation for convection dominated flow with particular emphasis on the incompressible Navier–Stokes equations*, Comput. Methods Appl. Mech. Engrg., 32 (1982), pp. 199–259.

[5] T. Chung, *Finite Element Analysis in Fluid Dynamics*, McGraw–Hill, New York, Beirut, Bogota, 1978.

[6] L. Demkowicz, J. Oden, W. Rachowicz, and O. Hardy, *Toward a universal h-p adaptive finite element strategy, part* 1. *Constrained approximation and data structure*, Comput. Methods Appl. Mech. Engrg., 77 (1989), pp. 79–112.

[7] S. Dennis and G.-Z. Chang, *Numerical solutions for steady flow past a circular cylinder at reynolds numbers up to* 100, J. Fluid Mech., 42, (1970), pp. 471–489.

[8] B. Fornberg, *A numerical study of viscous flow past a circular cylinder*, J. Fluid Mech., 98 (1980), pp. 819–855.

[9] A. Gupta, *A finite element for transition from a fine to a coarse grid*, Internat. J. Numer. Methods Engrg., 12 (1978), pp. 35–45.

[10] J. Holman, *Heat Transfer*, 8th ed., McGraw–Hill, New York, 1997, pp. 77–80.

[11] T. Hughes, L. Franca, and M. Balestra, *A new finite element formulation for computational fluid dynamics:* V. *Circumventing the Babuska–Brezzi conditions: A stable Petrov–Galerkin formulation of the Stokes problem accommodating equal-order interpolations*, Comput. Methods Appl. Mech. Engrg., 59 (1986), pp. 85–99.

[12] T. Hughes, L. Franca, and G. Hulbert, *A new finite element formulation for computational fluid dynamics:* VII. *The Galerkin/least-squares method for advective-diffusive equations*, Comput. Methods Appl. Mech. Engrg., 73 (1989), pp. 173–189.

[13] R. Kornhuber and G. Wittum, *Discretization and iterative solution of convection diffusion equations*, in Incomplete Decompositions (ILU) Algorithms, Theory and Applications, Notes Numer. Fluid Mech. 41, Vieweg, Braunschweig, Germany, 1993, pp. 67–77.

[14] S. Patankar, *Numerical Heat Transfer and Fluid Flow*, McGraw–Hill, New York, 1981.

[15] L. Plank, E. Stein, and D. Bischoff, *Accuracy and adaptivity in the numerical analysis of thin-walled structures*, Comput. Methods Appl. Mech. Engrg., 82 (1990), pp. 223–256.

[16] R. Shapiro, *Adaptive Finite Element Solution Algorithm for the Euler Equations*, Vieweg, Braunschweig, Germany, 1991.

[17] E. Stein, W. Rust, and S. Ohnimus, *h- and d-adaptive FE methods for two-dimensional structure problems including post-bucking of shells*, Comput. Methods Appl. Mech. Engrg., 101 (1992), pp. 315–354.

[18] C. Taylor and T. Hughes, *Finite Element Programming of the Navier-Stokes Equations*, Pineridge Press, Swansea, Wales, 1981.

[19] R. Underwood, *Calculation of imcompressible flow past a circular cylinder at moderate Reynolds numbers*, J. Fluid Mech., 37, (1980), pp. 95–114.

[20] W. Wang, *Special quadratic quadrilateral finite elements for local refinement with irregular nodes*, Comput. Methods Appl. Mech. Engrg., 182 (2000), pp. 109–134.

SIAM J. SCI. COMPUT.
Vol. 22, No. 6, pp. 2051–2074
© 2001 Society for Industrial and Applied Mathematics

CONSTRAINT PARTITIONING FOR STABILITY IN PATH-CONSTRAINED DYNAMIC OPTIMIZATION PROBLEMS*

SOUMYENDU RAHA† AND LINDA R. PETZOLD‡

Abstract. In this paper an algorithm for extracting a stable differential-algebraic subsystem from a path-constrained dynamical system is proposed. The subsystem may be integrated directly by a differential-algebraic system integrator to evaluate constraints in shooting- or multiple shooting-type direct methods for solving path-constrained dynamic optimization problems. The algorithm appends algebraic constraints to the unconstrained ordinary differential equation subsystem based on a stability estimate for the resulting differential-algebraic system. The logarithmic norm is used to compute a stability estimate for index 1 and index 2 subsystems. The working of the algorithm is illustrated with examples.

Key words. dynamic optimization, stability, path constraints, constraint partitioning

AMS subject classifications. 65L80, 65L07, 65K99

PII. S1064827500372390

1. Introduction.

Dynamic optimization problems where some of the state variables of the dynamic system are required to follow prescribed functions in time (path constraints) are referred to as path-constrained dynamic optimization problems. This kind of preprogramming gives rise to constraints which, along with the ordinary differential equations (ODEs) or differential-algebraic equations (DAEs) describing the dynamic system, may increase the index [1], [5] and/or alter the stability of the resulting DAE problem. Since available algorithms can integrate DAEs up to differential index 2, the dynamic optimization problem is often solved [19], [18], [16] by an optimization scheme that incorporates integrating a lower-index (1 or 2) DAE system originating from the dynamical system. This paper addresses the problems of finding an optimal set of criteria based on stability of the DAE for identifying constraints which are best included in the DAE, rather than enforced by the optimizer.

A general path-constrained dynamics system can be written as

$$\text{(1a)} \qquad \text{Minimize} \qquad J = \int_0^{t_{\text{final}}} \chi(x,u,t)\,dt$$

$$\text{subject to}$$

$$\text{(1b)} \qquad F(\dot{x},x,u,t) = 0,$$

$$\text{(1c)} \qquad g(x,u,t) \le 0,$$

$$\text{(1d)} \qquad b_u \ge \begin{pmatrix} x^T & u^T \end{pmatrix}^T \ge b_l.$$

*Received by the editors May 18, 2000; accepted for publication (in revised form) November 30, 2000; published electronically March 28, 2001. This research was supported by NSF Multidisciplinary Challenge grant CCR-98-9896198, by NSF KDI grant ATM-9873133, by the NSF/DARPA Virtual Integrated Processing program, and by DOE DE-FG03-98ER25354.

http://www.siam.org/journals/sisc/22-6/37239.html

†Scientific Computation Program, Department of Computer Science, University of Minnesota, Minneapolis, MN 55455 (raha@cs.umn.edu).

‡Department of Mechanical and Environmental Engineering and Department of Computer Science, University of California Santa Barbara, Santa Barbara, CA 93106 (petzold@engineering.ucsb.edu). All communications should be addressed to this author.

These problems may be solved by shooting- or multiple shooting-type methods [2], [16], [18], [19]. In a multiple shooting-type scheme, the original time interval of the problem is subdivided into subintervals $[t_k, t_{k+1}] \in [t_0, t_{\text{final}}]$. The DAEs (1b)–(1c) or subsystems thereof are integrated over the subintervals. Enforcement of continuity of the variables across the subintervals leads to constraints for the overall nonlinear optimization problem (1). Variables not included in the DAE are modeled as polynomial functions in time, and their coefficients are optimization variables. Smoothness and continuity of the polynomial approximations across the subinterval boundaries are expressed as constraints in the nonlinear optimization problem (1). All remaining constraints and bounds that have not been included in the DAE (1b)–(1c) or its subsystem for integration are handled from the overall optimization problem (1). Typically, the algebraic constraints excluded from the DAE system are entered in the optimizer as some positive function representing the constraint violation integrated over a shooting subinterval, which is required not to exceed a small positive quantity.

For this algorithm to work we must be able to integrate the DAE subsystem efficiently to the end of each subinterval. Thus we must choose the subsystem so that it is lower index (index 1 or 2) and stable. For nonlinear problems it may also be important that the subsystem is physically reasonable.

The choice of variables and constraints to include in the partitioned DAE (pDAE), can influence the stability of the pDAE, as illustrated by the following example. Consider the system

$$\begin{aligned} \dot{x} &= 1000.0x + u, \\ bx + u &= 0. \end{aligned}$$

The system is unstable when the constraint is included in the optimizer and the ODE is integrated (u being handled by the optimizer). But the system is stable for $b = 10000$ when integrated as an index 1 DAE. Additional constraints can also destabilize a pDAE, as we will see later.

Several algorithms have been proposed in the literature for partitioning DAE systems based on *structure*. In [9], a graph-based partitioning algorithm is proposed to yield a system with index 1 structure, and in [19] and [10] partitioning algorithms based on nonsingularity of the index 1 Jacobian are described. Noting that recent DAE software, e.g., RADAU5 [12] and DASPK3.0 [15], can handle index 1 and certain kinds of index 2 systems, in a forthcoming paper [17] we outline an algorithm for partitioning to obtain index 1 and index 2 structure.

The focus of the present paper is on partitioning for *stability*. Thus we assume that, if necessary, the problem has already been partitioned for index 1 or index 2 structure (i.e., that index 1 and index 2 constraints and corresponding variables have been identified) and can be written in the form

$$\text{Minimize} \qquad J = \int_{t_0}^{t_{\text{final}}} \chi(x, u, t)\, dt, \tag{2a}$$

subject to pDAE

$$\dot{x} = f(x, v, w, u_2(t), t), \tag{2b}$$

index 1 constraints

$$\Psi_1(x, v, t) = 0, \tag{2c}$$

index 2 constraints

$$\Psi_2(x, t) = 0, \tag{2d}$$

path constraints

$$p(x,t) = 0, \tag{2e}$$

and inequality constraints

$$c(x,t) \le 0, \tag{2f}$$

$$b_u \ge \begin{pmatrix} x^T & v^T & w^T & u_2(t)^T \end{pmatrix}^T \ge b_l, \tag{2g}$$

where $\frac{\partial \Psi_1}{\partial v}$ is nonsingular for index 1 and $\frac{\partial \Psi_2}{\partial x}\frac{\partial f}{\partial w}$ is nonsingular for index 2. Here the control vector u has been partitioned as $(v,w)^T$ and $u_2(t)$.

We note that partitioning is an issue for shooting and multiple shooting methods. An alternative to these methods is to apply a boundary value approach with a full discretization of the DAE system [4], [7]. For that approach, the stability concerns addressed here are no longer relevant and partitioning would not be needed.

In the next sections, stability criteria and computable estimates for building the pDAEs are developed. The criteria are based on logarithmic norms of quantities associated with the stability of the pDAE and physical information available about the dynamical system. An algorithm based on these criteria is given, and its working is illustrated with examples.

2. Measures of stability. To assess the stability we will consider the DAE system linearized at $t = t_k$

$$F_d := \dot{x} - f(x,v,w,u_2(t),t) := \dot{x} - (Hx + Qv + Tw + r(t)) = 0, \tag{3a}$$

$$\Psi_1 := C_v v + C_1 x + r_1(t) = 0, \tag{3b}$$

$$\Psi_2 := C_2 x + r_2(t) = 0, \tag{3c}$$

where $H \in \mathcal{R}^{n_x \times n_x}$, $Q \in \mathcal{R}^{n_x \times n_v}$, $T \in \mathcal{R}^{n_x \times n_w}$, $C_v \in \mathcal{R}^{n_v \times n_v}$, $C_1 \in \mathcal{R}^{n_v \times n_x}$, and $C_2 \in \mathcal{R}^{n_w \times n_x}$. The vectors $x \in \mathcal{R}^{n_x}$, $v \in \mathcal{R}^{n_v}$, $w \in \mathcal{R}^{n_w}$, $r(t) : \mathcal{R} \to \mathcal{R}^{n_x}$, $r_1(t) : \mathcal{R} \to \mathcal{R}^{n_v}$, and $r_2(t) : \mathcal{R} \to \mathcal{R}^{n_w}$. The local stability of an ODE of the form $\dot{x} = f(x,t)$ at $t = t_k$ is measured in terms of the largest real part in the set of eigenvalues of $\frac{\partial f}{\partial x}$ evaluated at $t = t_k$. Thus, $\max_\lambda \Re(\lambda(\frac{\partial f}{\partial x})) \le 0$ for a locally stable ODE. For nonlinear ODEs the logarithmic norm has often been used as a measure of stability [11]. In this section we outline and derive some basic properties of the logarithmic norm that will be needed in the derivation of the stability estimates of section 3.

For estimating bounds we shall use of the following well-known property.

LEMMA 2.1. *For any square symmetric real matrix H and any unit vector e, the expression $e^H H e$ subject to the condition $e^H e - 1 = 0$ is a maximum when e is an eigenvector corresponding to the maximum eigenvalue of H, and then* $\max_e e^H H e = \lambda_{\max}(H)$. *Also, it is known that* $\min_e e^H H e = \lambda_{\min}(H)$. *Hence for any unit vector* $e \in \mathcal{C}^{n_H}$, $\lambda_{\min}(H) \le e^H H e \le \lambda_{\max}(H)$.

The logarithmic norm provides an upper bound of the real part of the eigenvalue of the stability matrix.

DEFINITION 2.2 (logarithmic norm). *The logarithmic norm for a square matrix H in real or complex space is defined as*

$$\mu(H) := \lim_{\epsilon \to 0, \epsilon > 0} \frac{||I + \epsilon H|| - 1}{\epsilon}. \tag{4}$$

The logarithmic norm depends on the choice of matrix norm. In this paper we use the matrix norm induced by the 2-norm. Then the logarithmic norm for a square matrix $H \in \mathcal{R}^{n\times n}$ is given by (following Theorem 10.5 in Chapter I10 in [11])

$$\mu(H) = \max_{\lambda} \lambda(0.5(H^T + H)) \quad \text{when} \quad ||.|| = ||.||_2, \tag{5}$$

where λ denotes the eigenvalues. Also, the logarithmic norm satisfies a triangle inequality property along with some other useful ones. For a square matrix $H, H_1, H_2 \in \mathcal{R}^{n\times n}$,

$$\mu(H_1 + H_2) \leq \mu(H_1) + \mu(H_2) \tag{6a}$$

$$-||H|| \leq \mu(H) \leq ||H||, \tag{6b}$$

$$\mu(\alpha H) = \alpha\mu(H) \quad \text{for} \quad \alpha \geq 0, \tag{6c}$$

$$\mu\left(\int H(t)dt\right) \leq \int \mu(H(t))dt. \tag{6d}$$

All the above properties can be proved from the basic definition (4) of the logarithmic norm and by using the triangle inequality and other properties of the 2-norm ($||.||$).

Besides the above properties, the logarithmic norm is preserved under a unitary transformation.

LEMMA 2.3. *If H is a square matrix and Q is a unitary square matrix of the same dimension, then*

$$\mu(QHQ^T) = \mu(H). \tag{7}$$

Proof. Using the property (5), we obtain

$$\mu(QHQ^T) = \max_{\lambda} \lambda(0.5(QHQ^T + QH^TQ^T)) = \max_{\lambda} \lambda(0.5Q(H + H^T)Q^T).$$

Since a unitary transformation leaves the eigenvalues of a matrix unchanged, we have

$$\max_{\lambda} \lambda(0.5Q(H + H^T)Q^T) = \max_{\lambda} \lambda(0.5(H + H^T)). \qquad \square$$

Next, we establish the connection between the logarithmic norm and the maximum real part in the set of eigenvalues of a real square matrix.

LEMMA 2.4. *The following estimate holds for a square matrix $H \in \mathcal{R}^{n_H\times n_H}$:*

$$\max_{\lambda} \quad \Re(\lambda(H)) \leq \mu(H).$$

Proof. The above relation follows from the property of the logarithmic norm (5) so that $\mu(H) = \max_\lambda \lambda(\frac{(H+H^T)}{2})$. Since the eigenvector $e_\mu^{\max}$ corresponding to the maximum eigenvalue $\lambda_\mu^{\max}$ of $\frac{(H+H^T)}{2}$ maximizes the function $e_\mu^T \frac{(H+H^T)}{2} e_\mu$ subject to $e_\mu^T e_\mu - 1 = 0$, the choice of any other unit vector such as the eigenvector $e_H^{\max}$ (i.e., the eigenvector corresponding to the eigenvalue of H having the maximum real part) of H leads to a lesser value of the function. Also $\lambda_\mu^{\max}$ (where the subscript denotes the eigenvalue of $H^T + H$ and the superscript denotes the maximum value of an eigenvalue) and e are always real for real H since $H + H^T$ is a real symmetric matrix. Thus, using Lemma 2.1, we get

$$\mu(H) = \lambda_\mu^{\max} = e_\mu^{\max T} \frac{(H + H^T)}{2} e_\mu^{\max} \geq 0.5(e_H^{\max H} H e_H^{\max} + e_H^{\max H} H^T e_H^{\max})$$
$$= \max_{\lambda} \Re(\lambda(H)). \qquad \square$$

The logarithmic norm of a rank 1 system can be used to get estimates for the change in stability of a pDAE system by appending a single algebraic constraint.

LEMMA 2.5 (logarithmic norm of a rank 1 update). *Let u_1 and u_2 be unit vectors of dimension n. The logarithmic norm of the rank 1 matrix $u_1 u_2^T$ is given by*

$$\mu(u_1 u_2^T) = \max(0.5(u_1^T u_2 \pm 1), 0) \quad for \quad n > 2, \tag{8a}$$

$$\mu(u_1 u_2^T) = \max(0.5(u_1^T u_2 \pm 1)) \quad for \quad n = 2. \tag{8b}$$

Proof. Using property (5) the eigenvalues of the rank 2 system $0.5(u_1 u_2^T + u_2 u_1^T)$ are $0.5(u_1^T u_2 \pm 1)$ with the corresponding eigenvectors being $\frac{u_1 \pm u_2}{2}$. If $n > 2$, the system being rank 2, the other eigenvalues are zero. ∎

3. Stability of the pDAE. In this section we study the stability of index 1 and index 2 DAEs. We present conditions based on the logarithmic norm for assessing the effect of stability of appending or deleting a single algebraic constraint from the pDAE.

3.1. Index 1 stability. The underlying ODE (UODE) for an index 1 subsystem is locally stable if the largest real part of the eigenvalues of $H - QC_v^{-1}C_1$ is less than or equal to zero. For practical computations we require it to be less than a positive real number (TOL) of suitable size. This condition can be written in terms of the logarithmic norm.

THEOREM 3.1 (index 1 stability). *The index 1 pDAE subsystem is stable if*

$$\mu(H - QC_v^{-1}C_1) \leq 0. \tag{9}$$

Proof. The proof follows from Lemma 2.4. ∎

COROLLARY 3.2. *An unstable ODE ($\mu(H) > 0$) or a pDAE already in place can be stabilized by appending index 1 constraints if for the resulting system*

$$\mu(H - QC_v^{-1}C_1) \leq \mu(H) + \mu(-QC_v^{-1}C_1) \leq 0,$$

or

$$\mu(-QC_v^{-1}C_1) \leq -\mu(H). \tag{10}$$

The condition for checking the stability of an index 1 subsystem upon appending a single algebraic constraint can be deduced using the analytical result in Lemma 2.5. For a single algebraic constraint, Q is a column vector (say, $\tilde{Q}$), C_1 is a row vector (say, $\tilde{C_1}$), and C_v is a scalar represented by c_v. Let $u_Q = \frac{-\tilde{Q}}{||\tilde{Q}||_2}$, $u_{C_1}^T = \frac{\text{sign}(c_v)\tilde{C_1}}{||\tilde{C_1}||_2}$, and $\gamma = \frac{||\tilde{Q}||_2 . ||\tilde{C_1}||_2}{|c_v|} \geq 0$. Then $\mu(H - \tilde{Q}c_v^{-1}\tilde{C_1}) = \mu(H + \gamma u_Q u_{C_1}^T)$. Computationally, the condition for stability is

$$\mu(H + \gamma u_Q u_{C_1}^T) \leq \mu(H) + \mu(\gamma u_Q u_{C_1}^T) = \mu(H) + \tilde{\gamma} \leq TOL, \tag{11}$$

where $\tilde{\gamma}$ is defined using Lemma 2.5 as

$$\mu(\gamma u_Q u_{C_1}^T) = \tilde{\gamma} = \gamma \max\left(0, \frac{(u_{C_1}^T u_Q \pm 1)}{2}\right) \tag{12}$$

for $n_x > 2$ (where n_x is the dimension of x). For $n_x = 2$, the maximum is taken over $\frac{(u_{C_1}^T u_Q \pm 1)}{2}$ only. ∎

Next we compute (12) for each index 1 constraint being appended to a group of already existing index 1 constraints. Thus let

$$\Psi_1 \equiv C_1 x + C_v v + c_c v_1 + r_1(t) = 0$$

be the existing group of constraints to which a constraint of the form

$$\tilde{C}_1 x + c_v v_1 + c_r v + r_{11}(t) = 0$$

is considered for being appended. This is equivalent to appending the constraint

$$\bar{\Psi}_1 \equiv \bar{C}_1 x + \bar{C}_v v_1 + \bar{r}_1(t) = 0$$

to the UODE

$$\dot{x} = (H - QC_v^{-1}C_1)x + Tw(t) + r(t),$$

where

$$\bar{C}_1 = \tilde{C}_1 - c_r C_v^{-1} C_1, \tag{13a}$$

$$\bar{C}_v = c_v - c_r C_v^{-1} c_c, \tag{13b}$$

and $\bar{r}_1(t)$ represents the terms that are functions of time only. While evaluating (12), u_{C_1} is computed as $u_{\bar{C}_1} = \frac{\text{sign}(\bar{C}_v)C_1}{||\bar{C}_1||_2}$. Then $\tilde{\gamma}$ for second or later index 1 constraints is computed as

$$\tilde{\gamma}_{\text{append}} = \frac{||\tilde{Q}||_2 ||\bar{C}_1||_2}{|\bar{C}_v|} \max\left(0, \frac{(u_{\bar{C}_1}^T u_Q \pm 1)}{2}\right) \tag{14}$$

(for $n_x > 2$), where $\tilde{Q}$ is the column vector in the differential equations corresponding to the algebraic variable chosen for the index 1 equation under consideration. Summarizing, we have the following theorem.

THEOREM 3.3 (stability of index 1 constraints). *Checking* (11)*, with* $\tilde{\gamma}$ *given by* (12) *or* (14) *as appropriate, is sufficient for determining the stability when including a single index* 1 *algebraic constraint in the pDAE.*

Proof. The proof follows from the discussion above. ∎

3.2. Index 2 stability. The stability analysis for an index 2 pDAE is done by reducing the system to an essential underlying ODE (EUODE) [3]. The index 2 system ((3a)–(3c)) can be reduced to a UODE by differentiating the index 2 constraint once with respect to t and eliminating the index 1 variables to obtain an expression for w. Thus

$$w = -(C_2T)^{-1}(C_2Hx - C_2QC_v^{-1}C_1x - C_2QC_v^{-1}r_1(t) + C_2r(t) + \dot{C}_2x + \dot{r}_2(t)).$$

Also define

$$R := (I_{n_x} - T(C_2T)^{-1}C_2). \tag{15}$$

Then the UODE can be written as

$$\begin{aligned}\dot{x} = &(R(H - QC_v^{-1}C_1) - T(C_2T)^{-1}\dot{C}_2)x \\ &+R(r(t) - QC_v^{-1}r_1(t)) - T(C_2T)^{-1}\dot{r}_2(t).\end{aligned} \tag{16}$$

The idempotent projection R has certain properties in connection with its decomposition. A decomposition of R (e.g., QR, SVD, or Schur) can be used to obtain an EUODE from (16). In this paper, for simplicity we use the QR-decomposition of R to construct an EUODE for the index 2 subsystem.

LEMMA 3.4. *The projection $R \in \mathcal{R}^{n_x \times n_x}$ is idempotent and has a rank of $n_x - n_w$ with exactly n_w eigenvalues of zero and n_x eigenvalues of* 1.

Proof. R is idempotent by construction: $RR = (I_{n_x} - T(C_2T)^{-1}C_2) = R$.

From the definition of R (15) it is obvious that $C_2R = 0$. If λ_R denotes an eigenvalue and e_R denotes an eigenvector of R, the eigenvalue equation for R is obtained as $Re_R - \lambda_R e_R = 0$, which after premultiplication by C_2 yields $\lambda_R C_2 e_R = 0$. Then either $\lambda_R = 0$ or $C_2 e_R = 0$. If $C_2 e_R = 0$, then $Re_R - \lambda_R e_R = 0$ gives $e_R - \lambda_R e_R = 0$, i.e., $\lambda_R = 1$ (e_R being a unit vector). Again, if $\lambda_R \neq 1$, then $Re_R = 0$, in which case the eigenvector corresponds to $\lambda_R = 0$. When $\lambda_R = 0$, $C_2 e_R \neq 0$. Since C_2 is of rank n_w, row vectors of C_2 have a basis of n_w independent vectors of dimension n_x. Thus it is possible to find a set of n_w unit vectors of dimension n_x such that $C_2 e_R \neq 0$ and such that they are the eigenvectors corresponding to $\lambda_R = 0$. Corresponding to $\lambda_R = 1$, it is then possible to find $n_x - n_w$ other unit eigenvectors orthogonal to row vectors of C_2. Thus R has n_w zero eigenvalues, and its other $n_x - n_w$ eigenvalues are 1. Thus the upper triangular matrix from the Schur decomposition of R has n_w zeroes and $n_x - n_w$ 1's on the diagonal and can be written in the form $\begin{pmatrix} I_{n_x \times n_w} & \Xi \\ 0 & 0 \end{pmatrix}$. Then R has $n_x - n_w$ nonzero singular values and hence a rank equal to $n_x - n_w$. ∎

LEMMA 3.5. *If $\bar{W} \in \mathcal{R}^{n_x \times n_x}$ and $\bar{S} \in \mathcal{R}^{n_x \times n_x}$ are the unitary matrix and the upper triangular matrices, respectively, obtained from the QR-decomposition of R, then $\bar{W}$ can be partitioned as $(W \quad \tilde{W})$ and $\bar{S}$ can be partitioned as $\binom{S}{0}$, where W is an $n_x \times n_x - n_w$ real matrix, S is an $n_x - n_w \times n_x$ real matrix, and $\tilde{W}$ is an $n_x \times n_w$ real matrix. Then $W^TW = I_{n_x - n_w}$, $W^T\tilde{W} = 0$, $SW = I_{n_x - n_w}$, $ST = 0$, and $C_2W = 0$.*

Proof. The matrix $\bar{S}$ is upper triangular and of the same rank as R, i.e., $n_x - n_w$, and n_w of its diagonal elements are zero. The rows corresponding to these diagonal elements must be zero; otherwise, the rank of $\bar{S}$ will be higher than that of R. These empty rows can be partitioned away as the lower n_w rows of $\bar{S}$. Thus $R = (W \quad \tilde{W})\binom{S}{0} = WS$.

Since $\bar{W}$ is a unitary matrix, the columns of $\bar{W}$ are orthonormal. Hence $W^TW = I_{n_x - n_w}$. Since the columns of $\tilde{W}$ are orthogonal to those of W, $\tilde{W}^TW = 0$.

Since $RT = 0$ by construction and $S = W^TR$, we have $ST = 0$. Since $ST = 0$, we have $SR = S$, i.e., $SWS = S$, i.e., $SW = (SS^T)^{-1}(SS^T) = I_{n_x - n_w}$. The matrix SS^T is invertible because S is of full row $(n_x - n_w)$ rank. Since $W = RW$ and $C_2R = 0$ by construction, we have $C_2W = 0$. ∎

Next the effect of transformation with S and W on eigenvalues of a square matrix is examined.

LEMMA 3.6. *If $A \in \mathcal{R}^{n_x \times n_x}$ and S and W have been defined as above, then*

$$\lambda_{\max}(SAW) = \lambda_{\max}(A)k, \qquad 0 \leq |k| \leq 1, \tag{17}$$

or

$$\max_{\lambda} |\lambda(SAW)| \leq \max_{\lambda} |\lambda(A)|, \tag{18}$$

where $\lambda_{\max}$ is the eigenvalue having the maximum absolute value.

Proof. The eigenvalue equation for SAW is $SAWe - \lambda_0 e = 0$, where λ_0 is an eigenvalue of SAW and $e \in \mathcal{C}^{n_x - n_w}$ is an eigenvector for SAW. Let $e = W^T p$, where p is a unit vector of dimension n_x. The eigenvalue problem is rewritten as $SAWW^T p - \lambda_0 W^T p = 0$. This is equivalent to solving $RAWW^T p - \lambda_0 WW^T p = 0$. Since WW^T is an idempotent matrix and $W^T W = I_{n_x - n_w}$, WW^T can be decomposed as $Q_1 \begin{pmatrix} I_{n_x - n_w} & 0 \\ 0 & 0 \end{pmatrix} Q_1^T$, Q_1 being a $n_x \times n_x$ unitary matrix. Also, R is decomposed as $Q \begin{pmatrix} I_{n_x \times n_w} & \Xi \\ 0 & 0 \end{pmatrix} Q^T$. Substituting the decompositions, we obtain

$$Q_1^T Q \begin{pmatrix} I_{n_x \times n_w} & \Xi \\ 0 & 0 \end{pmatrix} Q^T A Q_1 \begin{pmatrix} I_{n_x - n_w} & 0 \\ 0 & 0 \end{pmatrix} Q_1^T p = \lambda_0 \begin{pmatrix} I_{n_x - n_w} & 0 \\ 0 & 0 \end{pmatrix} Q_1^T p,$$

$$Q_1^T Q \begin{pmatrix} I_{n_x \times n_w} & \Xi \\ 0 & 0 \end{pmatrix} Q^T A Q_1 \begin{pmatrix} v \\ 0 \end{pmatrix} = \lambda_0 \begin{pmatrix} v \\ 0 \end{pmatrix},$$

where v is a vector of dimension $n_x - n_w$ and $||v||_2 \leq 1$. A is decomposed as $Q_A N Q_A^T$, where Q_A is a unitary matrix and N is an upper triangular matrix. The eigenvalues λ_0 are then based on upper $n_x - n_w$ rows and columns of N. Thus $k \max(\lambda_0) = \max(\lambda(A))$ and $|\max(\lambda_0)| \leq \max(\lambda(A))$. ∎

COROLLARY 3.7. *In Lemma* 3.6, *if the transformation is* $W^T AW$ *instead of* SAW, *the results in* (18) *also hold true.*

The proof can be completed by substituting W^T for S. Since WW^T is idempotent, the eigenvalue equation for WW^T is $WW^T e - \lambda_{WW^T} e = 0$, i.e., $WW^T(1 - \lambda_{WW^T})e = 0$, where e is a unit vector of dimension n_x. Then $\lambda_{WW^T} = 1$ or $WW^T e = 0$. If $WW^T e = 0$, $\lambda_{WW^T} = 0$. If $\lambda_{WW^T} = 1$, then $WW^T e = e$ and $C_2 e = 0$. Since $\text{rank}(C_2) = n_w$, there will be exactly $n_x - n_w$ eigenvalues equal to 1. The other n_w eigenvalues are 0. Hence $0 \leq |p_{\max}^T WW^T p_{\max}| \leq 1$ (Lemma 2.1). Hence the results in (17) hold for a $W^T AW$ type transformation.

An EUODE can be deduced by substituting $x \in \mathcal{R}^{n_x}$ by a variable $\xi \in \mathcal{R}^{n_x - n_w}$. Let $\xi = Sx$. Then it follows from (3c) that $x = W\xi - T(C_2 T)^{-1} r_2$. It is assumed that R is at least once-differentiable (with respect to t) almost everywhere over the interval of integration ($t_0 \leq t \leq t_{\text{final}}$) and that R is of constant rank everywhere over the interval. Then W is differentiable at least once over the interval of integration. [8].

LEMMA 3.8. *If* $R \in \mathrm{C}^1(t)$, *then* $W \in \mathrm{C}^1(t)$ *and* $S \in \mathrm{C}^1(t)$.

Proof. Since R has constant rank all over the interval $t_0 \leq t \leq t_{\text{final}}$, Corollary 2.5 in [8] can be applied to R to get a QR-decomposition differentiable at least once with respect to t over the interval of integration. ∎

The UODE can be written in terms of ξ and premultiplied by S to obtain an EUODE

$$\dot{\xi} = -S\dot{W}\xi + S(H - QC_v^{-1}C_1)W\xi + \{\text{terms in } t\}. \tag{19}$$

The stability of the EUODE is then determined by the largest real part of the eigenvalues of

$$M := S(H - QC_v^{-1}C_1)W - S\dot{W}. \tag{20}$$

Let $\bar{H} := H - QC_v^{-1}C_1$. Then

$$M := W^T R\bar{H}W - W^T R\dot{W} = W^T \bar{H}W - W^T T(C_2 T)^{-1} C_2 \bar{H}W \\ -W^T \dot{W} + W^T T(C_2 T)^{-1} C_2 \dot{W}.$$

The logarithmic norm can be expressed as

$$\mu(M) = \mu(W^T\bar{H}W - W^TT(C_2T)^{-1}C_2\bar{H}W - W^T\dot{W} + W^TT(C_2T)^{-1}C_2\dot{W}).$$

Since $W^TW =$ a constant (identity) matrix and $\dot{W}^TW + W^T\dot{W} = \dot{I} = 0$, using (5), we have

$$\mu(M) = \mu(W^T\bar{H}W - W^TT(C_2T)^{-1}C_2\bar{H}W + W^TT(C_2T)^{-1}C_2\dot{W}). \tag{21}$$

Using the corollary to Lemma (3.6), the definition (5), and that $C_2W = 0$, we get

$$\begin{aligned}\mu(M) &= \mu(W^T\bar{H}W - W^TT(C_2T)^{-1}C_2\bar{H}W - W^TT(C_2T)^{-1}\dot{C}_2W)\\ &\le \mu(R\bar{H} - T(C_2T)^{-1}\dot{C}_2).\end{aligned} \tag{22}$$

The last expression on the right is the logarithmic norm for the stability matrix of the UODE (16). Thus checking $\mu(R\bar{H} - T(C_2T)^{-1}\dot{C}_2) \le 0$ is sufficient for the stability of the index 2 pDAE. This implies that if the index 1 differentiated form of the constraint yields a stable pDAE in terms of the logarithmic norm measure, the index 2 system is then stable too. In (21), $\mu(M) \le \mu(W^T\bar{H}W) \le \mu(\bar{H})$ when $T = C_2^T$, i.e., when $W^TT = 0$. Thus index 2 constraints with $T = C_2^T$ can be appended to the pDAE without destabilizing it. Also, when $(C_2\bar{H} + \dot{C}_2)R = 0$, the index 2 constraint does not change the stability of the existing pDAE.

Using (20) and Lemma 3.6, we obtain

$$\max(\Re(\lambda(S\bar{H}W - S\dot{W}))) = \max(\Re(\lambda(S\bar{H}W + SW\dot{S}W))),$$

leading to

$$\begin{aligned}&\max(\Re(\lambda(S(\bar{H} + W\dot{S})W)))\\ &= \max(\Re(\lambda(S(\bar{H} + W\dot{W}^T - WW^TT(C_2T)^{-1}\dot{C}_2)W)))\end{aligned} \tag{23}$$

and

$$\max(\Re(\lambda(S\bar{H}W + \dot{S}W))) \le \max(\Re(\lambda(\bar{H} + W\dot{W}^T - WW^TT(C_2T)^{-1}\dot{C}_2))).$$

For $T \ne C_2^T$ a computable criterion for the stability of the index 2 pDAE can be derived. Using (8), we can write

$$\begin{aligned}&\max(\Re(\lambda(\bar{H} + W\dot{W}^T - WW^TT(C_2T)^{-1}\dot{C}_2)))\\ &\le \mu(\bar{H} + W\dot{W}^T - WW^TT(C_2T)^{-1}\dot{C}_2).\end{aligned} \tag{24}$$

The use of the triangle inequality (6a) on (24) separates the index 2 logarithmic norm terms from the index 1 subsystem of the pDAE yielding

$$\mu(\bar{H} + W\dot{W}^T - WW^TT(C_2T)^{-1}\dot{C}_2) \le \mu(\bar{H}) + \mu(W\dot{W}^T) + \mu(-WW^TT(C_2T)^{-1}\dot{C}_2).$$

The term $\mu(W\dot{W}^T)$ can be bounded as follows.

LEMMA 3.9. *With W defined as above, $\mu(-T(C_2T)^{-1}\dot{C}_2) \ge \mu(W\dot{W}^T) \ge 0$.*

Proof. The following relationship can be derived.

$$\begin{aligned}\mu(W\dot{W}^T) &= 0.5\max_{\lambda}\lambda(W\dot{W}^T + \dot{W}W^T) = 0.5\mu(\overline{\dot{WW^T}})\\ &= 0.5\mu(-\overline{\dot{\tilde{W}}\tilde{W}^T}) = \max_{\lambda}\lambda\left(-\frac{\dot{\tilde{W}}\tilde{W}^T + \tilde{W}\dot{\tilde{W}}^T}{2}\right),\end{aligned}$$

where $\tilde{W}$ is a part of the partitioned matrix $\bar{W}$ as described in Lemma (3.5) and $\dot{\tilde{W}\tilde{W}^T} = \frac{d}{dt}(\tilde{W}\tilde{W}^T)$. Thus

$$
\begin{aligned}
&\mu(W\dot{W}^T)\\
&= \max_{\lambda} \lambda \left(-0.5 \begin{pmatrix} W^T \\ \tilde{W}^T \end{pmatrix} (\dot{\tilde{W}}\tilde{W}^T + \tilde{W}\dot{\tilde{W}}^T) \begin{pmatrix} W & \tilde{W} \end{pmatrix} \right)\\
&= \max_{\lambda} \lambda \left(-0.5 \begin{pmatrix} 0 & W^T\dot{\tilde{W}} \\ (W^T\dot{\tilde{W}})^T & \tilde{W}^T\dot{\tilde{W}} + \dot{\tilde{W}}^T\tilde{W} \end{pmatrix} \right)\\
&= \max_{\lambda} \lambda \left(-0.5 \begin{pmatrix} 0 & W^T\dot{\tilde{W}} \\ (W^T\dot{\tilde{W}})^T & 0 \end{pmatrix} \right)\\
&= \max_{\lambda} \lambda \left(-0.5 \begin{pmatrix} 0 & W^T(T(C_2T)^{-1}\dot{C}_2)^T\tilde{W} \\ \tilde{W}^T(T(C_2T)^{-1}\dot{C}_2)W & 0 \end{pmatrix} \right)\\
&= \max_{\lambda} \lambda \left(-0.5 \begin{pmatrix} W & \tilde{W} \end{pmatrix} \begin{pmatrix} 0 & W^T(T(C_2T)^{-1}\dot{C}_2)^T\tilde{W} \\ \tilde{W}^T(T(C_2T)^{-1}\dot{C}_2)W & 0 \end{pmatrix} \begin{pmatrix} W^T \\ \tilde{W}^T \end{pmatrix} \right)\\
&= \max_{\lambda} \lambda \left(-0.5(\tilde{W}\tilde{W}^T(T(C_2T)^{-1}\dot{C}_2)^TWW^T + (\tilde{W}\tilde{W}^T(T(C_2T)^{-1}\dot{C}_2)^TWW^T)^T) \right)\\
&= \mu(-\tilde{W}\tilde{W}^T(T(C_2T)^{-1}\dot{C}_2)WW^T)\\
&= \mu(-(T(C_2T)^{-1}\dot{C}_2)WW^T + WW^T(T(C_2T)^{-1}\dot{C}_2)WW^T).
\end{aligned}
$$

The above derivation uses the preservation property of eigenvalues under a unitary transformation (Lemma 2.3) and the following relationships from the definition of W, $\tilde{W}$, and R as given in the last section.

$$
\begin{aligned}
\tilde{W}^T R = \tilde{W}^T - \tilde{W}^T(T(C_2T)^{-1}C_2) = 0,\\
\dot{\tilde{W}}^T = \dot{\tilde{W}}^T(T(C_2T)^{-1}C_2) + \tilde{W}^T\frac{d(T(C_2T)^{-1})}{dt}C_2 + \tilde{W}^TT(C_2T)^{-1}\dot{C}_2,\\
\dot{\tilde{W}}^TW = \tilde{W}^TT(C_2T)^{-1}\dot{C}_2W.
\end{aligned}
$$

Using (5) the logarithmic norm can be evaluated through the eigenvalue equation for

$$-\tilde{W}\tilde{W}^T(T(C_2T)^{-1}\dot{C}_2)WW^Te - WW^T(T(C_2T)^{-1}\dot{C}_2)^T\tilde{W}\tilde{W}^Te - 2\lambda e = 0,$$

where λ is the eigenvalue and the unit vector e is the corresponding eigenvector. Also, C_2 and $\tilde{W}$ being at least of rank n_w, $C_2\tilde{W} \in \mathcal{R}_{n_w \times n_w}$ is invertible so that

$$\tilde{W}^T\tilde{W} = \tilde{W}^TT(C_2T)^{-1}C_2\tilde{W} = I_{n_w},$$

whence

$$(C_2\tilde{W})^{-1} = \tilde{W}^TT(C_2T)^{-1}.$$

Since $C_2W = 0$, after premultiplying the above by C_2, we have $2\lambda C_2e = -\dot{C}_2WW^Te$. We can substitute this in the eigenvalue equation, and, by using $\tilde{W}^T = \tilde{W}^TT(C_2T)^{-1}C_2$, we get

$$\lambda e^T\tilde{W}\tilde{W}^TT(C_2T)^{-1}C_2e = 0.5\lambda,$$

leading to

$$\lambda e^T\tilde{W}\tilde{W}^Te = 0.5\lambda,$$

whence either $\lambda = 0$ or $e^T\tilde{W}\tilde{W}^T e = 0.5$ (or $e^T WW^T e = 0.5$). Corresponding to the latter relationships, $\mu(W\dot{W}^T)$ is the logarithmic norm of a partial matrix obtained from orthogonal transformation of the matrix $(-T(C_2T)^{-1}\dot{C}_2)$, padded with zeros. Both WW^T and $\tilde{W}\tilde{W}^T$ are matrices with eigenvalues at 0 and 1 only. Hence their Schur decompositions can be written as $WW^T = Q_0\begin{pmatrix} I_{n_w \times n_x} & 0 \\ 0 & 0 \end{pmatrix} Q_0^T$ and $\tilde{W}\tilde{W}^T = Q_0\begin{pmatrix} 0 & 0 \\ 0 & I_{n_w} \end{pmatrix} Q_0^T$, respectively. Then

$$\mu(-\tilde{W}\tilde{W}^T(T(C_2T)^{-1}\dot{C}_2)WW^T)$$
$$= \mu\left(-\begin{pmatrix} 0 & 0 \\ 0 & I_{n_w} \end{pmatrix} Q_0(T(C_2T)^{-1}\dot{C}_2)Q_0^T \begin{pmatrix} I_{n_w \times n_x} & 0 \\ 0 & 0 \end{pmatrix}\right) \leq \mu(-T(C_2T)^{-1}\dot{C}_2).$$

Hence $\mu(W\dot{W}^T) \leq \mu(-T(C_2T)^{-1}\dot{C}_2)$.

Differentiating both sides in (6d), we get the property by which

$$\mu(W\dot{W}^T) = 0.5\mu(\dot{\overline{WW^T}}) \geq \dot{\mu}(WW^T) = 0. \qquad \square$$

Since $W^TW = I_{n_w \times n_x}$ and WW^T is an idempotent matrix with eigenvalues of 0 and 1 only, the matrix WW^T has a Schur decomposition of the form $Q_0\begin{pmatrix} I_{n_w \times n_x} & 0 \\ 0 & 0 \end{pmatrix} Q_0^T$. Using the corollary to Lemma 3.6, we can estimate $\mu(-WW^TT(C_2T)^{-1}\dot{C}_2) \leq \mu(-T(C_2T)^{-1}\dot{C}_2)$. Hence we get

$$\mu(\bar{H}) + \mu(W\dot{W}^T) + \mu(-WW^TT(C_2T)^{-1}\dot{C}_2) \leq \mu(H) + 2\mu(-T(C_2T)^{-1}\dot{C}_2). \tag{25}$$

If C_2 is constant, the index 2 constraint does not destabilize the existing pDAE significantly.

Applying the triangle inequality of the logarithmic norm to (25), we have obtained the stability estimate for index 2 pDAE as $\mu(\bar{H})+\mu(W\dot{W}^T)+\mu(-WW^TT(C_2T)^{-1}\dot{C}_2)$, which has been shown to be less than or equal to $\mu(\bar{H}) + 2\mu(-T(C_2T)^{-1}\dot{C}_2)$. Using (17), $\max(\Re(\lambda(S\bar{H}W + \dot{S}W)))$ can be estimated as follows. Since for all practical purposes H is the dominant matrix term in $\bar{H}$, k as described in (17) is estimated as $p_{H,\max}^T R p_{H,\max}$, where $p_{H,\max}$ is the normalized eigenvector corresponding to the largest eigenvalue of H. Then, applying Lemma 2.4, we obtain

$$\begin{aligned}
&\max(\Re(\lambda(S(\bar{H} + W\dot{S})W))) \leq \max(|\lambda(S(\bar{H} + W\dot{S})W)|) \\
&= |k\lambda_{\max}(\bar{H} + W\dot{S})| = |k||\lambda_{\max}(\bar{H} + W\dot{S})| \leq |k|\mu(\bar{H} + W\dot{S}) \\
&= |k|(\mu(\bar{H}) + 2\mu(-T(C_2T)^{-1}\dot{C}_2)).
\end{aligned}$$

Thus we have shown the following theorem.

THEOREM 3.10 (index 2 stability). *The index* 2 *pDAE is stable if*

$$\mu(\bar{H}) + 2\mu(-T(C_2T)^{-1}\dot{C}_2) \leq 0.$$

For practical computations we take $|k| \approx 1$ (when $0 < |k| \leq 1$) and require

$$\mu(\bar{H}) + 2\mu(-T(C_2T)^{-1}\dot{C}_2) \leq \frac{TOL}{|k|} = TOL \tag{26}$$

for the index 2 subsystem to be stable.

Any index 2 algebraic constraint which is a candidate for being appended to the already constituted pDAE system is thus checked for $T = C_2^T$, C_2 constant, and for $(C_2\bar{H} + \dot{C}_2)R = 0$, failing which the check (26) is done.

In the case of a single index 2 constraint being checked for stability, an analytical expression for $\mu(-T(C_2T)^{-1}\dot{C}_2)$ is obtained using the result in Lemma 2.5. In this case, C_2 is a row vector (denoted by $\tilde{C}_2$), T is a column vector (denoted by $\tilde{T}$), and $R = I - \frac{u_T u_{C_2}^T}{u_{C_2}^T u_T}$, where $u_{C_2} = \tilde{C}_2^T/||\tilde{C}_2||_2$ and $u_T = \tilde{T}/||\tilde{T}||_2$. Then $||S|| = ||R|| = \frac{1}{u_{C_2}^T u_T}$. The change in the stability measure by appending a single index 2 constraint can be estimated via

$$\mu\left(-\frac{u_T\dot{\tilde{C}}_2}{\tilde{C}_2 u_T}\right) = \max\left(\frac{-\text{sign}(\tilde{C}_2\tilde{T})\dot{\tilde{C}}_2\tilde{T} \pm ||\dot{\tilde{C}}_2||_2||\tilde{T}||_2}{2|\tilde{C}_2\tilde{T}|}, 0\right) \tag{27}$$

for $n_x > 2$. For $n_x = 2$, the maximum is taken over $\frac{-\text{sign}(\tilde{C}_2\tilde{T})\dot{\tilde{C}}_2\tilde{T} \pm ||\dot{\tilde{C}}_2||_2||\tilde{T}||_2}{2|\tilde{C}_2\tilde{T}|}$ only. The term $\dot{C}_2$ is obtained through automatic differentiation of the coefficient matrix C_2 in the linearized constraint equations. Further, the term $\mu(W\dot{W}^T)$, i.e., $0.5\mu(-\dot{\overline{\tilde{W}\tilde{W}^T}})$, can be analytically evaluated for a single index 2 constraint. From the property of $\tilde{W}$ (refer to Lemma 3.9), for a single constraint it can be constructed as a unit column vector: $\tilde{W} = \frac{\tilde{C}_2^T}{||\tilde{C}_2||_2} = u_{C_2}$. Let $\dot{u}_{C_2} = \gamma_2\bar{u}_{C_2}$ (recall that the rate of change of a unit vector is orthogonal to it), where $\bar{u}_{C_2}^T u_{C_2} = 0$ and $\gamma_2 = ||\dot{u}_{C_2}||_2$. Hence, using the result in Lemma 2.5,

$$\mu(W\dot{W}^T) = 0.5\mu(-\dot{\overline{\tilde{W}\tilde{W}^T}}) = 0.5\mu(-\dot{\overline{u_{C_2}u_{C_2}^T}}) = \gamma_2/2 = 0.5||\dot{u}_{C_2}||_2. \tag{28}$$

Putting (28) and (27) together, (26) can be revised for a single index 2 constraint. Thus we have shown that the following holds.

THEOREM 3.11 (stability for index 2 constraints). *Checking*

$$\mu(\bar{H}) + 0.5||\dot{u}_{C_2}||_2 + \max\left(\frac{-\text{sign}(\tilde{C}_2\tilde{T})\dot{\tilde{C}}_2\tilde{T} \pm ||\dot{\tilde{C}}_2||_2||\tilde{T}||_2}{2|\tilde{C}_2\tilde{T}|}, 0\right) \le \frac{TOL}{|k|} \tag{29}$$

(for $n_x > 2$) is sufficient for determining the stability when including a single index 2 algebraic constraint in the pDAE.

3.3. A practical criterion for constraint inclusion. For some types of systems, in particular, nonlinear highly oscillatory systems such as the stiff spring pendulum and wheelset on rails examples described later, the logarithmic norm of the original system may already be large. Thus the criteria (14) and (29) would partition all constraints out of the DAE, even if they do not further increase the logarithmic norm. Hence in practice we relax the criteria to require only that the constraints do not increase the logarithmic norm by much. Thus, instead of checking simply if $\mu \le TOL$, the stability check for accepting a constraint is modified as

$$\mu \le \max\left(\frac{TOL}{|t_{\text{final}} - t_0|}, 3\ TOL_0\right), \tag{30}$$

where TOL_0 is the logarithmic norm of the unconstrained dynamic system.

3.4. Stability of the nonlinear pDAE system. For nonlinear systems, the stability can be examined only in the neighborhood of $t = t_k$, i.e., at the beginning of a shooting (integration) subinterval. The logarithmic norm can be used to get an estimate of the stability of the nonlinear system. Consider an index 2 pDAE:

$$\dot{x} = f(x, w, u_2(t), t),$$

$$\text{where} \quad f := h(x,t) + \varphi(t, x, w, u_2(t)), \tag{31a}$$

$$\Psi_2(t, x) = 0. \tag{31b}$$

Let the UODE (16) be rewritten in the form

$$\dot{x} = h(x,t) + \varphi(t, x, w(x,t), u_2(t)), \tag{32}$$

where $w(x,t) = -(C_2T)^{-1}(C_2Hx - C_2QC_v^{-1}C_1x - C_2QC_v^{-1}r_1(t) + C_2r(t) + \dot{C}_2x + \dot{r}_2(t))$. Then the largest real part in the eigenvalues of the matrix

$$M_0 := \frac{\partial h}{\partial x} + \frac{\partial \varphi}{\partial x} - \varphi_{,w}\left(\Psi_{2,x}\varphi_{,w}\right)^{-1}\left(\frac{\partial\left(\Psi_{2,x}h + \Psi_{2,t}\right)}{\partial x} + \Psi_{2,x}\varphi_{,x}\right)$$

gives an estimate of the stability of the UODE (32). Using the properties of the logarithmic norm [11], we get

$$\max_{\lambda} \quad \Re(\lambda(M_0)) \le \mu\left(\frac{\partial h}{\partial x}\right) + \mu\left(\frac{\partial \varphi}{\partial x} - \varphi_{,w}\left(\Psi_{2,x}\varphi_{,w}\right)^{-1}\left(\frac{\partial\left(\Psi_{2,x}h + \Psi_{2,t}\right)}{\partial x} + \Psi_{2,x}\varphi_{,x}\right)\right).$$

Let D be defined as

$$D := -\varphi_{,w}\left(\left(\Psi_{2,x}\varphi_{,w}\right)^{-1}\left(\frac{\partial\left(\Psi_{2,x}h + \Psi_{2,t}\right)}{\partial x} + \Psi_{2,x}\varphi_{,x}\right)\right).$$

Then the above relationship gives $\max_\lambda \Re(\lambda(M_0)) \le 0$ when $\mu(\frac{\partial h}{\partial x}) \le -\mu(\frac{\partial \varphi}{\partial x} + D)$. If at $t = t_k$, $\mu(\frac{\partial h}{\partial x}) + \mu(\frac{\partial \varphi}{\partial x} + D) \le 0$, then the index 2 pDAE is stable at the beginning of the shooting subinterval.

If $\tilde{x}$ is an approximate solution to the UDOE (32), then by Theorem I10.6 in [11], given

$$\mu\left(\frac{\partial h}{\partial x}(t,\varsigma) + \frac{\partial \varphi}{\partial x}(t,\varsigma)\right) \le -\mu(D), \qquad \varsigma \in [x, \tilde{x}],$$

$$||\dot{\tilde{x}} - f||_2 \le \delta(t),$$

$$||x(t_k) - \tilde{x}(t_k)||_2 \le \rho,$$

it can be stated that for $t \ge t_k$,

$$||x(t) - \tilde{x}(t)||_2 \le e^{L(t)}\left(\rho + \int_{t_k}^{t} e^{-L(\tau)}\delta(\tau)d\tau\right), \qquad t_k \le t \le t_{k+1}.$$

For the UODE (32) to be stable, it is sufficient that $L(t) \le 0$. This is satisfied if

$$L(t) = -\int_{t_k}^{t} \mu(D)(\tau)d\tau \le 0, \qquad t_k \le t \le t_{k+1}. \tag{33}$$

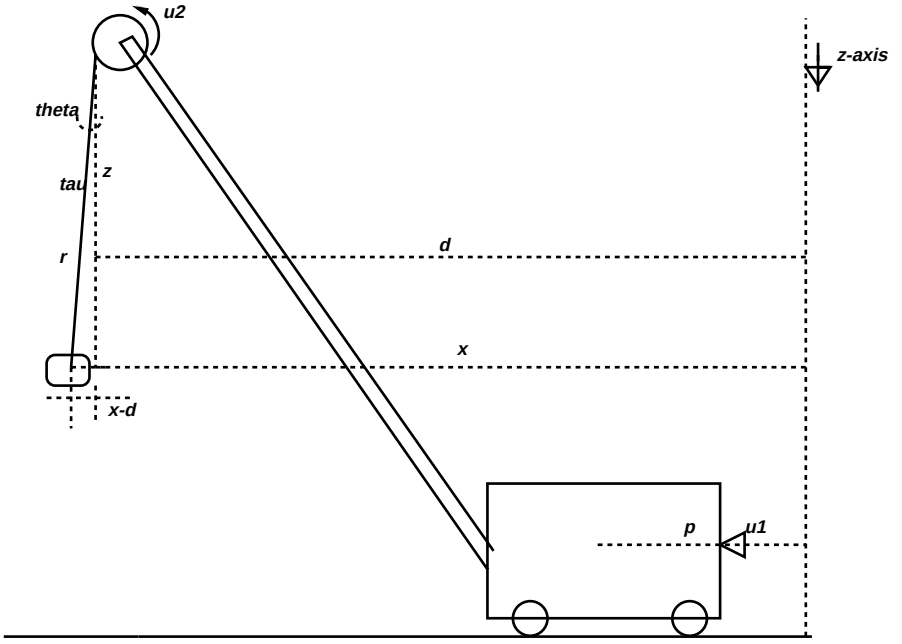


FIG. 4.1. *Planar crane model.*

This condition can be used to get an estimate over the entire interval of integration of the nonlinear pDAE as compared to a local estimate at a particular t via the logarithmic norm of the linearized system, although we do not pursue this further here.

4. Physically important scleronomic constraints. Scleronomic constraints describe additional physical or geometrical features of the dynamics system. In mechanics, these constraints are defined as algebraic kinematic equations of the form $\Psi_3(x) = 0$, in which time does not appear explicitly. Beyond the stability of the pDAE, constraints describing physical or geometrical relationships among the state variables play an important role in the general well-conditioning of the dynamic optimization problem. The preservation of these constraints is important in order to obtain physically meaningful results. Since many modern optimization methods are infeasible methods (i.e., the constraints handled by the optimization method are satisfied only upon convergence of the optimizer), the integration of the pDAE without these constraints may produce unphysical results.

4.1. Example: Planar model of a crane. This example is a planar rigid body model of a crane manipulating a payload, which must be lifted along a specified trajectory [6]. The problem is illustrated schematically in Figure 4.1. The crane is a mechanical system in planar Cartesian coordinates, x and z, denoting the horizontal and vertical positions of the payload. The other state variables are d, the horizontal distance traveled by the crane trolley from the origin, and r, the paid out cable length. The equations are

$$M_2\ddot{x} = -\tau \sin(\theta), \tag{34a}$$

$$M_2\ddot{z} = -\tau \cos(\theta) + mg, \tag{34b}$$

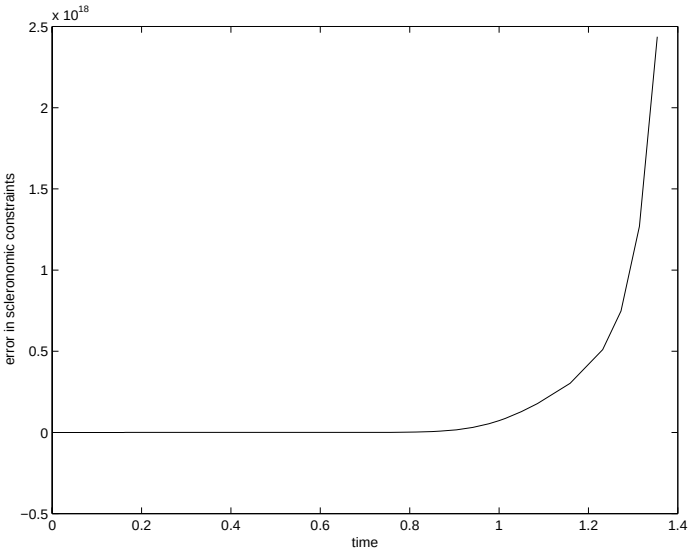


FIG. 4.2. *Partial plot of errors neglecting scleronomic constraint.*

$$M_1\ddot{d} = -C_1\dot{d} + u_1 + \tau\sin(\theta), \tag{34c}$$

$$J\ddot{r} = -C_2\dot{r} - C_3u_2 + C_3^2\tau, \tag{34d}$$

$$0 = \theta - \tan^{-1}((x-d)/z), \tag{34e}$$

$$0 = r^2 - (x-d)^2 - z^2, \tag{34f}$$

$$x = \phi_1(t), \tag{34g}$$

$$z = \phi_2(t), \tag{34h}$$

where M_1, M_2, and m are the masses of the trolley, cable, and payload system and the payload alone, respectively. C_1, C_2, and C_3 are constants, and J is the mass moment of inertia for the pulley paying out the cable. The algebraic variables are τ, the tension in the cable, and θ, the cable angle with vertical. The horizontal force driving the trolley (u_1) and torque driving the pulley in the winch (u_2) are the two control variables. Equations (34g) and (34h) describe the specified path of the payload.

Figure 4.2 illustrates the error ($\int_{t0}^{t_{\text{final}}} ||r^2 - (x-d)^2 - z^2)||_2 dt$) in simulating the planar crane problem with an index 1 pDAE consisting of (34a)–(34e) only and excluding equation (34f). The exclusion of (34f) physically means violation of the condition that the load is tied to the pulley by a cable of finite length measured as the square root of the sum of the vertical and horizontal deflections squared.

While the physical constraints inherently should be included with the pDAE, they can raise the index of the pDAE to higher than 2. As an example, (34f) in the crane example when included in the pDAE (34a)–(34e) raises the index of the system to 3. In cases where the constraints raise the index of the pDAE to higher than 2, a suitable index reduction method can be adopted. If the kinematic degrees of freedom of the physical system is known, then the number of differential equations *minus* the number of kinematic degrees of freedom gives the number of physical constraints which must be included in the pDAE.

A measure for identifying a scleronomic constraint for possible inclusion in the pDAE may be introduced. If a scleronomic constraint Ψ_3 which raises the index of the

pDAE to $q+2$ is being introduced into an index 2 pDAE via differentiating q times (written as $\Psi_3^{(q)}$), then the following estimate for the possible shift in eigenvalues at $t = t_k$ can be used:

$$\mu(\bar{H}) + 2\mu(-T(\Psi_{3,x}^{(q)}T)^{-1}\dot{\Psi}_{3,x}^{(q)}) \leq \frac{TOL_n}{|k|}. \tag{35}$$

The above relationship can be obtained by substituting $\Psi_{3,x}^{(q)}$ for C_2 in (26). The tolerance TOL_n is chosen as per the practical criterion for constraint inclusion given in section 3.3.

5. The partitioning algorithm. Based on considering one algebraic constraint at a time, a partitioning algorithm that returns a locally (at the point of linearization) index 2 or lower pDAE is designed. Constraints are examined for their impact on stability. In this section we discuss the essential components.

5.1. Design of the algorithm. Our strategy for partitioning for stability is to examine one constraint at a time for its effect on stability via criterion (14) and (29). First we examine index 1 constraints and then index 2 constraints. Finally, a check-back of blocks of constraints is done to ensure full system stability.

5.1.1. Partitioning of index 1 and index 2 subsystems. When the constraint under consideration has the option of choosing from more than one algebraic variable to which it can be linked, the search for the dominant algebraic variable corresponding to the constraint must be done in an economical way. The direct way of associating the possible variables is to mark all the possible variables and then go on to the next equation and so on. After the algebraic variables have been identified, all possible combinations of the variables are explored, and the one that gives the pDAE with the lowest logarithmic norm is associated with the index 1 subsystem. For large systems this method will be computationally very expensive. Hence a trade-off policy is adopted. The following explains the policy for both index 1 and index 2 systems.

Policy 1. For small systems, all possible algebraic variables corresponding to the current algebraic equation under consideration for entry into the index 1 or index 2 subsystem are tried out. Thus for all algebraic variables the logarithmic norm for the index 1 or index 2 subsystem (up to and including the current algebraic equation) is evaluated. The one that gives the lowest logarithmic norm is associated with the index 1 or index 2 constraint under consideration. This is feasible only for very small systems since a complete search involves exponential computational complexity.

Policy 2. For a very large system with many coupled terms across the algebraic variables, the above policy may prove to be too expensive. In those cases, the policy is to look for a variable "moderately connected" with the differential variables in the current equation. This can be done by directly starting the search for a nonzero scalar $\frac{||\tilde{C}_1||_2}{c_v} \leq 0.5(\min_v(\frac{||\tilde{C}_1||_2}{c_v}) + \max_v(\frac{||\tilde{C}_1||_2}{c_v}))$ (for index 2, substitute $\tilde{C}_2\tilde{T}$ in place of c_v), where the minimum and the maximum are taken over the possible algebraic variables in the equation being considered. If this choice yields an unacceptable logarithmic norm, the next higher c_v ($\tilde{C}_2\tilde{T}$) is considered and so on. Since the logarithmic norm is inversely proportional to c_v ($\tilde{C}_2\tilde{T}$ for index 2), the larger ratios need not be explored. The scalars c_v or $\tilde{C}_2\tilde{T}$ may be suitably scaled over the range of their values. This policy was implemented in the test program that produced the results in section 6.

5.1.2. Rejecting constraints during check back for stability. The algorithm has the limitation that during stability analysis the effect of the terms (corresponding to the new algebraic variable being included in the pDAE) coupling existing

constraints and the constraint under consideration are ignored. Hence, after accepting a few constraints, a check back for the overall stability of the pDAE is needed. While performing the local stability check for a single algebraic equation, the new algebraic variable corresponding to this equation must be picked in such a way that the local stability check is a fair estimate of the stability of the pDAE and that the global check pointing will not change the built-up pDAE unless absolutely necessary.

If it is found during checking back that the global logarithmic norm of the pDAE violates the practical criterion (30), then some of the constraints must be rejected to keep the system stable. In keeping with Policy 2, for index 1 constraints we leave out the last appended constraint and the algebraic variable having the maximum $\frac{||\tilde{C}_1||_2}{|c_v|}$ (scaled over the range of values c_v takes in the problem), and we check for overall stability in terms of the recomputed overall logarithmic norm. If the system is still unstable, the procedure is repeated on the next constraint on the block of constraints being checked. This is done until a stable index 1 pDAE is found. For an index 2 system, the same thing is done on the constraint having the largest $||\dot{\tilde{C}}_2||_2/|\tilde{C}_2\tilde{T}|$ in the block of constraints being checked, and variables with maximum $||\dot{\tilde{C}}_2||_2/|\tilde{C}_2\tilde{T}|$ are rejected.

6. Examples of partitioning. In this section we illustrate the partitioning strategy with several examples. The results are shown only at a representative multiple shooting node. For a DAE that changes its type interval to interval, the application of the partition algorithm at the beginning of the interval automatically decides the subsystem that can be integrated directly and the constraints and variables to be handled from the optimizer.

6.1. Partitioning the crane model. For the crane model described earlier, constraints (34e) and (34f) are both scleronomic constraints, i.e., they describe some geometric structure of the physical problem. Constraint (34e) introduces the measurement of θ, the angle by which the payload deviates from the vertical. Constraint (34f) connects the payload to the pulley by relating its horizontal and vertical positions to the length of the cable paid out. Violation of constraint (34f) in the pDAE would produce an unphysical simulation since the payload is now modeled as detached from the cable. This is evident from the plot of x, z, d, and r in Figure 6.1.

Applying the algorithm to this model, constraint (34e) is included in the pDAE since it satisfies the index 1 structure and stability criterion. The resulting UODE in x does not have large eigenvalues in $\mathcal{C}^+$ at $t = t_0$. For a practical physical application, the eigenvalues of the pDAE system can be expected to be of moderate size, and the system can be expected to be stable. The constraint equation (34f) is an index 3 scleronomic constraint (known from the physics of the problem) and is reduced to its index 2 form via one differentiation of constraint (34f) with respect to t,

$$r\dot{r} - (x - d)(\dot{x} - \dot{d}) - z\dot{z} = 0, \tag{36}$$

or to index 1 via two differentiations of constraint (34f) with respect to t (if the available DAE integration software can integrate index 1 or 0 systems only). The other constraints (the preprogrammed trajectory of the payload, (34g), and (34h)) are handled from the SQP method.

In the crane model the partition algorithm detects and accepts the index 1 constraint (34e) at $t = 0$. The value of TOL was set at 20.0. The interval of integration is from $t_0 = 0$ to $t_{\text{final}} = 8$. The value of TOL_0 in (30) was set at 0. The initial values

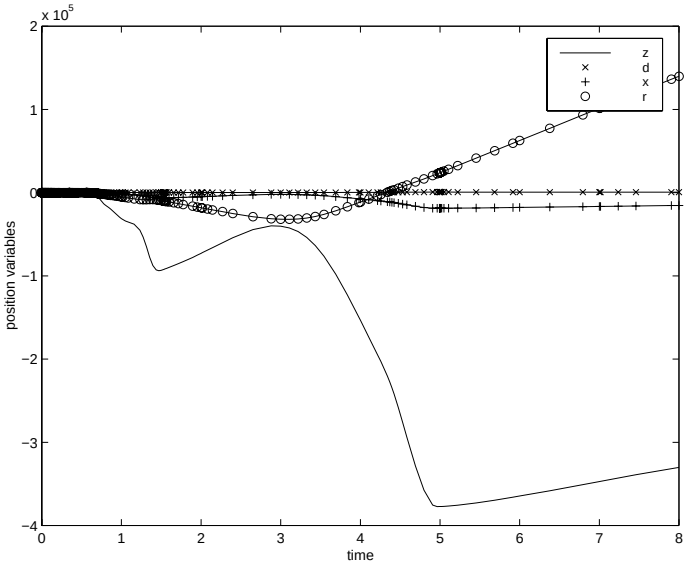


FIG. 6.1. *Crane problem using equations* (34a)–(34e) *in the pDAE.*

are $x = 3.0$, $\dot{x} = 2.7$, $z = 21.0$, $\dot{z} = -2.0$, $d = 1.0$, $\dot{d} = 2.7$, $r = ((x-d)^2 + z^2)^{0.5}$, $\dot{r} = -2.0$, $\tau = 980.0$, $u_1 = 0$, $u_2 = 0$, and $\theta = \tan^{-1}(\frac{(x-d)}{z})$. The logarithmic norm of the stability matrix H for the crane model at $t = 0$ is 0.5, and with the addition of the index 1 constraint (34e) it goes up to 0.82794. This is less than $\max(TOL/8, 3TOL_0)$ (see (30)), and the constraint (34e) is accepted. The angle θ of deflection from the vertical is chosen as the index 1 variable.

The search for index 2 constraints returns the empty set. In order to search for index 3 scleronomic constraints, the remaining algebraic equations are differentiated using automatic differentiation once with respect to time. The index 2 search algorithm is reapplied, and the algorithm returns the differentiated form of constraint (34f), i.e., (36), as an index 2 constraint suitable for inclusion in the pDAE system. Here $|k| = 0.9999$. The logarithmic norm of the index 2 pDAE stability matrix (using (29)) is 1.61502, and the tension in the cable τ is returned as the corresponding index 2 variable. The matrix S from the stability analysis changes very slowly in the neighborhood of $t = 0$ since $T = (0, 0, 0, 0, -\sin(\theta), -\cos(\theta), \sin(\theta), C_3^2)^T$ remains almost unchanged due to very small changes in $\theta \approx 0$ (i.e., an almost vertical cable undergoing only small swings). Hence the logarithmic norm changes only slightly even after the constraint (36) has been appended to the pDAE in its differentiated form. During check back, the overall logarithmic norm of the accepted pDAE system is computed as 1.09717, which is smaller than $\max(TOL/8, 3TOL_0)$ and is acceptable. Building the whole system takes 9.82×10^{-2} CPU seconds on a 180 MHZ IP32 processor SGI O2 computer.

The crane model yields the same partitioning result when the algorithm is applied at $t = 2, 4$, and 6 seconds.

Even though the constraint (36) slightly raises the logarithmic norm estimate for the index 2 pDAE system thus constructed, this partition leads to a more physical and well-conditioned problem. The plots in Figures 6.2 and 6.3 are the results obtained from a multiple shooting-type scheme using DASPK3.0 [15] as the DAE integrator

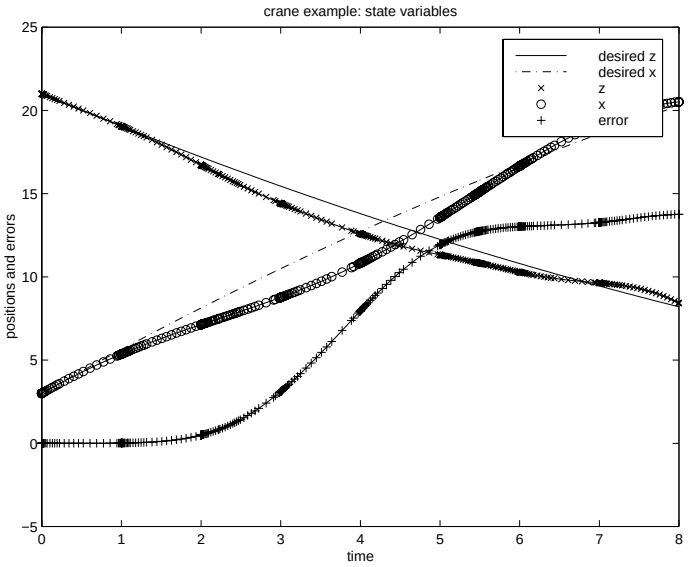


FIG. 6.2. *Crane problem using equations* (34a)–(34e) *and* (36) *in the pDAE.*

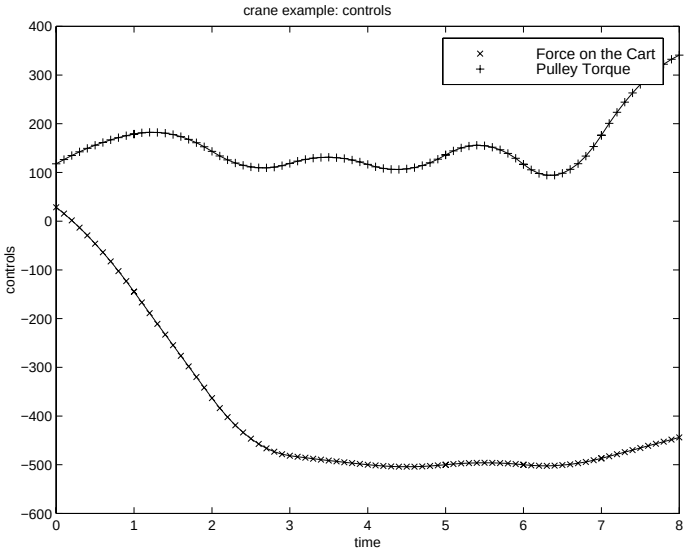


FIG. 6.3. *Crane controls using equations* (34a)–(34e) *and* (36) *in the pDAE.*

and SNOPT as the NLP method [16]. In this case, the path constraints are chosen as $p_1 \equiv x - (-0.0675t^2 + 2.7t + 3.0) = 0$ and $p_2 \equiv z - (-21.0 - 0.05t^2 + 2.0t) = 0$. The optimizer SNOPT stops at an acceptable point which cannot be improved further after three major iterations (with 188, 182, 112, and 1 minor iterations). The objective function is given by the error integral $\int_{t_0}^{t_{\text{final}}} (p_1^2 + p_2^2)^{0.5} dt$. The error integral has been reduced to 13.76067. This corresponds to a reduction in the SNOPT merit function from $1.79271101E + 10$ to $1.25966891E + 07$. The time interval was $0 \leq t \leq 8$. The controls (u_1 and u_2) have been modeled as quadratic polynomials over each of the

eight shooting intervals used for the problem. The tolerances for DASPK3.0 were $rtol = 10^{-7}$ and $atol = 10^{-7}$, and all tolerances for SNOPT were 10^{-5}. The control and state continuity constraints across the shooting intervals have been satisfied to less than or equal to 0.1 when the optimizer has stopped. The time taken for solving this problem is 1421.91 seconds.

With the same parameter settings for the numerical methods, a pDAE formulation leaving (34f) with the optimizer as a path constraint and treating τ as an additional control variable has been attempted. The plot in Figure 6.1 is obtained from the results. The solution is incorrect. After six major iterations in SNOPT (162, 50, 72, 8, 1, 9, and 22 minor iterations) the merit function is reduced from $3.28975960E+21$ to $5.63316979E+12$. But in the process the error integral $\int_{t_0}^{t_{\text{final}}}(p_1^2+p_2^2+((x-d)^2+z^2-r^2)^2)^{0.5}dt$ is minimized to 2446227.48766, beyond which the optimizer fails to find descent directions. Clearly the stopping point is an unacceptable solution to the physical problem. Many of the continuity constraints in the multiple shooting method are unacceptably violated.

6.2. Analytical example. Consider the system

$$\dot{x}_1 = (2-t)\nu y + q_1(t), \tag{37a}$$

$$\dot{x}_2 = (\nu-1)y + q_2(t), \tag{37b}$$

$$0 = (t+2)x_1 + (t^2-4)x_2 + r(t) \tag{37c}$$

for $0 \le t \le 1$ and $\nu \ge 1$. This is an index 2 system since $\begin{pmatrix}(t+2) & (t^2-4)\end{pmatrix}\begin{pmatrix}(2-t)\nu \\ (\nu-1)\end{pmatrix}$ is nonzero for the given interval of t and ν. The idempotent projection $R(t,\nu) = \begin{pmatrix}1-\nu & \nu(2-t) \\ \frac{\nu-1}{t-2} & \nu\end{pmatrix}$. The EUODE (the homogeneous part only) is

$$\dot{\xi} = -\frac{\nu}{2-t}\xi.$$

Then

$$W = \begin{pmatrix}\frac{2-t}{(5-4t+t^2)^{0.5}} \\ \frac{1}{(5-4t+t^2)^{0.5}}\end{pmatrix}, \qquad \dot{W} = \begin{pmatrix}\frac{-1}{((5-4t+t^2))^{1.5}} \\ \frac{-(t-2)}{(5-4t+t^2)^{1.5}}\end{pmatrix},$$

and

$$\mu(-T(C_2T)^{-1}\dot{C}_2) = \frac{(1+0.5t)\nu - t + (0.25+t^2+(-0.5-2t^2)\nu+\nu^2(0.25+t^2)(5-4t+t^2))^{0.5}}{t^2-4}.$$

The term $\mu(H) = 0$ here, and y is the index 2 variable. In this case the stability criterion reduces to $\mu(W\dot{W}^T) + \mu(-T(C_2T)^{-1}\dot{C}_2) \le TOL$. Here $\mu(W\dot{W}^T)$ takes values between 0 and 0.25 for the same values of ν. The index 2 stability is determined by $\mu(-T(C_2T)^{-1}\dot{C}_2)$, which varies between 0 and 30 for $0 \le t \le 1$ and $1 \le \nu \le 1000$. The value of TOL was chosen as 20.0. In this example the practical tolerance from (30) is same as the original TOL value, since the time interval is unity and the logarithmic norm of the unconstrained dynamics is zero. For $\nu < 1$ the value of $Y(\nu,t) := \mu(-T(C_2T)^{-1}\dot{C}_2)$ goes up with decreasing ν and increasing t. For $\nu < 0$ the constraint is rejected. Table 6.1 illustrates the decision making of the partitioning algorithm. The plot in Figure 6.4 shows that the index 2 constraint is acceptable for

TABLE 6.1
Output from PARTITION algorithm, $t = 0$, $|k| = 1$, $H = 0$, $TOL = 20$.

Eqn.	ν	$\mu(-2T(C_2T)^{-1}\dot{C}_2)$	Decision
37	1	0.4828	constraint accepted
37	10	5.7903	constraint accepted
37	100	58.9052	constraint rejected

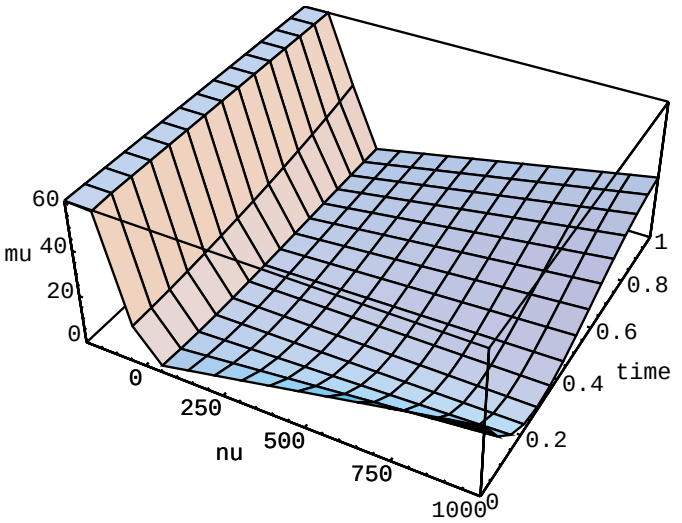


FIG. 6.4. *$\mu(-T(C_2T)^{-1}\dot{C}_2)$ for the analytical example.*

$1 \leq \nu \leq 30$ when the value of TOL is chosen as 20.0 and the value of TOL_0 in (30) is chosen as 0. But for $\nu \leq 0$ the constraint destabilizes the system, and for $\nu >> 0$ the constraint becomes unacceptable for inclusion in the DAE.

The logarithmic norm based stability estimate for the analytical example are overestimates from the eigenvalue point of view. But eigenvalues do not give the complete picture of stability for the numerical integration method [14]. This is especially true when the stability matrix is nonnormal or has a high norm compared to the maximum real part of its eigenvalues. In such cases, the maximum real part of the pseudoeigenvalues of the stability matrix often gives a more realistic estimate of stability with respect to a numerical integration method. From the definition of pseudoeigenvalue a relationship between the maximum real part of the pseudoeigenvalues and the logarithmic norm is obtained. The pseudoeigenvalue (λ_ϵ) of a matrix $A \in \mathcal{R}^{n\times n}$ is defined [14] as

$$\lambda_\epsilon(A) = \lambda(A + E), \quad \text{where} \quad \epsilon \geq ||\mathrm{E}||_2 \geq 0. \tag{38}$$

For stability it is required that $\max(\Re(\lambda_\epsilon)) \leq TOL_1$, where TOL_1 is a small positive number. In our experiments we set TOL_1 to 2.0. The number ϵ is usually small and is chosen as $10^{-6}, 10^{-3}, 10^{-1}, 1$ for plotting the contour lines (or a surface) on the complex plane. Using the triangle inequality property of the logarithmic norm, the bound $-||A||_2 \leq \mu(A) \leq ||A||_2$, and Lemma 2.4, we get

$$\max_{\lambda_\epsilon} \Re(\lambda_\epsilon(A)) = \max_{\lambda} \Re(\lambda(A + E)) \leq \mu(A + E) \leq \mu(A) + \mu(E) \leq \mu(A) + \epsilon. \tag{39}$$

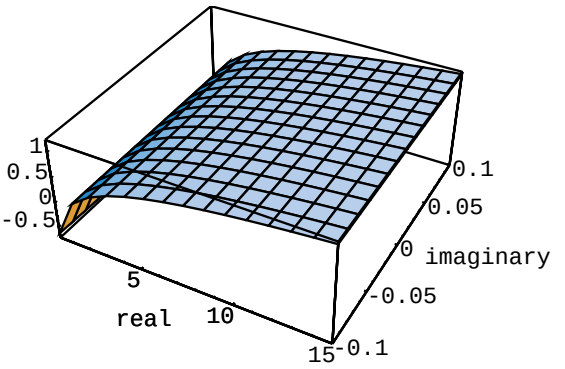


FIG. 6.5. $\lambda_\epsilon(-T(C_2T)^{-1}\dot{C}_2)$ *at* $t = 0$ *and* $\nu = 1000.0$ *for the analytical example* (37). *The vertical axis is* $\log_{10}\epsilon$.

In the analytical example, one would expect to obtain a highly stable EUODE (eigenvalue $= -1000$ for $\nu = 1000$ and $t = 1$) from the index 2 system for $\nu >> 1$. But the pseudoeigenvalues (Figure 6.5) indicate that numerical perturbation of the order of 0.1 can destabilize the system for $\nu >> 1$. The behavior of the system predicted by the logarithmic norm is confirmed by results from numerical simulations in [3].

6.3. Nonlinear oscillatory problems. The logarithmic norm can be very large for some nonlinear oscillatory systems, without the systems being unstable. Here we consider two such systems and discuss the implications for logarithmic norm-based stability criteria.

Stiff spring pendulum. Consider the ODE for a pendulum suspended at one end by a massless spring

$$\begin{aligned} \dot{p} - u &= 0, \\ \dot{q} - v &= 0, \\ \dot{u} + \frac{p((p^2+q^2)^{0.5} - 1)}{\epsilon^2(p^2+q^2)^{0.5}} &= 0, \\ \dot{v} + \frac{q((p^2+q^2)^{0.5} - 1)}{\epsilon^2(p^2+q^2)^{0.5}} - 1 &= 0, \end{aligned}$$

where p and q are the generalized position coordinates, u and v are the corresponding speeds, and $\epsilon \to 0$ is a nonzero small number. The system is an ODE for which the logarithmic norm (with $\epsilon = 10^{-3.0}$, $p = 0.6$, $q = 0.8$) is $O(\frac{1}{\epsilon^2})$ (499999.5), but the largest real part of the eigenvalues is zero, and the largest imaginary part of the eigenvalues is $O(\frac{1}{\epsilon})$ (1000). Thus the logarithmic norm captures the local (transient) high frequency oscillations.

The stiff spring pendulum is highly oscillatory. For a more complete description of this system, see [13]. For any numerical method to be able to follow the oscillations, it must take steps of size $O(\frac{1}{\epsilon})$. The large value of $O(\frac{1}{\epsilon^2})$ for the logarithmic norm gives an indication that a large number of time steps may be needed to accurately integrate this system over a time interval of length $O(1)$. Although a linear oscillatory system has a logarithmic norm of zero, this mechanical system has a high logarithmic norm due to the highly oscillatory transformation of variables which would be needed to

transform from Cartesian coordinates to a set of coordinates where the system would be nearly linear. In general, one might expect that for highly oscillatory mechanical systems of this type, the number of time steps which would be needed to accurately solve the problem will be proportional to the square root of the logarithmic norm times the length of the time interval for integration. Although this may be a large number of time steps, users who want to solve this problem will have to go ahead and do this work. Thus a large value of the logarithmic norm, especially as in this case for the ODE, is not necessarily an indicator that the problem should not be solved but rather a warning that this may require many time steps.

Wheelset on rails. This problem is a nonlinear highly oscillatory DAE and illustrates that with the practical stability criterion (30) our estimates make a reasonable decision regarding partitioning of the constraints. This DAE system was obtained from a standard test set at CWI, the Netherlands, at http://www.cwi.nl/cwi/projects/IVPtestset/descrip.htm. The system consists of equations describing oscillatory and rotational dynamics of two wheels joined by an axle through a spring-damper (damped suspension). The constraints describe the geometrical integrity of the system and the requirement that the wheels travel on a set of rails. The first 11 equations and variables 1 through 11 are differential equations and variables. The logarithmic norm for the differential system is computed as 8648. The time interval of integration is 20 seconds. The value of TOL was set to 20.0, and TOL_0 was set to 9000, since we expect the logarithmic norm to reflect the high frequency oscillations rather than the stability of the system. The output verifies the correct working of the present algorithm. The constraint equations 14 through 17 are index 1. The corresponding index 1 variables are 15, 14, 13, and 12. The estimated logarithmic norm increases cumulatively to 13166 (ODE + constraint 14), 13170 (ODE + constraints 14 and 15), 16647 (ODE + constraints 14, 15, 16), and then to 16649 (ODE + constraints 14, 15, 16 and 17) as the index 1 constraints are appended one after another. The remaining system (equations 12 and 13) includes index 2 equations and variables. Appending constraint 12 changes the estimated logarithmic norm (29) to 16483, and after appending 13, the logarithmic norm of the complete (ODE + constraints 12–17) system (using (29)) is estimated as 16403. The parameter $|k|$ was computed as 0.998. The CPU time taken to build the complete system was 0.1343s. All the constraints were accepted in the pDAE (ODE + constraints 12–17). During check back, the eigenvalue of the UODE for the pDAE system (ODE + constraints 12–17) was computed as 5.46020. Considering the relatively small interval of simulation, we decided to integrate the pDAE with all the constraints.

The wheel-axle problem was integrated using DASPK3.0 from 0 to 10 seconds with all the relative and absolute tolerances set to 0.0001. The integration was successfully completed in 5951 steps of which 5094 steps were accepted. For the tolerances set to 0.000001, the integration process took 15893 steps (of which 14204 steps were accepted) to complete. This confirms our rough estimates.

REFERENCES

[1] U. M. Ascher and L. R. Petzold, *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations*, SIAM, Philadelphia, 1998.

[2] U. M. Ascher, R. M. M. Mattheij, and R. D. Russell, *Numerical Solution of Boundary Value Problems for Ordinary Differential Equations*, SIAM, Philadelphia, 1995.

[3] U. M. Ascher and L. R. Petzold, *Stability of computational methods for constrained dynamics systems*, SIAM J. Sci. Comput., 14 (1993), pp. 95–120.

[4] J. T. Betts, *Exploiting sparsity in the direct transcription method for optimal control point to point path optimization*, in Proceedings of the International Symposium on Mathematical Programming, EPFL, Lausanne, Switzerland, 1997.

[5] K. E. Brenan, S. L. Campbell, and L. R. Petzold, *Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations*, 2nd ed., SIAM, Philadelphia, 1996.

[6] S. L. Cambell, *High index differential algebraic equations*, Mech. Structures Mach., 23 (1995), pp. 199–222.

[7] A. Cervantes and L. T. Biegler, *Large-scale DAE optimization using simultaneous nonlinear programming formulations*, AIChE Journal, 44 (1998), p. 1038.

[8] J.-L. Chern and L. Dieci, *Smoothness and periodicity of some matrix decompositions*, SIAM J. Matrix Anal. Appl., 22 (2000), pp. 772–792.

[9] I. S. Duff and C. W. Gear, *Computing the structural index*, SIAM J. Algebraic Discrete Methods, 7 (1986), pp. 594–603.

[10] W. F. Feehery, J. R. Banga, and P. I. Barton, *A novel approach to dynamic optimization of ODE and DAE systems as high index problems*, in Proceedings of the AIChE Annual Meeting, Miami Beach, FL, 1995.

[11] E. Hairer, S. P. Nørsett, and G. Wanner, *Solving Ordinary Differential Equations:* Vol. 1, 2nd ed., Springer-Verlag, Berlin, 1993.

[12] E. Hairer and G. Wanner, *Solving Ordinary Differential Equations: Differential-Algebraic and Stiff Systems: Vol.* 2, Springer-Verlag, Berlin, 1991.

[13] E. Hairer, Ch. Lubich, and M. Roche, *The Numerical Solution of Differential-Algebraic Systems by Runge-Kutta Methods*, Lecture Notes in Math. 1409, Springer-Verlag, Berlin, 1989.

[14] D. J. Higham and L. N. Trefethen, *Stiffness of ODEs*, BIT, 33 (1993), pp. 285–303.

[15] S. Li and L. Petzold, *Software and algorithms for sensitivity analysis of large-scale differential-algebraic systems*, J. Comput. Appl. Math., 125 (2000), pp. 131–145.

[16] L. Petzold, P. Gill, L. O. Jay, M. Leonard, and V. Sharma, *An SQP method for the optimal control of large-scale dynamical systems*, J. Comput. Appl. Math., 20 (2000), pp. 197–213.

[17] S. Raha and L. R. Petzold, *Constraint partitioning for structure in path-constrained dynamic optimization problems*, Appl. Numer. Math., submitted.

[18] V. H. Schulz, H. G. Bock, and M. C. Steinbach, *Exploiting invariants in the numerical solution of multipoint boundary value problems for DAEs*, SIAM J. Sci. Comput., 19 (1998), pp. 440–467.

[19] V. S. Vassiliadis, R. W. H. Sargent, and C. Pantelides, *Solution of a class of multi-stage dynamic optimization problems.* 2. *Problems with path constraints*, IEC Research, 33 (1994), pp. 2123–2133.

SIAM J. SCI. COMPUT.
Vol. 22, No. 6, pp. 2075–2092
© 2001 Society for Industrial and Applied Mathematics

TWO ELEMENT-BY-ELEMENT ITERATIVE SOLUTIONS FOR SHALLOW WATER EQUATIONS*

C. C. FANG† AND TONY W. H. SHEU†

Abstract. In this paper we apply the generalized Taylor–Galerkin finite element model to simulate bore wave propagation in a domain of two dimensions. For stability and accuracy reasons, we generalize the model through the introduction of four free parameters. One set of parameters is rigorously determined to obtain the high-order finite element solution. The other set of free parameters is determined from the underlying discrete maximum principle to obtain the monotonic solutions. The resulting two models are used in combination through the flux correct transport technique of Zalesak, thereby constructing a finite element model which has the ability to capture hydraulic discontinuities. In addition, this paper highlights the implementation of two Krylov subspace iterative solvers, namely, the bi-conjugate gradient stabilized (Bi-CGSTAB) and the generalized minimum residual (GMRES) methods. For the sake of comparison, the multifrontal direct solver is also considered. The performance characteristics of the investigated solvers are assessed using results of a standard test widely used as a benchmark in hydraulic modeling. Based on numerical results, it is shown that the present finite element method can render the technique suitable for solving shallow water equations with sharply varying solution profiles. Also, the GMRES solver is shown to have a much better convergence rate than the Bi-CGSTAB solver, thereby saving much computing time compared to the multifrontal solver.

Key words. Taylor–Galerkin finite element model, discrete maximum principle, flux correct transport technique, Bi-CGSTAB, GMRES, multifrontal direct solver, sharply varying

AMS subject classifications. 65F10, 65N30, 76B15

PII. S1064827599360881

1. Introduction.

Many environmental problems, such as tides in oceans, breaking waves on shallow beaches, flood waves in rivers, mountain torrents, and estuary flows [1] are closely related to the motion of unsteady free-surface flow. Predicting the height and speed of the bore wave is the first step in providing useful information for flood control and for the design of channel walls. It is the practical importance of simulating shallow water equations that motivated the present study.

The shallow water height is analogous to gas density in gas dynamic equations. Since gas dynamic equations admit discontinuous solutions, called shocks and contact discontinuities, this analogy between two fields of equations implies that it is possible to observe hydraulic jumps and bores in water and in the atmosphere. Numerically capturing these discontinuous phenomena in hydraulics has become a major area of theoretical and computational study. For suppressing dispersive oscillations exhibited near the shock front, significant effort has been directed toward the development of high-resolution hydraulic methods. Shock-capturing methods were first developed by Godunov [2] and Van Leer [3]. Development of high-resolution schemes was followed by adoption of the total variation diminishing (TVD) scheme of Harten [4], the parabolic method (PPM) of Colella and Woodward [5], and the essentially nonoscillatory (ENO) schemes of Harten and Osher [6] to obtain numerically very accurate but computationally absolute stable solutions. Most of these high-resolution schemes

*Received by the editors September 2, 1999; accepted for publication (in revised form) August 23, 2000; published electronically March 28, 2001. This work was supported by National Science Council of Republic of China grant NSC 88-2611-E-002-025.

http://www.siam.org/journals/sisc/22-6/36088.html

†Department of Naval Architecture and Ocean Engineering, National Taiwan University, 73 Chou-Shan Rd., Taipei, Taiwan, Republic of China (Tony.Sheu@cfd.na.ntu.edu.tw).

were, unfortunately, developed within the one-dimensional framework. The desire to avoid this limitation has prompted many researchers to capture bore waves in an open-channel flow and hydraulic jumps in a genuinely multidimensional dam-break problem [7, 8, 9, 10, 11, 12, 13, 14, 15]. Since the flux corrected transport (FCT) algorithm was originally developed without resorting to a single spatial dimension [16, 17], we consider the FCT algorithm to be one of the most suitable choices for multidimensional hydraulic calculations.

As is typical with other finite element flow simulations, we work with a large matrix equation. Thus, we must minimize this disadvantage if we are to compete with other structured-type discretization methods. To this end, we consider in the present work two iterative solvers and one very effective direct solver. Both iterative solvers, namely, the bi-conjugate gradient stabilized (Bi-CGSTAB) of Van der Vorst [18] and the generalized minimum residual (GMRES) of Saad and Schultz [19], are the ensemble of conjugate gradient method for solving the non-Hermitian linear system of algebraic equations. These two Krylov subspace methods differ in the vectors used to iterate the approximation solution. GMRES iterates the approximate solution in terms of Arnoldi vectors while Bi-CGSTAB accomplishes the same task through the use of unsymmetric Lanczos vectors. These vectors are chosen to overcome the difficulty regarding the nonexistence of the orthogonal tridiagonalization of the finite element stiffness matrix for shallow water equations. Since we wish to address the performance of solution solvers, we also consider the direct solver for completeness. In order to compete with the above state-of-the-art iterative solvers, we consider in this assessment study the multifrontal direct solver of Duff and Reid [20]. This solver is regarded as a refinement of the frontal solver of Irons [21].

The remainder of this paper is organized in six sections. Section 2 presents assumptions that lead to the St. Venant shallow water equations. In section 3, we present the Taylor–Galerkin finite element model, which can faithfully preserve the inherent hyperbolic conservation law [22]. Specific to the present finite element model is that four free parameters and one damping parameter are used to obtain higher prediction accuracy in the high-order scheme while accommodating the monotonicity property in the low-order scheme [23]. Section 3.3 explains how the high-order and low-order Taylor–Galerkin formulations can be used in combination to obtain a nonoscillatory solution. This is followed by presentation of two iterative solution solvers and one direct solver that can be used to solve the nonsymmetric finite element equations. The objective is to assess the efficiency of the solution solvers used to obtain finite element solutions from indefinite and unsymmetric matrix equations. In section 5, we provide analytical evidence by showing that the numerical model and solution solvers can render techniques suitable for hydraulic problems with sharp gradients. All results of model tests are well in agreement with the analytic data. With this success in analytical validation, we proceed to study the dam-break problem. Finally, a brief discussion together with some conclusions is presented in section 6.

2. Mathematical formulation. Shallow water equations are derived under zero fluid viscosity and surface tension assumptions. Wind shear and Coriolis forces are not taken into account. Working equations for incompressible free-surface fluid flows are derived under the small bottom slope condition. Another key assumption in the derivation is that the vertical component of the flow acceleration has negligible influence on the pressure. A hydrostatic pressure distribution is thus assumed. Given the above assumptions, the St. Venant shallow water equations, which govern mass and momentum conservation, are derived in terms of a solution vector $\mathbf{U}$ of the

conservative type [24]

$$\mathbf{U}_t + \mathbf{F}_x + \mathbf{G}_y = \mathbf{S}, \tag{2.1}$$

where $\mathbf{U} = (h, hu, hv)^{\mathrm{T}}$. In the above, h is the water depth, and u and v are the depth-averaged velocity components in the x- and y-directions, respectively. Denote g as the gravitational acceleration; the physical fluxes are derived as $\mathbf{F} = (h\,u, h\,u^2 + \frac{1}{2}g\,h^2, h\,u\,v)^{\mathrm{T}}$ and $\mathbf{G} = (h\,v, h\,u\,v, h\,v^2 + \frac{1}{2}g\,h^2)^{\mathrm{T}}$. Without loss of generality, both friction losses and bed slopes are neglected to simplify the analysis.

Given initially smooth data for the present homogeneous case ($\mathbf{S} = \mathbf{0}$), the quasi-linear hyperbolic system of partial differential equations may admit discontinuities, such as bore waves that are often observed in practice, owing to nonlinear advective terms in the equations [25]. In time-accurate simulation of St. Venant equations, it is customary to rewrite working equations in their nonconservative equivalent forms to better show the characteristic nature of the hyperbolic system. Transformation of the conservative form into its nonconservative counterpart involves using the gravity wave velocity $c = (g\,h)^{1/2}$. The resulting eigenvectors and eigenvalues are critical in hydraulic simulation since they represent the characteristic speed and direction of signal transmission.

3. Finite element model. Within the weighted residual context, (2.1) can be approximated in its weak form through the use of a test function $\mathbf{W}$. This leads to

$$\sum_{el=1}^{n_{el}} \int_{\Omega^{el}} \int_{t_n}^{t_{n+1}} \mathbf{W} \left[\frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{F}}{\partial x} + \frac{\partial \mathbf{G}}{\partial y} \right] d\,t\,d\,\Omega^{el} = 0. \tag{3.1}$$

Define $\delta\mathbf{U}^n = \mathbf{U}^{n+1} - \mathbf{U}^n$ and $\mathbf{I} = \frac{\partial}{\partial x}\int_{t_n}^{t_{n+1}} \mathbf{F}\,dt + \frac{\partial}{\partial y}\int_{t_n}^{t_{n+1}} \mathbf{G}\,dt$; (3.1) can be expressed as follows through time integration:

$$\sum_{el=1}^{n_{el}} \int_{\Omega^{el}} (\mathbf{W}\delta\mathbf{U}^n - \mathbf{W}\,\mathbf{I})\,d\Omega^{el} = 0. \tag{3.2}$$

Analysis is carried out by performing Taylor series expansion of $\mathbf{F}$ and $\mathbf{G}$ with respect to t_n. Take $\mathbf{F}$ as an example; we can represent this vector in terms of Taylor series expansion terms terminated at the time increment $(t - t_n)^3$:

$$\mathbf{F} = \mathbf{F}^n + \left.\frac{\partial \mathbf{F}}{\partial t}\right|^n (t - t_n) + \frac{1}{2}\left.\frac{\partial^2 \mathbf{F}}{\partial t^2}\right|^n (t - t_n)^2 + \mathcal{O}(t - t_n)^3. \tag{3.3}$$

Recall that $\frac{\partial \mathbf{U}}{\partial t} = -\mathbf{F}_x - \mathbf{G}_y$ and $\frac{\partial \mathbf{F}}{\partial t} = \frac{\partial \mathbf{F}}{\partial \mathbf{U}}\frac{\partial \mathbf{U}}{\partial t} = \mathbf{A}\frac{\partial \mathbf{U}}{\partial t}$; we introduce two free parameters α and β and rewrite $\partial\mathbf{F}/\partial t$ exactly as $\frac{\partial \mathbf{F}}{\partial t} = \alpha\mathbf{A}\frac{\partial \mathbf{U}}{\partial t} + \beta\mathbf{A}[-\frac{\partial \mathbf{F}}{\partial x} - \frac{\partial \mathbf{G}}{\partial y}]$, provided that α and β are constrained by $\alpha + \beta = 1$. Moreover, we can approximate the time derivative term $\partial\mathbf{F}/\partial t$ to obtain

$$\begin{aligned} \frac{\partial^2 \mathbf{F}}{\partial t^2} = &-\gamma \left(\mathbf{A}^2 \frac{\partial^2 \mathbf{U}}{\partial t \partial x} + \mathbf{A}\,\mathbf{B} \frac{\partial^2 \mathbf{U}}{\partial t \partial y} \right)\Bigg|^n \\ &+ \mu \left[\mathbf{A}^2 \left(\frac{\partial^2 \mathbf{F}}{\partial x^2} + \frac{\partial^2 \mathbf{G}}{\partial x \partial y} \right) + \mathbf{A}\,\mathbf{B} \left(\frac{\partial^2 \mathbf{F}}{\partial x \partial y} + \frac{\partial^2 \mathbf{G}}{\partial y^2} \right) \right]\Bigg|^n . \end{aligned} \tag{3.4}$$

As in the above, the free parameters γ and μ, which are constrained by $\gamma + \mu = 1$, are also introduced. Substitution of $\frac{\partial \mathbf{F}}{\partial t}$ and $\frac{\partial^2 \mathbf{F}}{\partial t^2}$ into (3.3) leads to

$$\begin{aligned}\mathbf{F} = \mathbf{F}^n &+ \left[\alpha \mathbf{A} \frac{\partial \mathbf{U}}{\partial t} - \beta \mathbf{A} \left(\frac{\partial \mathbf{F}}{\partial x} + \frac{\partial \mathbf{G}}{\partial y} \right) \right] \Bigg|^n (t - t_n) \\ &- \frac{1}{2} \left\{ \gamma \left(\mathbf{A}^2 \frac{\partial^2 \mathbf{U}}{\partial t \partial x} + \mathbf{AB} \frac{\partial^2 \mathbf{U}}{\partial t \partial y} \right) - \mu \left[\mathbf{A}^2 \left(\frac{\partial^2 \mathbf{F}}{\partial x^2} + \frac{\partial^2 \mathbf{G}}{\partial x \partial y} \right) \right. \right. \\ &\left. \left. + \mathbf{AB} \left(\frac{\partial^2 \mathbf{F}}{\partial x \partial y} + \frac{\partial^2 \mathbf{G}}{\partial y^2} \right) \right] \right\} \Bigg|^n (t - t_n)^2 + \mathcal{O}((t - t_n)^3). \end{aligned} \tag{3.5}$$

Similarly, we can expand $\mathbf{G}$ with respect to quantities evaluated at time t_n:

$$\begin{aligned}\mathbf{G} = \mathbf{G}^n &+ \left[\alpha \mathbf{B} \frac{\partial \mathbf{U}}{\partial t} - \beta \mathbf{B} \left(\frac{\partial \mathbf{F}}{\partial x} + \frac{\partial \mathbf{G}}{\partial y} \right) \right] \Bigg|^n (t - t_n) \\ &- \frac{1}{2} \left\{ \gamma \left(\mathbf{BA} \frac{\partial^2 \mathbf{U}}{\partial t \partial x} + \mathbf{B}^2 \frac{\partial^2 \mathbf{U}}{\partial t \partial y} \right) - \mu \left[\mathbf{BA} \left(\frac{\partial^2 \mathbf{F}}{\partial x^2} + \frac{\partial^2 \mathbf{G}}{\partial x \partial y} \right) \right. \right. \\ &\left. \left. + \mathbf{B}^2 \left(\frac{\partial^2 \mathbf{F}}{\partial x \partial y} + \frac{\partial^2 \mathbf{G}}{\partial y^2} \right) \right] \right\} \Bigg|^n (t - t_n)^2 + \mathcal{O}((t - t_n)^3). \end{aligned} \tag{3.6}$$

By substituting (3.5)–(3.6) into (3.2) and choosing bilinear polynomials as test and basis functions, we can derive the finite element equation in the δ-form as follows:

$$\mathbf{M}_c \, \delta \mathbf{U}^n = \mathbf{R}. \tag{3.7}$$

In the above, $\mathbf{M}_c$ denotes the consistent mass matrix:

$$\begin{aligned}\mathbf{M}_{ij}^{el} = &\int_{\mathbf{\Omega}^{el}} \left\{ \mathbf{N}_i \mathbf{N}_j - \frac{1}{2} \alpha \Delta t \left(\frac{\partial \mathbf{N}_i}{\partial x} A + \frac{\partial \mathbf{N}_i}{\partial y} B \right) \mathbf{N}_j \right. \\ &+ \frac{1}{6} \gamma \Delta t^2 \left[\frac{\partial \mathbf{N}_i}{\partial x} \left(A^2 \frac{\partial \mathbf{N}_j}{\partial x} + AB \frac{\partial \mathbf{N}_j}{\partial y} \right) + \frac{\partial \mathbf{N}_i}{\partial y} \left(BA \frac{\partial \mathbf{N}_j}{\partial x} + B^2 \frac{\partial \mathbf{N}_j}{\partial y} \right) \right] \Bigg\} d\mathbf{\Omega}^{el} \\ &- \int_{\mathbf{\Gamma}} \left\{ -\frac{1}{2} \alpha \Delta t \mathbf{N}_i \left(n_x A + n_y B \right) \mathbf{N}_j + \frac{1}{6} \gamma \Delta t^2 \mathbf{N}_i \left[n_x \left(A^2 \frac{\partial \mathbf{N}_j}{\partial x} + AB \frac{\partial \mathbf{N}_j}{\partial y} \right) \right. \right. \\ &\left. \left. + n_y \left(BA \frac{\partial \mathbf{N}_j}{\partial x} + B^2 \frac{\partial \mathbf{N}_j}{\partial y} \right) \right] \right\} d\mathbf{\Gamma}. \end{aligned} \tag{3.8}$$

In (3.7), $\delta \mathbf{U}^n (\equiv \mathbf{U}^{n+1} - \mathbf{U}^n)$ is the vector of nodal increment and $\mathbf{R}$ $(\equiv \mathbf{C}\mathbf{F}^n + \tilde{\mathbf{C}}\mathbf{G}^n)$ is the vector of element contributions added to the finite element nodes. For detailed expressions of $\mathbf{C}$ and $\tilde{\mathbf{C}}$, refer to [23].

3.1. High-order Taylor–Galerkin finite element model. To resolve discontinuous solutions, we employ the FCT technique of Zalesak [17]. The idea behind this algorithm is to combine an accurate high-order scheme with a monotonic low-order scheme. To make this scheme effective, we require that the former scheme be used in the smooth regime and the low-order scheme only in regions near discontinuities.

The key to constructing an efficient FCT finite element model is to develop a model that can provide a high level of accuracy. To this end, we exploit the modified equation analysis for the determination of the free parameters α, β, γ, and μ a priori to obtain higher-order accuracy. The strategy we adopt to achieve this goal is to take into consideration the scalar transport equation, $\phi_t + a\,\phi_x + b\,\phi_y = 0$, in the flow

with the constant velocity vector $\vec{u} = (a, b)$. The modified equation analysis reveals the rational use of $(\alpha^h, \beta^h, \gamma^h, \mu^h) = (0, 1, 1, 0)$ [23]. With these parameters, the resulting modified equation reads as

$$\begin{aligned} &\phi_t + a\,\phi_x + b\,\phi_y \\ &= T_1\,\phi_{xxxx} + T_2\,\phi_{xxxy} + T_3\,\phi_{xxyy} + T_4\,\phi_{xyyy} + T_5\,\phi_{yyyy} + \cdots, \end{aligned} \tag{3.9}$$

where $T_1 = \frac{1}{24}\,a\,\Delta x^3\,\nu_x\,({\nu_x}^2 - 1)$, $T_2 = \frac{1}{6}\,b\,\Delta x^3\,{\nu_x}^3$, $T_3 = -\frac{1}{12}\,b^2\,\Delta x^2\,\Delta t\,{\nu_x}^2$, $T_4 = \frac{1}{6}\,a\,\Delta y^3\,{\nu_y}^3$, $T_5 = \frac{1}{24}\,b\,\Delta y^3\,\nu_y\,({\nu_y}^2 - 1)$. In light of the spatial third-order accuracy, and the first-order temporal accuracy, the above Taylor–Galerkin finite element model shows promise as a means of predicting a smoothly distributed water height in shallow water equations.

3.2. Low-order Taylor–Galerkin finite element model. The next step in the development of the Taylor–Galerkin FCT (TG-FCT) finite element model is to derive the low-order model from the generalized Taylor–Galerkin finite element model. To achieve this goal, the model is not allowed to produce any nonphysical or numerical wiggles. This monotonicity and strictly positive field variable requirement is a key to success in any FCT method. The better the low-order scheme, the easier the task of limiting fluxes.

The development of a low-order finite element model proceeds as follows. We first rewrite (3.7) as

$$\mathbf{M}_c\,\mathbf{U}^{n+1} = \mathbf{R}^n + \mathbf{M}_c\mathbf{U}^n. \tag{3.10}$$

The derivation is followed by lumping the above equation to get

$$\mathbf{M}_l\,\mathbf{U}^{n+1} = \mathbf{R}^n + \mathbf{M}_c\mathbf{U}^n. \tag{3.11}$$

The above lumping-mass approximation helps to stabilize the discretized equation. Subtracting $\mathbf{M}_l\,\mathbf{U}^n$ from both sides of (3.11), we obtain

$$\mathbf{M}_l\,\delta\mathbf{U}^n = \mathbf{R}^n + (\mathbf{M}_c - \mathbf{M}_l)\,\mathbf{U}^n. \tag{3.12}$$

Further refinement of (3.12) can be made by multiplying c_d $(0 \le c_d \le 1)$ by the added mass diffusion term to better control the predicted solution. This helps us avoid the introduction of unnecessarily large diffusion errors. The resulting model for obtaining the lower-order Taylor–Galerkin finite element solution $\mathbf{U}^n$ reads as

$$\mathbf{M}_l\,\delta\mathbf{U}^n = \mathbf{R}^n + c_d\,(\mathbf{M}_c - \mathbf{M}_l)\,\mathbf{U}^n, \tag{3.13}$$

where $\mathbf{M}_l = \mathcal{A}_{el=1}^{n_{el}}(\mathrm{diag}(\sum_{j=1}^{n_{\max}} \mathbf{M}_{cij}^{el}))$. In order to satisfy the requirement placed on the low-order scheme in any FCT method, we employ the discrete maximum theory [26, 27, 28]. Based on this underlying theory, the matrices involved in (3.13) are of the M-matrix type provided that the five free parameters introduced into the scalar formulation are prescribed as $(\alpha^l, \beta^l, \gamma^l, \mu^l, c_d) = (0, 0, 0, 0, 0.425)$.

When simulating nonlinear shallow water equations, we may encounter a sonic flow situation. In this case, entropy fix must be invoked in order to avoid nonphysical rarefaction shocks at the sonic point. To satisfy the entropy satisfaction property when the sonic condition is detected, we should add an entropy flux term, $\frac{\partial}{\partial x}(b(\nu_x)\frac{\partial \mathbf{U}}{\partial x}) + \frac{\partial}{\partial y}(b(\nu_y)\frac{\partial \mathbf{U}}{\partial y})$, to the region where it is needed. The damping coefficients used in the entropy flux are as follows [29]:

$$b(\nu_i) = c_e\,\frac{\Delta t}{2\lambda^2}\,q(\nu_i), \tag{3.14}$$

where $\lambda = \Delta t/\Delta x$ (or $\Delta t/\Delta y$), $\nu_i = \frac{\Delta t}{\Delta x}(u-c)$ (or $\nu_i = \frac{\Delta t}{\Delta y}(v-c)$), and

$$q(\nu_i) = \begin{cases} 0, & |\nu_i| \geq \epsilon, \\ \epsilon^2 - \nu_i^2, & |\nu_i| < \epsilon. \end{cases} \tag{3.15}$$

In what follows, we set $\epsilon = 0.2$ and $c_e = 2.0$.

3.3. FCT filtering algorithm. Having determined the values $(\alpha, \beta, \gamma, \mu, c_d)$ needed to obtain high- and low-order finite element solutions, we can use two Taylor–Galerkin models in combination to obtain positive and accurate results which are free of nonphysical fluctuations. Use of the two schemes in combination was proposed by Zalesak [17]. We follow closely the FCT scheme of Zalesak by calculating $\delta\mathbf{U}^h$ from the high-order model using either the iterative solvers or the multifrontal direct solver [30] discussed below.

Upon obtaining the solution $\delta\mathbf{U}^h$, we can compute the antidiffusive flux array $\mathbf{F}^{el^h}$ in each element:

$$\mathbf{F}^{el^h} = [F_i^{el^h}] = \mathbf{M}_{l_h}^{-1}\,[\mathbf{R}^{el^h} - (\mathbf{M}_c{}^{el^h} - \mathbf{M}_l{}^{el^h})\,\delta\mathbf{U}^h]. \tag{3.16}$$

The calculation is followed by computing the antidiffusive flux array $\mathbf{F}^{el^l}$ from the low-order Taylor–Galerkin solution $\delta\mathbf{U}^l$. The resulting antidiffusive flux array, $\mathbf{F}^{el^l}$, in each element is computed according to $\mathbf{F}^{el^l} = [\mathbf{M}_l{}^{-1}\mathbf{R}^{el^l}]$. When antidiffusive fluxes $\mathbf{F}^{el^h}$ and $\mathbf{F}^{el^l}$ become available, we can calculate the corrected antidiffusive flux array $\mathbf{F}^{el^c}$ [17]. The filtering processes finish with the calculation of $\mathbf{U}^{n+1}$ by means of $\mathbf{U}^{n+1} = \mathbf{U}^l + \mathcal{A}_{el=1}^{n_{el}}(\mathbf{F}^{el^c})$.

4. Solution solvers. The iterative solution solver is a strong rival to its direct counterpart because it is less prone to fill-in problems. However, though the storage problem can be considerably resolved, iterative methods have shortcomings of their own. Chief among these shortcomings is the poor control of convergence behavior. Due to space limitations, we will confine our review to iterative solvers based on the minimization concept. The conjugate gradient method of Hestenes and Stiefel [31], considered to be the pioneering work of this class of solvers, works effectively only for a matrix equation having clustered eigenvalues and suffers from pivoting breakdown when matrix symmetry is lost. Refinement of this Krylov subspace method in order to overcome the matrix asymmetry difficulty has been the primary focus of research during the last two decades.

In the literature, nonstationary iterative methods, which have the ability to resolve matrix asymmetry, are frequently referred to [32]. The Chebyshev methods are applicable only to positive definite equations [33]. Also, use of this class of methods requires knowledge of the eigenvalue spectrum a priori. To circumvent deficiencies in irregular convergence behavior and the indispensable transpose operation of the coefficient matrix inherent in the bi-conjugate gradient (Bi-CG) method [34], the Arnoldi or Lanczos algorithms were proposed. Like the Arnoldi algorithm, the GMRES method [19] iterates the approximation solution through use of a self-orthogonal sequence. Due to the prohibitive storage demand, the residual can be minimized optimally by adding a restart capability. In the iteration, no more than n steps are needed for an n by n matrix to reach convergence.

Within the Lanczos framework, product methods, such as conjugate gradient squares (CGS) [35], quasi-minimal residual (QMR) [36], and Bi-CGSTAB [18], are

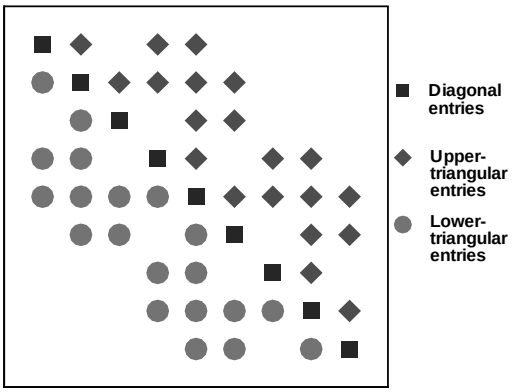


FIG. 4.1. *An illustration of the sparse matrix assembled from* 2×2 *elements.*

preferable for tackling equation asymmetry. The exploration of a set of dual orthogonal vectors is the building block of this class of methods. The QMR method of Freund and Nachtigal [36] was designed to avoid irregular convergence behavior. This method, unfortunately, suffers from the need to transpose the stiffness matrix. CGS, on the other hand, avoids the need for matrix transpose but inhibits irregular convergence behavior because it accommodates the same contraction polynomial as does Bi-CG. Besides the transpose-free version of QMR [37], the Bi-CGSTAB method of Van der Vorst [18] is a rational alternative. Bi-CGSTAB iterates the approximation solution in terms of unsymmetric Lanczos vectors, in conjunction with the local minimization method GMRES(1). Through manipulation of equal-order contraction polynomials of different kinds, one can dispense with transpose matrix procedures and suppress irregular convergence behavior. Nevertheless, much work still needs to be done so that pivoting breakdown and Lanczos breakdown can be avoided.

The choice of an appropriate solver for obtaining finite element solutions depends on the type of matrix equations employed. Take matrix equations constructed from 2×2 elements as an example; the sparse matrix, as shown in Figure 4.1, is found to be unsymmetric. Further eigenvalue analysis reveals that the matrix equations are indefinite, thus limiting application of conventional iterative solvers. In this study, we will consider two state-of-the-art iterative solvers and attempt to make a definite assessment of the multifrontal direct solver [30]. Two iterative solvers are the GMRES solver of Saad and Schultz [19] and the Bi-CGSTAB iterative solver of Van der Vorst [18]. These iterative solvers are variants of conjugate gradient methods.

4.1. The GMRES iterative solver. We will consider first the GMRES method of Saad and Schultz [19]. This nonstationary iterative solution solver is considered to be an extension of minimal residual (MINRES) which can be used to solve unsymmetric matrix equations. A sequence of tridiagonal matrices is used to obtain a progressively improved distribution of the eigenvalues of the original non-Hermitian linear stiffness matrix system. GMRES iterates the solution with the aid of Arnoldi vectors to overcome the difficulty of the nonsymmetry of the matrix equations. The employed Arnoldi algorithm involves partial tridiagonalization of the original matrix equations using one set of orthogonal vectors $\underline{\mathbf{Q}}$ to yield $\underline{\mathbf{Q}}^T \underline{\underline{\mathbf{A}}}\, \underline{\mathbf{Q}} = \underline{\mathbf{H}}$, where $\underline{\mathbf{H}}$ is

the Hessenberg reduction. The column-by-column generation of $\underline{\mathbf{Q}}$ has the property of $\underline{\mathbf{Q}}^T\,\underline{\mathbf{Q}} = \underline{\mathbf{I}}$ (i.e., identity matrix).

GMRES follows the modified Gram–Schmidt orthogonalization procedure. It invokes a restart capability to control the storage requirement. In GMRES, the main steps are as follows:

Set $\underline{\mathbf{x}}_0$ as an initial guess
For $j = 1, 2, \ldots$
  Solve $\underline{\mathbf{r}}$ from $\underline{\mathbf{r}} = \underline{\mathbf{b}} - \underline{\underline{\mathbf{A}}}\,\underline{\mathbf{x}}_0$ ⟵ element-by-element procedure
  $\underline{\mathbf{v}}_1 = \underline{\mathbf{r}}/\|\underline{\mathbf{r}}\|_2$
  $s := \|\underline{\mathbf{r}}\|_2$
  for $i = 1, 2, 3, \ldots, m$
    Solve $\underline{\mathbf{w}}$ from $\underline{\mathbf{w}} = \underline{\underline{\mathbf{A}}}\,\underline{\mathbf{v}}_i$ ⟵ element-by-element procedure
    for $k = 1, \ldots, i$
      $h_{k,i} = (\underline{\mathbf{w}}, \underline{\mathbf{v}}_k)$
      $\underline{\mathbf{w}} = \underline{\mathbf{w}} - h_{k,i}\,\underline{\mathbf{v}}_k$
    end
    $h_{i+1,i} = \|\underline{\mathbf{w}}\|_2$
    $\underline{\mathbf{v}}_{i+1} = \underline{\mathbf{w}}/h_{i+1,i}$
    apply $J_1, \ldots, J_{i-1}$ on $(h_{1,i}), \ldots, h(i+1, i))$
    construct J_i, acting on the ith and $(i+1)$st components of $h_{\cdot,i}$,
      such that the $(i+1)$st component of $J_i\,h_{\cdot,i}$ is with the value of 0
    $s := J_i\,s$
    if $s(i+1)$ is small enough, then (**UPDATE** $(\tilde{\underline{\mathbf{x}}}, i)$ and quit)
  end
  UPDATE $(\tilde{\underline{\mathbf{x}}}, m)$
End

The **UPDATE** $(\tilde{\underline{\mathbf{x}}}, i)$ procedure is as follows:

Compute $\underline{\mathbf{y}}$ from $\underline{\underline{\mathbf{H}}}\,\underline{\mathbf{y}} = \underline{\mathbf{s}}$,
  in which the upper $i \times i$ triangular part of $\underline{\underline{\mathbf{H}}}$ has $h_{i,j}$ as its elements,
  $\underline{\mathbf{s}}$ is the first i components of s
$\tilde{\underline{\mathbf{x}}} = \underline{\mathbf{x}}_0 + \underline{\mathbf{y}}_1\underline{\mathbf{v}}_1 + \underline{\mathbf{y}}_2\underline{\mathbf{v}}_2 + \cdots + \underline{\mathbf{y}}_i\underline{\mathbf{v}}_i$
$s_{i+1} = \|\underline{\mathbf{b}} - \underline{\underline{\mathbf{A}}}\,\tilde{\underline{\mathbf{x}}}\|_2$ ⟵ element-by-element procedure
if $\tilde{\underline{\mathbf{x}}}$ is accurate enough, then quit
else $\underline{\mathbf{x}}_0 = \tilde{\underline{\mathbf{x}}}$.

In the above, the inner product coefficients $\|w^i\|$ and (w^i, v^k) are stored in a Hessenberg matrix. Upon obtaining the values of y_k, which are designed to minimize the residual norm $\|\underline{\mathbf{b}} - \underline{\underline{\mathbf{A}}}\,\underline{\mathbf{x}}^{(j)}\|$, the GMRES iterations are constructed as $x^j = x^0 + \sum_{i=1} y_i v^i$. GMRES will converge in no more than n iterations when an n by n matrix equation is solved. This is practically infeasible since the storage and computational requirements are prohibitive if n is large. In fact, the crucial factor for successful application of GMRES lies in the restart coded in the program. For this reason, the restarting capability is a built-in feature that can yield the above restarted GMRES(m), where m denotes the termination number of iteration. The choice of m, however, has no theoretical foundation and thus is very difficult to determine. Since there is no definitive rule for the choice of m, we determine it through numerical experiments. After conducting extensive investigations, we consider $m = 5$ in all the calculations. For additional details of GMRES(m), see Van der Vorst [18].

4.2. The Bi-CGSTAB iterative solver. The Bi-CG method of Fletcher [34] suffers from instability problem arising from the unsymmetric Lanczos process when

it is used to solve the non-Hermitian system of equations. As a result, considerable effort has been directed toward developing a more stable algorithm. The CGS method is considered as a variant of Bi-CG, known as Bi-CGSTAB. Since the Bi-CGSTAB method is known for its smooth approach to convergence, we will consider this method in our assessment study.

Like the Bi-CG method, Bi-CGSTAB iterates the approximate solution by means of unsymmetric Lanczos vectors. The difficulty arises from the nonsymmetry property of the finite element stiffness matrix $\underline{\underline{\mathbf{A}}}$, which results in the nonexistence of the orthogonal tridiagonalization $\underline{\mathbf{Q}}^T \underline{\underline{\mathbf{A}}}\, \underline{\mathbf{Q}} = \underline{\mathbf{T}}$, where $\underline{\mathbf{T}}$ is a tridiagonal matrix. As a result, partial tridiagonalization of $\underline{\underline{\mathbf{A}}}$ is needed to implement the algorithm. The unsymmetric Lanczos algorithm allows partial tridiagonalization of $\underline{\underline{\mathbf{A}}}$ by making use of two sets of biorthogonal vectors. This approach involves computing columns of $\underline{\mathbf{Q}}$ and $\underline{\mathbf{P}}$, which are subject to $\underline{\mathbf{P}}^T \underline{\mathbf{Q}} = \underline{\mathbf{I}}$, so that $\underline{\mathbf{P}}^T \underline{\underline{\mathbf{A}}}\, \underline{\mathbf{Q}} = \underline{\mathbf{T}}$ is tridiagonal.

It has been known for quite some time that effective use of iterative methods depends highly upon the nonzero profile of the coefficient matrix. The strategies of ordering nodal points and allocating working variables are essential because they have a direct effect on the matrix bandwidth and, thus, matrix sparsity. A means of storing the matrix in the core memory is needed in the finite element analysis, where sparse matrix equations are encountered. Like the compressed matrix used in the finite difference setting, we can store a matrix at the element level so as to dispense with unnecessary storage of voids. This motivates us to conduct finite element analysis on an element-by-element basis. In this paper, we incorporate the element-by-element capability into the Bi-CGSTAB of Van der Vorst [18]:

Compute $\underline{\mathbf{r}}_0 = \underline{\mathbf{b}} - \underline{\underline{\mathbf{A}}}\, \underline{\mathbf{x}}_0$ for an initial guess vector $\underline{\mathbf{x}}_0$
Choose $\overline{\underline{\mathbf{r}}}$, such that $(\overline{\underline{\mathbf{r}}}, \underline{\mathbf{r}}_0) \neq 0$
For $i = 1, 2, \ldots$
  $\rho_{i-1} = (\overline{\underline{\mathbf{r}}}, \underline{\mathbf{r}}_{i-1})$
  if $\rho_{i-1} < \epsilon_1$ [**near break down**]
  if $i = 1$
    $\underline{\mathbf{p}}_i = \underline{\mathbf{r}}_{i-1}$
  else
    $\beta_{i-1} = (\rho_{i-1}/\rho_{i-2})(\alpha_{i-1}/\omega_{i-1})$
    $\underline{\mathbf{p}}_i = \underline{\mathbf{r}}_{i-1} + \beta_{i-1}\,(\underline{\mathbf{p}}_{i-1} - \omega_{i-1}\,\underline{\mathbf{v}}_{i-1})$
  endif
  $\underline{\mathbf{v}}_i = \sum_{elem} (\underline{\underline{\mathbf{A}}}_{elem}\, \underline{\mathbf{p}}_i)$ ⟵ element-by-element procedure
  $\alpha_i = \rho_{i-1}/(\overline{\underline{\mathbf{r}}}, \underline{\mathbf{v}}_i)$
  if $(\overline{\underline{\mathbf{r}}}, \underline{\mathbf{v}}_i) < \epsilon_2$ [**near break down**]
  $\underline{\mathbf{s}} = \underline{\mathbf{r}}_{i-1} - \alpha_i\, \underline{\mathbf{v}}_i$
  $\underline{\mathbf{t}} = \sum_{elem} (\underline{\underline{\mathbf{A}}}_{elem}\, \underline{\mathbf{s}})$ ⟵ element-by-element procedure
  if $\|\underline{\mathbf{s}}\|_2 < \epsilon$
    $\omega_i = 0$
  else
    $\omega_i = (\underline{\mathbf{t}}, \underline{\mathbf{s}})/(\underline{\mathbf{t}}, \underline{\mathbf{t}})$
  endif
  $\underline{\mathbf{x}}_i = \underline{\mathbf{x}}_{i-1} + \alpha_i\, \overline{\underline{\mathbf{p}}}_i + \omega_i\, \overline{\underline{\mathbf{s}}}$
  $\underline{\mathbf{r}}_i = \underline{\mathbf{b}} - \underline{\underline{\mathbf{A}}}\, \underline{\mathbf{x}}_i$
  check convergence; continue if necessary $(\omega_i \neq 0)$
End

For the two iterative methods considered here, the calculation is terminated when the

residual-norm criterion $\|\underline{\mathbf{r}}\|_2 < 10^{-10}$ is satisfied.

4.3. Multifrontal direct solver. One of the significant advances in finite element computations was the frontal direct solver developed in 1970 [21]. The frontal solver begins by assembling the matrix for each element. This is followed by incorporating element matrices into the global system of matrices. The elimination of equations is allowed whenever possible, rather than assembling the whole system of elementary finite element matrices. Instead, we examine whether there exists any row which corresponds to the fully contributed nodes; if there is, we store the row, the variables associated with it, and the right-hand side, and then eliminate this row. This process continues until all the elements have been assembled and the elimination procedure is completed. The calculation of solutions is followed by performing backward-substitution.

The multifrontal direct solver can be refined in different ways. The most important solver of this kind is the one developed in 1983 [20]. As the name indicates, many frontal matrices are involved in the course of applying multifrontal solver. The matrix is divided into several balanced substructures. A tree structure is needed to define the order of assembly of element matrices. After the finite element mesh is partitioned, a frontal method is applied to each user's defined substructure to eliminate the interior nodes. A set of substructure matrices is thus generated to complete the elimination process. This is followed by backward-substitution to obtain finite element solutions.

5. Numerical results. We will first consider test problems which are amenable to analytical solutions in order to demonstrate the validity and usefulness of the TG-FCT finite element model. The first problem is shown schematically in Figure 5.1. In the square of unit length, a sharp scalar profile was set as the initial condition. The centroid of the square profile was located at (0.25, 0.25). This initially discontinuous scalar profile was transported in the flow specified by $u = \sqrt{2}/2$, $v = \sqrt{2}/2$. Rectangular Cartesian grids were uniformly overlaid on the region of interest. The grid spacings were set as $\Delta x = \Delta y = 0.01$, and the time increment for this study was chosen to be $\Delta t = 0.001$. The calculation for this study was terminated at $t = 0.5$. The numerical models were run using iterative and direct solvers with a tolerance equal to 10^{-10}. The result shown in Figure 5.2 clearly indicates that the passive scalar was well predicted without observable oscillations. This validation test shows that the scheme adopted here has the ability to resolve discontinuities.

We now turn to making a comparison of the employed frontal, GMRES(5), and Bi-CGSTAB solution solvers. The user, system, and CPU times shown in Table 5.1 are obtained for the codes run on an Intel Celeron/466 MHz processor. As this table shows, GMRES(5) consumes only 1/9 CPU time as that computed by the frontal solver. As for the Bi-CGSTAB solver, it is slower than the GMRES(5) by a factor of 7/9. One plausible reason for explaining the savings in CPU time of the GMRES solver is due to its relatively regular convergence. Taking $t = \Delta t$ as an example, the convergent histories, shown in Figure 5.3, clearly show that the specified tolerance is reached much more quickly when GMRES is employed. As for the CPU time spent in this arbitrarily chosen time step, the ratio of $\text{CPU}|_{\text{GMRES(5)}}$ / $\text{CPU}|_{\text{Bi-CGSTAB}}$ is equal to 4/5.

The next problem was intended to test the ability of the scheme to resolve discontinuous field variables in shallow water equations. In this validation, we solve for a one-dimensional dam-break problem using the proposed two-dimensional finite element code. The problem, shown in Figure 5.4, involved a dam 100 m in length.

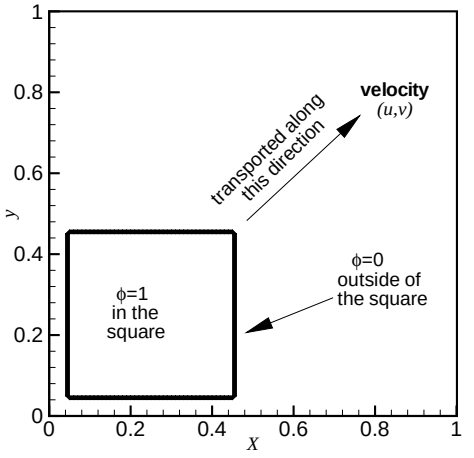
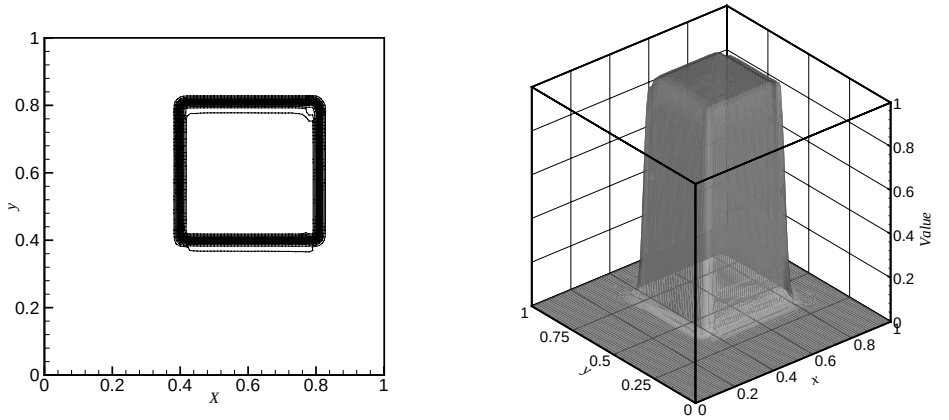


FIG. 5.1. *The schematic illustration of the scalar transport problem.*



(a) (b)

FIG. 5.2. (a) *The contours of the computed solution at* $t = 0.5$*;* (b) *the three-dimensional view of the solution at* $t = 0.5$*. The grid size used for this study is* $\Delta x = \Delta y = 10^{-2}$*.*

TABLE 5.1

The comparison of CPU times (in seconds) for solving the problem, schematically shown in Figure 5.1*, using the frontal, GMRES, and Bi-CGSTAB solvers.*

Method	User time	System time	CPU time
frontal [21]	20842	1187	22029
GMRES(5) [19]	1898	664	2562
Bi-CGSTAB [18]	2409	664	3073

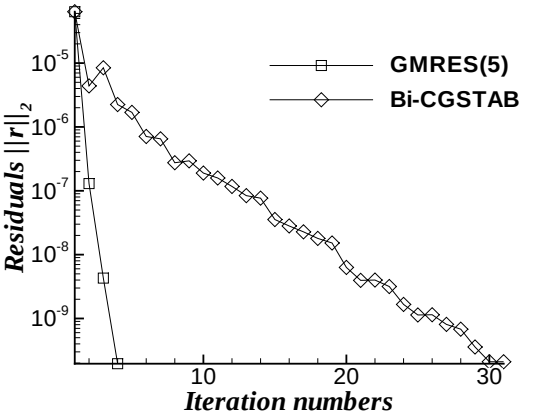


FIG. 5.3. *The convergent histories of the GMRES*(5) *and Bi-CGSTAB methods within a time step* $0 \le t \le \Delta t$ *for the transport problem given in Figure* 5.1.

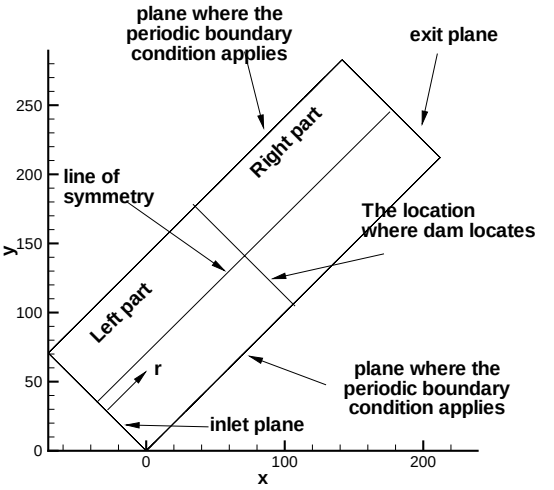


FIG. 5.4. *The configuration of the one-dimensional dam-break problem.*

For the present study, a channel 300 m long and 100 m wide was used and was discretized into 301×7 grids for numerical calculation. The case under investigation was a subcritical flow with a water-height ratio of 2. Both the upstream and downstream boundary conditions remained unchanged during the calculation. For this purpose, the test was run at $t = 10$ s.

Figure 5.5 shows the water height, which compares very favorably with the following analytic data of Stoker [38]:

$$h(\tilde{x},t) = \begin{cases} h_1 & \text{if } \frac{\tilde{x}}{t} \le -\sqrt{gh_1}, \\ \left(\frac{1}{9g}\right)\left[2\sqrt{gh_1} - \frac{\tilde{x}}{t}\right]^2 & \text{if } -\sqrt{gh_1} \le \frac{\tilde{x}}{t} \le \left[u_m - \sqrt{gh_m}\right], \\ h_m & \text{if } \left[u_m - \sqrt{gh_m}\right] \le \frac{\tilde{x}}{t} \le s, \\ h_2 & \text{if } s \le \frac{\tilde{x}}{t} \le \infty. \end{cases} \tag{5.1}$$

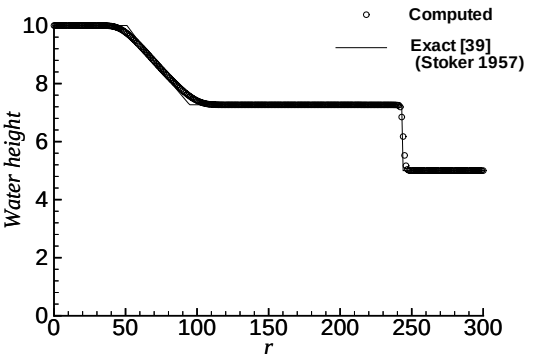


FIG. 5.5. *The comparison of the solution of the one-dimensional dam-break problem with the Stoker's analytic solution. The grid size for this problem is* $\Delta r = 1$.

We denote here $\tilde{x} = x - x_0$, where x_0 is the location of the discontinuity. In the above, h_m and u_m are the water height and velocity in the middle of this channel, and their values are related to the shock propagation speed s:

$$h_m = \frac{1}{2}\left[\sqrt{1 + \frac{8s^2}{gh_2}} - 1\right] h_2, \tag{5.2}$$

$$u_m = s - \frac{gh_2}{4s}\left[1 + \sqrt{1 + \frac{8s^2}{gh_2}}\right]. \tag{5.3}$$

The shock speed s is the positive real root of the following equation:

$$u_m + 2\sqrt{gh_m} - 2\sqrt{gh_1} = 0. \tag{5.4}$$

It is seen from the computed solutions that the shock wave can be resolved within 4 mesh points. No postshock oscillations are observed in the solution. Also, the improved accuracy is attributable to the high-order Taylor–Galerkin scheme, which is applied in regions away from the discontinuity.

We will now consider wave propagation in a basin of simple geometry. Figure 5.6 shows a typical configuration extensively used to study shallow water where bore waves may develop. At the midpoint of the square basin, a dam with a width of 10 m equally divides the water into two parts. On both sides of this idealized dam, the water elevations have a height ratio other than 1. At time $t = 0^+$, the dam is partially broken, leading to a breach with a width of 75 m. The resulting flow pattern in this partially breached dam depends on whether it is classified as being subcritical or supercritical. The case examined here is a subcritical flow with $h_L/h_R = 2$, where $h_L\,(\equiv 10$ m) denotes the initial water elevation on the left side of the basin while $h_R\,(\equiv 5$ m) represents the water height on the right side. Given that the ratio h_L/h_R has a value other than 1, water proceeds toward the downstream side through the breach, which is located in the region $y = 95$ m to $y = 170$ m. When the dam breaks, a bore wave starts to propagate forward and spread laterally. At the same time, a negative depression wave spreads upstream. In addition, a standing wave will appear

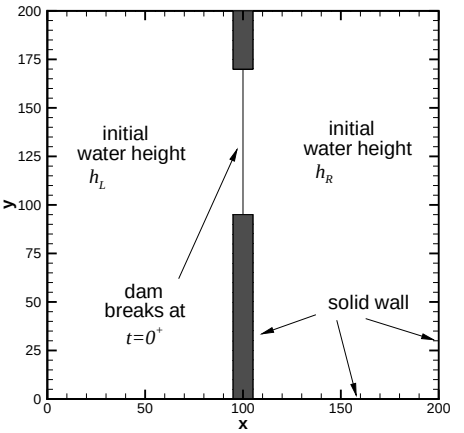


FIG. 5.6. *The configuration of the two-dimensional dam-break problem.*

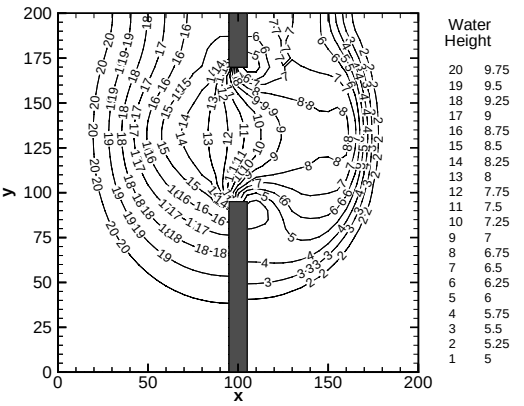


FIG. 5.7. *The contours of water height of the two-dimensional dam-break problem.*

due to the reduction of velocity at the two side walls. Our goal is to numerically predict the time-evolving propagation and spreading of the wave into the reservoir.

With the Courant number set as 0.1, the numerical code was run on a domain of uniform spacing, $\Delta x = \Delta y = 5$. Figure 5.7 plots the contour values of the water elevation obtained using the proposed TG-FCT method at $t = 7.2$ s. These values show that the right traveling bore wave and left traveling depression wave have both been predicted. The results also reveal the ability of the FCT scheme to capture sharp solutions, as seen from the abrupt depression of the water surface elevation in the vicinity of the breach edge. The appearance of this sharp depression wave is theoretically justified since strong rarefactions are established in the inviscid flow regime where large velocity gradients appear. In light of this fact, we plot in Figure 5.8 the velocity vector to show the sharp depression in the water surface elevation around the breach edge. From this velocity vector plot, we are led to conclude that the regions with large velocity gradients observed near the edge of the breach appear to be those which are most subject to abrupt depression in the water surface elevation.

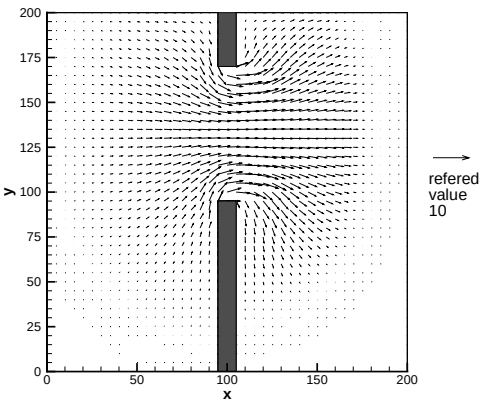


FIG. 5.8. *The velocity vector of the two-dimensional dam-break problem.*

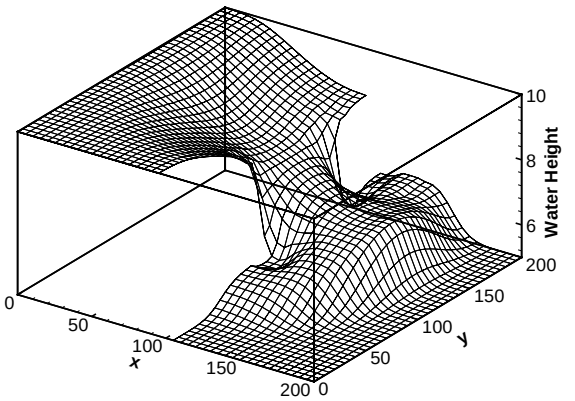


FIG. 5.9. *The three-dimensional view of the water height of the two-dimensional dam-break problem.*

To present the solution clearly, we plot in Figure 5.9 a three-dimensional water surface at $t = 7.2$ s after the dam breaks.

Having demonstrated the advantage of using the GMRES(5) solver over the Bi-CGSTAB solver, we simply consider the GMRES solver in the shallow water calculation and make a computational assessment with the two employed direct solvers. Table 5.2 shows that GMRES(5) is ten times faster than the frontal solver. As for the multifrontal direct solver, it takes only 1/3 of the CPU time needed for the direct solver. Note that the times summarized in Table 5.2 are obtained on an SGI Origin 2000 computer. This performance test demonstrates the effective utility of the multifrontal solver and, more importantly, ensures the advantage of applying the GMRES iterative solver to shallow water analyses.

To show that this method is applicable to predicting a more severely changing solution profile, we considered a supercritical flow. The initial water height ratio was prescribed as $h_R/h_L = 0.05$. As Figure 5.10 shows, water heights were captured in a sharp and nonoscillatory way. This test also sheds light on the effectiveness of adding entropy flux to the region where needed (near the sonic point) since no expansion

TABLE 5.2

The comparison of CPU times (in seconds) for solving the dam-break problem using the frontal, multifrontal, and GMRES solvers.

Method	User time	System time	CPU time
frontal [21]	28961	199	29160
multifrontal [20]	9657	142	9799
GMRES(5) [19]	2816	143	2959

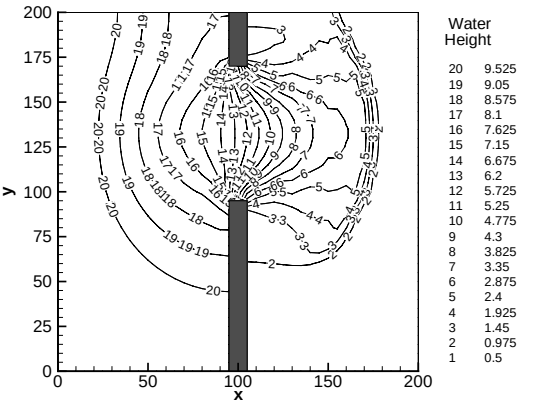


FIG. 5.10. *The contours of water height of the two-dimensional dam-break problem with* $h_R/h_L = 0.05$.

shock is observed. It is thus concluded that the FCT technique incorporated into the Taylor–Galerkin formulation has the ability to suppress dispersive errors near a discontinuity without adding dissipation error which could deteriorate the solution.

6. Concluding remarks. We have applied in this paper a generalized Taylor–Galerkin finite element model to simulate shallow water equations in two dimensions. By prescribing different sets of free parameters a priori, we can render the technique suitable for hydraulic problems having sharply or smoothly varying solution profiles. We have applied the FCT filtering scheme to obtain high-resolution solutions. The main idea of developing the high-order TG-FEM model is the adoption of modified equation analysis. As for the free parameters used in the low-order Taylor–Galerkin model, we employ the discrete maximum principle to construct a stiffness matrix of the M-matrix type. The avoidance of numerical oscillations near the discontinuity and the higher level of prediction accuracy in the smooth region make this scheme a robust tool for solving differential equations governing shallow water height. The code was run on several test problems to study the method's performance and the solver's efficiency, with particular attention paid to the shock-capturing ability and the computational advantage. For the purpose of validation, we have chosen ones for which exact solutions are available. These include the scalar equation and the shallow water equations. Numerical results show that field variables were captured in a sharp and nonoscillatory way in both subcritical and supercritical situations. Through computational exercises, we advocate the use of iterative solution solvers. Of two investigated iterative solvers, the GMRES outperforms the Bi-CGSTAB solver. The present study also shows the advantage of applying the multifrontal direct solver over the frontal direct solver as far as the present shallow water analysis is considered.

Acknowledgment. The authors would like to thank the National Center for High-Performance Computing (NCHC) for providing the SGI Origin 2000 computer which made this study possible.

REFERENCES

[1] E. F. Toro, *Riemann problems and the WAF method for solving the two-dimensional shallow water equations*, Philos. Trans. Roy. Soc. London Ser. A, 338 (1992), pp. 43–68.
[2] S. K. Godunov, *Finite-difference method for numerical computation of discontinuous solutions of the equations of fluid dynamics*, Mat. Sb., 47 (1959), pp. 271–306 (in Russian).
[3] B. Van Leer, *Towards the ultimate conservative difference scheme,* II. *Monotonicity and second order combined in a second order scheme*, J. Comput. Phys., 14 (1973), pp. 361–376.
[4] A. Harten, *High-resolution schemes for hyperbolic conservation laws*, J. Comput. Phys., 49 (1983), pp. 357–393.
[5] P. Colella and P. R. Woodward, *The piecewise parabolic method (PPM) for gas dynamics simulation*, J. Comput. Phys., 54 (1984), pp. 174–201.
[6] A. Harten and S. Osher, *Uniformly high-order accurate nonoscillatory schemes.* I, SIAM J. Numer. Anal., 24 (1987), pp. 279–309.
[7] P. Garcia-Navaroo, M. E. Hubband, and A. Priestley, *Genuinely multidimensional upwinding for the* 2D *shallow water equations*, J. Comput. Phys., 121 (1995), pp. 79–93.
[8] H. Paillere, G. Dergez, and H. Deconinck, *Multidimensional upwind schemes for the shallow water equations*, Internat. J. Numer. Methods Fluids, 26 (1998), pp. 987–1000.
[9] T. Molls and F. Molls, *Space-time conservation method applied to Saint Venant equations*, J. Hydr. Engrg., 124 (1998), pp. 501–508.
[10] C. G. Mingham and D. M. Causon, *High-resolution finite-volume method for shallow water flows*, J. Hydr. Engrg., 124 (1998), pp. 605–614.
[11] R. J. Fernema and M. H. Chaudhry, *Explicit methods for* 2*-D transient free-surface flows*, J. Hydr. Engrg., 116 (1990), pp. 1013–1034.
[12] L. Fraccarollo and E. Toro, *Experimental and numerical assessment of the shallow water model for two dimensional dam-break type problems*, J. Hydr. Res., 33 (1995), pp. 843–864.
[13] R. Garcia and R. Kahawita, *Numerical solution of the St. Venant equations with MacCormack finite-difference scheme*, Internat. J. Numer. Methods Fluids, 6 (1986), pp. 507–527.
[14] D. H. Zhao, H. W. Shen, J. S. Lai, and G. Q. Tabios, *Approximate Riemann solvers in FVM for* 2D *hydraulic wave modeling*, J. Hydr. Engrg., 122 (1996), pp. 692–702.
[15] N. D. Katopodes and T. Strelkoff, *Computing two-dimensional dam-break flood waves*, J. Hydr. Div., 110 (1988), pp. 1269–1288.
[16] J. P. Boris and D. L. Book, *Flux corrected transport, SHASTA, a fluid transport algorithm that works*, J. Comput. Phys., 11 (1973), pp. 38–69.
[17] S. T. Zalesak, *Fully multidimensional flux-corrected transport algorithm for fluids*, J. Comput. Phys., 31 (1979), pp. 335–362.
[18] H. A. Van der Vorst, *BI-CGSTAB: A fast and smoothly converging variant of BI-CG for the solution of nonsymmetric linear system*, SIAM J. Sci. Statist. Comput. 13 (1992), pp. 631–644.
[19] Y. Saad and M. H. Schultz, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear system*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 856–869.
[20] I. S. Duff and J. K. Reid, *The multifrontal solution of indefinite sparse symmetric linear equations*, ACM Trans. Math. Software, 9 (1983), pp. 302–325.
[21] B. M. Irons, *A frontal solution program for finite element analysis*, Internat. J. Numer. Methods Engrg., 4 (1970), pp. 5–32.
[22] J. Donea, *A Taylor-Galerkin method for convective transport problems*, Internat. J. Numer. Methods Engrg., 20 (1984), pp. 101–119.
[23] W. H. Sheu and C. C. Fang, *A numerical study of nonlinear propagation of disturbances in two-dimensions*, J. Comput. Acoustics, 4 (1996), pp. 291–319.
[24] M. H. Chaudhry, *Open-Channel Flow*, Prentice–Hall, Englewood Cliffs, NJ, 1993.
[25] G. A. Sod, *Numerical Methods in Fluid Dynamics*, Cambridge University Press, Cambridge, UK, 1985.
[26] T. Meis and U. Marcowitz, *Numerical Solution of Partial Differential Equations*, Appl. Math. Sci. 32, Springer-Verlag, New York, 1981.
[27] T. Ikeda, *Maximal Principle in Finite Element Models for Convection-Diffusion Phenomena*, Lecture Notes Numer. Appl. Anal. 4, Kinokuniya, Tokyo, Japan, 1983.

[28] M. Ahués and M. Telias, *Petrov-Galerkin Scheme for the Steady State Convection Diffusion Equation*, Finite Elements in Water Resources 2/3, 1982.
[29] A. Harten, P. D. Lax, and B. van Leer, *On upstream differencing and Godunov-type schemes for hyperbolic conservation laws*, SIAM Rev., 25 (1983), pp. 35–61.
[30] Y. E. Campbell, *Multifrontal Algorithms for Sparse Inverse Subsets and Incomplete LU Factorization*, Technical Report TR-95-025, Computer and Information Sciences Department, University of Florida, Gainesville, FL, 1995.
[31] M. R. Hestenes and E. Stiefel, *Methods of conjugate gradients for solving linear systems*, J. Research Nat. Bur. Standards, 49 (1952), pp. 409–436.
[32] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 2nd ed., North Oxford Academic, Oxford, 1986.
[33] R. Varga, *Matrix Iterative Analysis*, Prentice–Hall, Englewood Cliffs, NJ, 1962.
[34] R. Fletcher, *Conjugate Gradient Methods for Indefinite Systems*, Lecture Notes in Math. 506, Springer-Verlag, New York, 1976, pp. 73–89.
[35] P. Sonneveld, *CGS, a fast Lanczos-type solver for nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 10 (1989), pp. 36–52.
[36] R. Freund and N. Nachtigal, *QMR: A quasi-minimal residual method for non-Hermitian linear system*, Numer. Math., 60 (1991), pp. 315–339.
[37] R. W. Freund, *A transpose-free quasi-minimal residual algorithm for non-Hermitian linear systems*, SIAM J. Sci. Comput., 14 (1993), pp. 470–482.
[38] J. J. Stoker, *Water Waves*, Interscience, New York, 1957.

SIAM J. SCI. COMPUT.
Vol. 22, No. 6, pp. 2093–2108
© 2001 Society for Industrial and Applied Mathematics

COULOMB INTERACTIONS ON PLANAR STRUCTURES: INVERTING THE SQUARE ROOT OF THE LAPLACIAN*

ZYDRUNAS GIMBUTAS†, LESLIE GREENGARD‡, AND MICHAEL MINION§

Abstract. We present an adaptive fast multipole method for inverting the square root of the Laplacian in two dimensions. Solving this problem is the dominant computational cost in many applications arising in electrical engineering, geophysical fluid dynamics, and the study of thin films. It corresponds to the evaluation of the field induced by a planar distribution of charge or vorticity. Our algorithm is direct and assumes only that the source distribution is discretized using an adaptive quad-tree. The amount of work grows linearly with the number of mesh points.

Key words. quasi-geostrophic fluid dynamics, integral equation methods, thin films, planar circuits

AMS subject classifications. 31B10, 47G30, 65R10, 78A30, 86A10

PII. S1064827599361199

1. Introduction.

In this paper, we present a fast, adaptive, numerical method for solving the pseudodifferential equation

$$(-\Delta)^{1/2}\psi = \omega \tag{1.1}$$

in the plane, where Δ denotes the Laplacian operator. This equation appears in a variety of different mathematical models for problems in surface physics, some of which are described below. We assume that $\omega(x) \in L^2(\mathbf{R}^2)$, with Fourier transform

$$\hat{\omega}(\mathbf{k}) = \int_{\mathbf{R}^2} \omega(\mathbf{x}) e^{-2\pi i\, \mathbf{k}\cdot\mathbf{x}} d\mathbf{x}.$$

The symbol of $-\Delta$ is $1/(2\pi|\mathbf{k}|)^2$, so that the solution to (1.1) can be written as

$$\psi(\mathbf{x}) = \int_{\mathbf{R}^2} \frac{\hat{\omega(\mathbf{k})}}{2\pi|\mathbf{k}|}\, e^{2\pi i\, \mathbf{k}\cdot\mathbf{x}} d\mathbf{k}. \tag{1.2}$$

Alternatively, we can seek a Green's function for this operator by solving the equation

$$(-\Delta)^{1/2} G(\mathbf{x}) = \delta(\mathbf{x}).$$

A straightforward calculation yields

$$G(\mathbf{x}) = \int_{\mathbf{R}^2} \frac{1}{2\pi|\mathbf{k}|} e^{2\pi i \mathbf{k}\cdot\mathbf{x}} d\mathbf{k} = \frac{1}{2\pi|\mathbf{x}|}, \tag{1.3}$$

*Received by the editors September 15, 1999; accepted for publication (in revised form) September 27, 2000; published electronically March 28, 2001.
http://www.siam.org/journals/sisc/22-6/36119.html

†Program in Applied and Computational Mathematics, Princeton University, Princeton, NJ 08544 (gimbutas@princeton.edu). The work of this author was supported in part by AFOSR under grant F49620-97-1-0011.

‡Courant Institute of Mathematical Sciences, New York University, New York, NY 10012 (greengard@cims.nyu.edu). The work of this author was supported by the Applied Mathematical Sciences Program of the U.S. Department of Energy under contract DEFGO288ER25053.

§Department of Mathematics, University of North Carolina, Chapel Hill, NC 27599 (minion@amath.unc.edu). The work of this author was supported in part by the Applied Mathematical Sciences Program of the U.S. Department of Energy under contract DEFGO288ER25053.

from which the inversion formula

$$\psi(\mathbf{x}) = \int_{\mathbf{R}^2} \frac{\omega(\mathbf{y})}{2\pi|\mathbf{x}-\mathbf{y}|}\, d\mathbf{y} \tag{1.4}$$

follows.

Since the function $G(\mathbf{x})$ obtained in (1.3) is the Green's function for the three-dimensional Laplacian, we can think of (1.1) and (1.4) as describing a three-dimensional Poisson equation for which $\omega(\mathbf{x})$ is a singular density lying entirely on the plane $z = 0$. Applications involving such surface interactions include the study of planar circuits in electrical engineering, a variety of problems in thin films [5, 19, 20], and certain problems in tomography [7]. In quasi-geostrophic fluid dynamic models [17], one encounters an equation analogous to the incompressible Navier–Stokes equations which take the form

$$\begin{aligned} \omega_t + (U \cdot \nabla)\omega &= \nu \Delta \omega, \\ U &= \nabla^{\perp}\psi, \\ (-\Delta^{1/2})(-\psi) &= \omega, \end{aligned} \tag{1.5}$$

where U is a velocity field, ω is a vorticity-like variable, and ψ is a stream function. In the past three years, these equations have seen a new wave of interest stemming, in part, from observations made in [6] which draw interesting analogies between the quasi-geostrophic equations and the three-dimensional incompressible Euler equations. Another interesting application is a two-fluid model system which supports internal waves and which can be applied to a variety of geophysical model systems [3]. One specific example is a vertically stratified fluid in which one layer is much thinner than the other, and in which the square root of the Laplacian plays a fundamental role.

All of the preceding application areas would benefit from adaptive numerical simulation tools in order to resolve complex solution features. Most existing numerical methods for solving (1.1), however, rely on the spectral form of the solution given in (1.2) and are implemented via the fast Fourier transform. This approach precludes the use of adaptivity and constrains the computational domain to be periodic.

Direct evaluation of the convolution integral (1.4) involves interactions between all pairs of grid points. Hence, with N points in the discretization, the work involved would require $O(N^2)$ operations. In this paper, we describe two special purpose fast multipole methods (FMMs) for the computation of the integral transform (1.4) for which the work required grows linearly with the number of grids points. This approach allows for adaptive mesh refinement and the imposition of either free-space or periodic boundary conditions. Related methods have been developed in [2, 14, 16].

2. Data structures and the FMM. We assume that the source distribution ω in (1.1) is supported inside the unit square D, centered at the origin, on which is superimposed a hierarchy of refinements (a quad-tree). Grid level 0 is defined to be D itself, and grid level $l+1$ is obtained recursively by subdividing each square at level l into four equal parts. Using standard terminology, if d is a fixed square at level l, the four squares at level $l+1$ obtained by its subdivision will be referred to as its children. In order to allow for adaptivity, we do not use the same number of levels in all regions of D. We do, however, assume that the quad-tree satisfies one fairly standard restriction, namely, that two leaf nodes which share a boundary point must be no more than one refinement level apart (Figure 1).

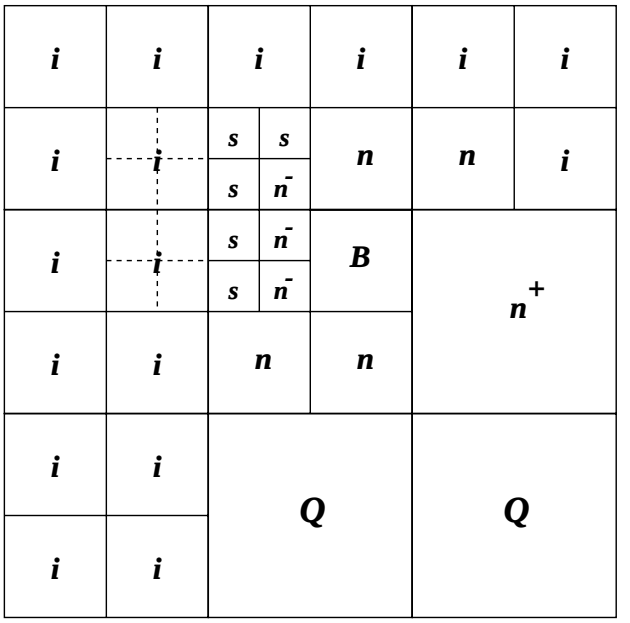


FIG. 1. *For the childless node* $\boldsymbol{B}$*, colleagues are labeled by* $\boldsymbol{n}$*, coarse neighbors are labeled* $\boldsymbol{n}^+$*, and fine neighbors are labeled* $\boldsymbol{n}^-$*. The interaction list for* B *consists of the boxes marked by* $\boldsymbol{i}$ *as well as those marked by* $\boldsymbol{Q}$*. The boxes marked by* $\boldsymbol{s}$ *are children of* B*'s colleagues which are separated from* B*, so they are not fine neighbors. They constitute the* s-list *for* B *(see Definition* 2.1*).*

The leaf nodes on which the source distribution is given will be denoted by D_i. Thus, $D = \cup_{i=1}^{M} D_i$ and we rewrite (1.4) in the form

$$\psi(\mathbf{x}) = -\sum_{i=1}^{M} \int_{D_i} \frac{\omega(\mathbf{y})}{2\pi|\mathbf{x}-\mathbf{y}|}\, d\mathbf{y}. \tag{2.1}$$

DEFINITION 2.1. *The* colleagues *of a square* B *are squares at the same refinement level which share a boundary point with* B*. (*B *is considered to be a colleague of itself.) The* coarse neighbors *of* B *are leaf nodes at the level of* B*'s parent which share a boundary point with* B*. The* fine neighbors *of* B *are leaf nodes one level finer than* B *which share a boundary point with* B*. Together, the union of the colleagues and coarse and fine neighbors of* B *will be referred to as* B*'s* neighbors*. The* s-list *of a box* B *consists of those children of* B*'s colleagues which are not fine neighbors of* B*.*

The interaction region *for* B *consists of the area covered by the neighbors of* B*'s parent, excluding the area covered by* B*'s colleagues and coarse neighbors. The* interaction list *for* B *consists of those squares in the interaction region which are at the same refinement level, as well as leaf nodes in the interaction region which are at coarser levels. When the distinction is important, the squares at the same refinement level will be referred to as the* standard interaction list*, while the squares at coarser levels will be referred to as the* coarse interaction list*.*

In our FMM, following [4, 11, 12], terms in the convolution integral (2.1) from neighbor leaf nodes are computed directly. More distant interactions are accounted for on coarser levels through the use of a hierarchy of far-field and local multipole expansions. We consider the local interactions first.

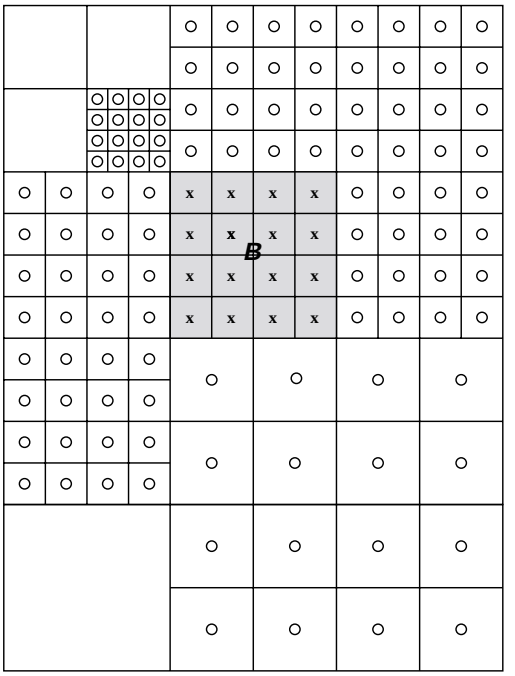


FIG. 2. *The source distribution ω is given on a cell-centered 4×4 grid in the central square B. The field induced by the distribution on B's neighbors can be tabulated and stored. In the adaptive grid, neighbors can be at the same refinement level as B, one coarser or one finer.*

2.1. Local interactions. We assume that we are given ω on a cell-centered 4×4 grid for each leaf node B. We can, therefore, take these 16 data points and construct a fourth-order polynomial approximation to ω of the form

$$\omega_B(x,y) \approx \sum_{j=1}^{10} c_B(j)\, b_j(x - x_B, y - y_B),$$

where (x_B, y_B) denotes the center of B. The basis functions $b_j(x,y)$ are given by

$$1,\ x,\ y,\ x^2,\ xy,\ y^2,\ x^3,\ x^2y,\ xy^2,\ y^3$$

for $j = 1,\ldots,10$, respectively. If we let $\vec{\omega_B} \in \mathbf{R}^{16}$ denote the discrete function values in standard ordering, then the calculation of the coefficient vector $\vec{c}_B$ is clearly overdetermined. We obtain it through a least squares fit based on the singular value decomposition. The pseudoinverse matrix $\mathcal{P} \in \mathbf{R}^{10\times16}$, such that

$$\vec{c}_B = \mathcal{P}\,\vec{\omega_B}$$

can be precomputed and stored.

Consider now a target point Q, which lies in a neighbor of B (Figure 2). Then, the field induced by ω_B is approximated by

$$\psi_B(Q) = \sum_{j=0}^{10} c_B(j) w(Q,j), \tag{2.2}$$

where

$$w(Q,j) = \int_B \frac{b_j(x - x_B, y - y_B)}{2\pi|Q - (x,y)|}\, dxdy. \tag{2.3}$$

Since the target points Q are regularly spaced in each neighboring square, we can precompute the weights (2.3) for each of the sixteen possible locations at each of 9 possible colleagues, 12 possible fine neighbors, and 8 possible coarse neighbors. To be more precise, we can precompute the weights assuming that B is the unit square $[-0.5, 0.5]^2$, because of the following straightforward lemma.

LEMMA 2.2. *Let B be a leaf node at level l and let Q denote a target point in one of B's neighbors. Let Q^* denote the scaled target point for the unit cell centered at the origin*

$$Q^* = 2^l \cdot (Q - (x_B, y_B)),$$

and let

$$w^*(Q^*, j) = \int_{-1/2}^{1/2} \int_{-1/2}^{1/2} \frac{b_j(x,y)}{2\pi|Q^* - (x,y)|}\, dxdy. \tag{2.4}$$

Then the integral $w(Q,j)$ defined in (2.3) *is given by*

$$w(Q,j) = w^*(Q^*, j) \cdot 2^{-[d+1]l},$$

where d is the degree of the polynomial basis function b_j.

Thus, we need only obtain weights for a box of unit area. Elementary counting arguments show that the storage required for this precomputation is

$$\begin{aligned} 16 \cdot 10 \cdot 9 &\text{ real numbers} \quad \text{for colleagues,} \\ 16 \cdot 10 \cdot 12 &\text{ real numbers} \quad \text{for fine neighbors,} \\ 16 \cdot 10 \cdot 8 &\text{ real numbers} \quad \text{for coarse neighbors,} \end{aligned}$$

for a total of 4640 real numbers.

2.2. Far-field interactions. We turn now to the calculation of far-field interactions, which are computed by means of multipole expansions. We refer the reader to [9, 15, 18] for more detailed discussions of potential theory. Our starting point is the usual multipole expansion for a charge distribution, which we state formally as a theorem.

THEOREM 2.3 (multipole expansion). *Let $\sigma(T)$ be a charge distribution contained within S, a sphere of radius a centered at the origin, and let $Q = (r, \theta, \phi) \in \mathbb{R}^3$ with $r > a$. Then the field at Q induced by the charge distribution*

$$\Phi(Q) = \int_S \frac{\sigma(T)}{2\pi|T - Q|}\, dT$$

can be described by the multipole expansion

$$\Phi(Q) = \sum_{n=0}^{\infty} \sum_{m=-n}^{n} \frac{M_n^m}{r^{n+1}} \cdot P_n^m(\cos\theta) e^{im\phi}\ , \tag{2.5}$$

where P_n^m denotes the standard associated Legendre function and

$$M_n^m = \frac{(n-|m|)!}{(n+|m|)!}\int_S \frac{1}{2\pi}\sigma(T)\cdot\rho^n\cdot P_n^{|m|}(\cos\alpha)e^{im\beta}dT. \tag{2.6}$$

In the preceding expression, (ρ,α,β) are the spherical coordinates of T. Furthermore, for any $p\geq 1$,

$$\left|\Phi(Q)-\sum_{n=0}^{p}\sum_{m=-n}^{n}\frac{M_n^m}{r^{n+1}}\cdot P_n^m(\cos\theta)e^{im\phi}\right| \leq \frac{1}{2\pi}\left(\frac{\int_S|\sigma(T)|\,dT}{r-a}\right)\left(\frac{a}{r}\right)^{p+1}. \tag{2.7}$$

In the setting of the present paper, the sources and targets are restricted to the plane $z=0$, so that in the preceding formulas, $Q=(r,\pi/2,\phi)$ and $T=(\rho,\pi/2,\beta)$ when expressed in spherical coordinates. Thus, for a charge distribution σ supported in a square D centered at the origin and a target point Q lying in the interaction list for B, $\Phi(Q)$ can be expressed as a multipole expansion of the form

$$\Phi(Q)=\sum_{n=0}^{\infty}\sum_{m=-n}^{n}M_n^m\cdot\frac{e^{im\phi}}{r^{n+1}}, \tag{2.8}$$

with

$$M_n^m=\frac{(n-|m|)!}{(n+|m|)!}[P_n^m(0)]^2\int_B\frac{1}{2\pi}\sigma(T)\rho^n e^{im\beta}\,dT, \tag{2.9}$$

where (ρ,β) are the polar coordinates of T (Figure 3). The error estimate (2.7) takes the special form

$$\left|\Phi(Q)-\sum_{n=0}^{p}\sum_{m=-n}^{n}M_n^m\cdot\frac{e^{im\phi}}{r^{n+1}}\right|\leq\frac{1}{2\pi}\left(\frac{\int_D|\sigma(T)|\,dT}{2a}\right)\left(\frac{\sqrt{2}}{3}\right)^{p+1}, \tag{2.10}$$

where a^2 is the area of D.

Within the FMM, it is convenient to be able to describe the field within a region due to sources which are far away. For this, suppose that Q lies in D and that

$$\Psi(Q)=\int_S\frac{\sigma(P)}{2\pi|P-Q|}\,dP,$$

where the region S lies outside the nine colleagues of D (Figure 3). Then

$$\Psi(Q)=\sum_{n=0}^{\infty}\sum_{m=-n}^{n}L_n^m\cdot r^n\,e^{im\phi}, \tag{2.11}$$

with

$$L_n^m=\frac{(n-|m|)!}{(n+|m|)!}[P_n^m(0)]^2\int_S\frac{1}{2\pi}\sigma(P)\frac{e^{im\theta}}{\rho^{n+1}}\,dP, \tag{2.12}$$

where (ρ,θ) are the polar coordinates of P. Furthermore,

$$\left|\Psi(Q)-\sum_{n=0}^{p}\sum_{m=-n}^{n}L_n^m\cdot r^n\,e^{im\phi}\right|\leq\frac{1}{2\pi}\left(\frac{\|\sigma(P)\|_{L_1}}{a}\right)\left(\frac{\sqrt{2}}{3}\right)^{p+1}. \tag{2.13}$$

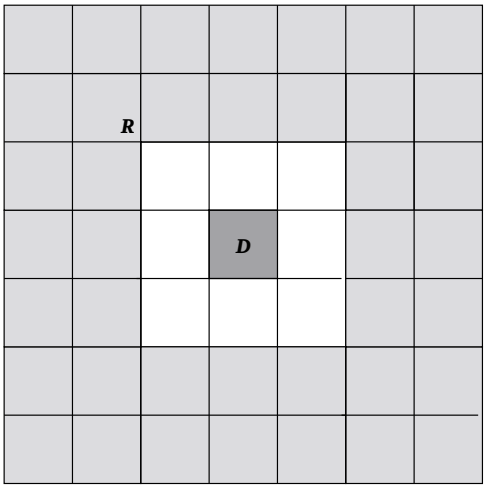


FIG. 3. *For the box D, the interaction region is contained within the outer shaded region.*

The FMM relies on the ability to manipulate multipole and local expansions for every box in the tree hierarchy. We omit the technical details and refer the reader to the original papers [4, 9, 11, 12].

DEFINITION 2.4. *We denote by $S_{l,k}$ the kth square at refinement level l.*

We denote by $\Phi_{l,k}$ the multipole expansion describing the far field due to the source distribution supported inside $S_{l,k}$.

We denote by $\Psi_{l,k}$ the local expansion describing the field due to the source distribution outside the neighbors of $S_{l,k}$.

We denote by $\tilde{\Psi}_{l,k}$ the local expansion describing the field due to the source distribution outside the neighbors of the parent *of $S_{l,k}$.*

REMARK 2.1. *Let $S_{l,k}$ be a square in the quad-tree hierarchy and let $S_{l',k'}$ be a square in its interaction list. Then there is a linear operator $\mathcal{T}_{MM}$ for which*

$$\Phi_{l,k} = \mathcal{T}_{MM}[\Phi(C_1), \Phi(C_2), \Phi(C_3), \Phi(C_4)], \tag{2.14}$$

where $\Phi(C_j)$ denotes the multipole expansion for the jth child of $S_{l,k}$. In other words, we can merge the expansions for four children into a single expansion for the parent. Similarly, there is a linear operator $\mathcal{T}_{LL}$ for which

$$[\tilde{\Psi}(C_1), \tilde{\Psi}(C_2), \tilde{\Psi}(C_3), \tilde{\Psi}(C_4)] = \mathcal{T}_{LL}\, \Psi_{l,k}, \tag{2.15}$$

where C_j denotes the jth child of $S_{l,k}$. In other words, we can shift the local expansion Ψ for a box to the corresponding expansion $\tilde{\Psi}$ for each of its children. Finally, there is a linear operator $\mathcal{T}_{ML}$ for which the field in $S_{l,k}$ due to the source distribution in $S_{l',k'}$ is described by $\Psi = T_{ML}\Phi_{l',k'}$. It is easy to verify that

$$\Psi_{l,k} = \tilde{\Psi}_{l,k} + \sum_{i \in \mathcal{IL}} T_{ML}\Phi_i, \tag{2.16}$$

where $\mathcal{IL}$ denotes the interaction list for square $S_{l,k}$.

REMARK 2.2. *One slight complication in the adaptive algorithm concerns the interaction between boxes of different sizes. Referring to Figure* 1*, we need to account for the influence of a childless square B on each box marked s and vice versa.*

(This interaction clearly doesn't arise if B undergoes further refinement.) For the box marked s, its multipole expansion is rapidly convergent at each of the sixteen target points in B. Thus, its influence can be computed by direct evaluation of the truncated series. For the reverse, however, note that B's multipole expansion is not so rapidly convergent. In this case, we directly compute the coefficients of the local expansion in s from the formula (2.12). *A more precise statement than* (2.16) *is*

$$\Psi_{l,k} = \tilde{\Psi}_{l,k} + \sum_{i \in SIL} T_{ML}\Phi_i, + \sum_{i \in CIL} L_{direct}(\vec{\omega}_i), \tag{2.17}$$

where SIL denotes the standard interaction list and CIL denotes the coarse interaction list. The operator L_{direct}, which maps the coefficients of the polynomial approximation of the density in the coarse box onto the p^2 coefficients of the local expansion can be precomputed and stored.

The bulk of the work in the FMM involves the application of the operators $\mathcal{T}_{MM}, \mathcal{T}_{ML}, \mathcal{T}_{LL}$ in a systematic fashion. Unfortunately, these operators are dense. Using multipole and local expansions truncated after p^2 terms, the naive cost of application is proportional to p^4. Modern versions of the FMM have reduced this cost to $O(p^3)$ or $O(p^2)$ [12].

ALGORITHM.

Initialization

Comment [We assume we are given a square domain $D = S_{0,0}$, on which is superimposed an adaptive hierarchical quad-tree structure. We let M be the number of leaf nodes and denote them by $D_i,\ i = 1, \ldots, M$. The number of grid points is, therefore, $N = 16M$. We let p denote the order of the multipole expansion ($p \approx \log_2 \epsilon$, where ϵ is the desired accuracy). We let l_{max} denote the maximum refinement level.]

Step I: Multipole sweep

Upward pass

for $l = levmax, \ldots, 0$
  for all boxes j on level l
    if j is childless **then**
      form the multipole expansion $\Phi_{l,j}$ from (2.9)
    else
      form the multipole expansion $\Phi_{l,j}$ by merging the expansions of its children using the operator $\mathcal{T}_{MM}$ (see (2.14))
  end
end

Downward pass

Initialize the local expansion $\Psi_{0,0} = 0$.
for $l = 1, \ldots, levmax$
  for all squares j on level l
    Compute $\tilde{\Psi}_{l,j}$ by shifting its parent's Ψ expansion using the operator $\mathcal{T}_{LL}$
    Compute $\Psi_{l,j}$ by adding in the contributions from all squares in j's interaction list according to (2.17).
    if j is childless **then**
      for all boxes k in the *s-list* of j:
        evaluate the multipole expansion Φ_k at each target in square j.
      end
      Evaluate the local expansion $\Psi_{l,j}$ at each

target in square j.
endif
end
end

Cost [The upward pass requires approximately Mp^2 work, where M is the number of leaf nodes. The downward pass requires approximately $12Mp^2 + 4Mp^3$ work (see Remark 2.3 below).]

Step II: Local interactions

Comment [At this point, for each leaf node D_i, we have computed the influence of the source distribution ω over all leaf nodes D_j outside the neighbors of D_i.]

do $i = 1, \ldots, M$
For each target point in D_i, evaluate the influence of each neighbor according to (2.2) using the precomputed tables of coefficients (2.4).
end

Cost [The maximum number of neighbors a square can have is thirteen (twelve fine neighbors and itself). Thus the local work is bounded by $12 \cdot 10 \cdot N$ operations.]

REMARK 2.3. *It is somewhat difficult to determine the cost of Step* I *precisely, since it depends on the actual structure of the adaptive quad-tree. A reasonable estimate for the total work is*

$$N\left(120 + \frac{12}{16}p^2 + \frac{4}{16}p^3\right).$$

2.3. The generalized FMM. One drawback of the preceding scheme is that it relies on spherical harmonic expansions. These are particularly efficient tools for fully three-dimensional calculations, but they make no use of the fact that we are restricted to a two-dimensional domain. In fact, a two-dimensional Taylor series would have served equally well. In the last few years, methods related to the FMM have been developed which are based only on the fact that the far field is smooth [1, 2, 8, 14]. The paper [8] presents a generalized FMM based on "compressing" operators with the singular value decomposition (SVD). To understand this approach, let us reconsider the situation depicted in Figure 3.

DEFINITION 2.5. *Let $\mathcal{C}$ denote the space of real analytic charge distributions defined on a square D and let $\mathcal{P}$ denote the space of real analytic functions defined on the outer shaded region R depicted in Figure* 3. *We define the operator $\mathcal{O} : \mathcal{C} \to \mathcal{P}$ according to $\mathcal{O}(\sigma) = \Phi_\sigma$, where*

$$\Phi_\sigma(Q) = \int_D \frac{\sigma(P)}{2\pi|P - Q|}\, dP.$$

We also define the dual operator $\mathcal{S} : \mathcal{P} \to \mathcal{C}$ according to $\mathcal{S}(\sigma) = \Psi_\sigma$, where σ is a charge distribution defined on R, $P \in D$, and

$$\Psi_\sigma(P) = \frac{1}{2\pi} \int_R \frac{\sigma(Q)}{2\pi|P - Q|}\, dQ.$$

REMARK 2.4. *The spaces $\mathcal{C}$ and $\mathcal{P}$ are infinite-dimensional, but both $\sigma \in \mathcal{C}$ and $\Phi_\sigma \in \mathcal{P}$ can be approximated arbitrarily well by a finite tensor-product Legendre series*

$$\sigma(x, y) \approx \sum_{n=0}^{\mathcal{N}} \sum_{m=0}^{\mathcal{N}} \alpha_n^m P_n(x) P_m(y), \tag{2.18}$$

$$\Phi_\sigma(\xi,\eta) \approx \sum_{n=0}^{\mathcal{N}} \sum_{m=0}^{\mathcal{N}} \beta_n^m P_n(\xi) P_m(\eta), \tag{2.19}$$

where P_n is the Legendre polynomial of degree n scaled to the dimensions of D. More precisely, we suppose that there is an expansion of the form (2.19) *for each of the* 40 *squares comprising the region R.*

For $\mathcal{N}$ sufficiently large, we can approximate the operator $\mathcal{O}$ by the finite-dimensional matrix

$$\mathcal{O}_{\mathcal{N}} : \mathbb{R}^{\mathcal{N}^2} \rightarrow \mathbb{R}^{40\mathcal{N}^2},$$

mapping the coefficients $\{\alpha_n^m\}$ in the expansion of the charge density σ to the 40 sets of coefficients $\{\beta_n^m\}$ in the expansions of Φ_σ. If we let the SVD of $\mathcal{O}_{\mathcal{N}}$ be given by

$$\mathcal{O}_{\mathcal{N}} = U_{\mathcal{N}} \Sigma_{\mathcal{N}} V_{\mathcal{N}}^T,$$

then $\mathcal{O}_{\mathcal{N}}$ can be *compressed* by retaining only the first k terms in the decomposition:

$$\mathcal{O}_{\mathcal{N}} \approx \; U_{\mathcal{N}}(k)\, \Sigma_{\mathcal{N}}(k)\, V_{\mathcal{N}}(k)^T, \tag{2.20}$$

where $V_{\mathcal{N}}(k) \in \mathbb{R}^{\mathcal{N}^2 \times k}$, where $U_{\mathcal{N}}(k) \in \mathbb{R}^{40\mathcal{N}^2 \times k}$, and $\Sigma_{\mathcal{N}}(k) \in \mathbb{R}^{k \times k}$. This leaves one open question: for a given precision ϵ, how large must $\mathcal{N}$ and k be so that

$$\|\mathcal{O}(\sigma) - \; U_{\mathcal{N}}(k)\, \Sigma_{\mathcal{N}}(k)\, V_{\mathcal{N}}(k)^T\, P_{\mathcal{N}}\sigma\| < \epsilon, \tag{2.21}$$

where $P_{\mathcal{N}}$ is the operator projecting σ onto its truncated Legendre series? Similarly, $\mathcal{S}$ has an approximate SVD

$$\mathcal{S}_{\mathcal{N}} = W_{\mathcal{N}}(k) \Omega_{\mathcal{N}}(k) Y_{\mathcal{N}}(k)^T,$$

where $Y_{\mathcal{N}}(k) \in \mathbb{R}^{40\mathcal{N}^2 \times k}$, where $W_{\mathcal{N}}(k) \in \mathbb{R}^{\mathcal{N}^2 \times k}$, and $\Omega_{\mathcal{N}}(k) \in \mathbb{R}^{k \times k}$. It remains also to determine $\mathcal{N}$ and k so that

$$\|\mathcal{S}(\sigma) - \; W_{\mathcal{N}}(k)\, \Omega_{\mathcal{N}}(k)\, Y_{\mathcal{N}}(k)^T\, P_{\mathcal{N}}\sigma\| < \epsilon. \tag{2.22}$$

This is a rather complicated matter to handle analytically [8, 13] but straightforward to determine computationally. One can simply increase $\mathcal{N}$ and k until the desired level of accuracy is achieved, and we summarize the results in Table 1. The generalized FMM then proceeds as above with the following changes.

1. $\Phi_{l,j}$ is used to denote the projection of the charge density in box j at level l onto the first k right-singular vectors $V_{\mathcal{N}}(k)$ of $\mathcal{O}$ scaled to that level. $\Phi_{l,j}$ is stored as a k-vector, which we refer to as the "outgoing" coefficients. The right-singular vectors correspond, in some sense, to the multipole terms $e^{im\phi}/r^{n+1}$.
2. $\Psi_{l,j}$ and $\tilde{\Psi}_{l,j}$ are defined as before, except that we describe the field due to distant sources in terms of its projections onto the left singular vectors $W_{\mathcal{N}}(k)$ of $\mathcal{S}$. These fields are stored as k-vectors of "incoming" coefficients.
3. In Step I: *Upward pass*, if box j is childless, we compute $\Phi_{l,j} = V_{\mathcal{N}}(k)^T\sigma$. Since σ is described by ten polynomial coefficients, this requires $10k$ operations; the inner product of each of the k singular vectors with each of the ten basis functions can be precomputed and stored.

TABLE 1
Degree of Legendre expansions $\mathcal{N}$ and number of terms in the SVD k required to approximate the operators $\mathcal{O}$ and $\mathcal{S}$ to the indicated precision.

ϵ	$k(\mathcal{O})$	$\mathcal{N}(\mathcal{O})$	$k(\mathcal{S})$	$\mathcal{N}(\mathcal{S})$
10^{-3}	9	4	9	4
10^{-6}	36	8	36	8
10^{-8}	49	10	49	10
10^{-11}	144	14	144	14

4. In Step I: *Upward pass*, if box j has children, we project the "outgoing coefficients" of the four children onto the first k right singular vectors at j's level. This merging operator requires k^2 operations per square, since each matrix entry in the transformation can be precomputed and stored.
5. In Step I: *Downward pass*, where we had made use of the operator $\mathcal{T}_{LL}$, we now project the "incoming" field of the parent's Ψ expansion to form $\tilde{\Psi}_{l,j}$ using left singular vectors.
   This requires k^2 operations, since each matrix entry in the transformation can be precomputed and stored.
6. In Step I: *Downward pass*, we compute $\Psi_{l,j}$ by analogy with (2.17). We need to convert the "outgoing" coefficients for a box in the interaction list of square j to the corresponding "incoming" coefficients. Each entry in the conversion matrix is rather complicated, coupling the right-singular vectors of $\mathcal{O}$ to the left-singular vectors of $\mathcal{S}$. This matrix, however, can be precomputed and stored, so that the total cost is at most $27k^2$ operations per square.
7. In Step I: *Downward pass*, we also need to evaluate the "local expansion." This information is now encoded in the left singular vector coefficients and a naive method would require access of each left singular vector (of dimension $\mathcal{N}^2$).
   Fortunately, we can precompute the influence of each singular vector at each of the sixteen target points, so that only $16k$ operations are required at this stage.

REMARK 2.5. *A reasonable estimate for the total work of the generalized FMM is*

$$N\left(120 + \frac{27}{16}k^2 + 2k\right).$$

The first term corresponds to the local work, the second term to the expansion shifting work, and the third term to the work involved in forming and evaluating expansions at leaf nodes.

2.4. Periodic boundary conditions. The inversion formula (1.4) and the fast algorithm described above assume that the right-hand side ω is supported within a unit square. In certain applications, however, one would like to consider ω to be periodically extended to cover the entire $x - y$ plane. This requires a modest modification of the FMM, following the ideas presented in [11]. At the end of the upward pass of the algorithm, we have a net multipole expansion

$$\Phi(Q) = \sum_{n=0}^{\infty} \sum_{m=-n}^{n} M_n^m \cdot \frac{e^{im\phi}}{r^{n+1}} \tag{2.23}$$

TABLE 2

Timing results for the two FMMs for 3-digit accuracy. The time for the generalized FMM using 9 singular functions is denoted by T_{alg}, while the L_2 error of the result is denoted by E_{alg}. The time and L_2 error for the classical FMM using 8th-order spherical harmonics are denoted by T_{sh} and E_{sh}, respectively. The time for direct calculation is denoted by T_{dir}.

N	T_{alg}	T_{sh}	T_{dir}	$E_2(alg)$	$E_2(sh)$
200	0.01	–	0.02	$.11 \cdot 10^{-3}$	$.10 \cdot 10^{-3}$
400	0.02	0.10	0.08	$.14 \cdot 10^{-3}$	$.93 \cdot 10^{-4}$
800	0.08	0.22	0.34	$.19 \cdot 10^{-3}$	$.11 \cdot 10^{-3}$
1600	0.12	0.47	1.40	$.20 \cdot 10^{-3}$	$.19 \cdot 10^{-3}$
3200	0.28	1.13	5.77	$.26 \cdot 10^{-3}$	$.21 \cdot 10^{-3}$
6400	0.51	2.14	22.98	$.28 \cdot 10^{-3}$	$.26 \cdot 10^{-3}$

for the whole unit square. This is also the expansion for each periodic image of the square (with respect to its own center). If we imagine that D represents the unit square in Figure 3, then all such images except for the nearest (unshaded) neighbors are separated from D. Thus, the totality of the field they induce inside D is accurately representable as a single local expansion of the form

$$\Psi(Q) = \sum_{n=0}^{\infty} \sum_{m=-n}^{n} L_n^m \cdot r^n \, e^{im\phi}. \tag{2.24}$$

It remains only to obtain the operator mapping the coefficients $\{M_n^m\}$ to the coefficients $\{L_n^m\}$. We refer the reader to [10] for a discussion of this operator, which is based on the precomputation of certain lattice sums. The local expansion which describes the field due to all squares outside the neighbors of D can be denoted by $\Psi_{0,0}$ within the context of the FMM described above. In the rest of the algorithm, only two slight modifications are required; the interaction list and the local computations must be adjusted for boxes near the boundary to account for periodic images. This involves no significant increase in the amount of work.

3. Numerical results. The two algorithms described above have been implemented using a combination of Fortran 77 and C. All of the timings listed below correspond to calculations performed on an UltraSparc-I/167 with 128Mb RAM.

Example 1. In our first experiment, we consider the discrete N-body problem

$$\phi(\mathbf{x}_j) = \sum_{\substack{i=1 \\ i \neq j}}^{N} \frac{q_i}{2\pi \|\mathbf{x}_i - \mathbf{x}_j\|}$$

with source locations $\{\mathbf{x}_i\}$ randomly but uniformly distributed in the unit square and source strengths randomly distributed in $[-1, 1]$. We compare the performance of the two FMMs at 3-, 6-, and 12-digit accuracy (Tables 2, 3, and 4).

REMARK 3.1. *Subsequent calculations will rely on the generalized FMM, since our numerical experiments show it to be three to four times faster than the spherical harmonic-based code.*

Example 2. We consider a smooth distribution of charge:

$$f(x, y) = x \, \exp(-500x^2).$$

In Table 5, we show the results of inverting the square root of the Laplacian on a uniform $N_1 \times N_1$ grid using the generalized FMM with 24 singular functions, the

TABLE 3

Timing results for the two FMMs for 6*-digit accuracy. The generalized FMM uses* 36 *singular functions, while the classical FMM uses* 18*th-order spherical harmonics. The columns are defined as in Table* 2*.*

N	T_{alg}	T_{sh}	T_{dir}	$E_2(alg)$	$E_2(sh)$
200	0.03	–	0.02	$.77 \cdot 10^{-7}$	–
400	0.04	–	0.08	$.60 \cdot 10^{-7}$	–
800	0.19	0.67	0.34	$.72 \cdot 10^{-7}$	$.25 \cdot 10^{-7}$
1600	0.32	1.42	1.50	$.79 \cdot 10^{-7}$	$.24 \cdot 10^{-7}$
3200	1.10	3.21	6.04	$.15 \cdot 10^{-6}$	$.30 \cdot 10^{-7}$
6400	1.72	6.63	24.21	$.15 \cdot 10^{-6}$	$.33 \cdot 10^{-7}$

TABLE 4

Timing results for the two FMMs for 12*-digit accuracy. The generalized FMM uses* 144 *singular functions, while the spherical harmonic FMM uses* 32*nd-order spherical harmonics. The columns are defined as in Table* 2*.*

N	T_{alg}	T_{sh}	T_{dir}	$E_2(alg)$	$E_2(sh)$
1600	0.85	4.32	1.41	$.81 \cdot 10^{-12}$	$.37 \cdot 10^{-12}$
3200	3.36	6.43	5.69	$.98 \cdot 10^{-12}$	$.49 \cdot 10^{-12}$
6400	5.30	14.61	22.81	$.78 \cdot 10^{-12}$	$.46 \cdot 10^{-12}$

TABLE 5

Timing results for the uniform mesh of Example 2*. The first column shows the value of* N_1 *defining the grid.* T_{24} *denotes the time required by the FMM using* 24 *singular functions, and* E_{24} *denotes the* L_2 *error incurred by the method.* T_{36} *and* E_{36} *are similarly defined.* T_{FFT} *denotes the time required by the FFT, and* E_{FFT} *denotes the corresponding error.*

N_1	$T_{FMM(24)}$	$E_{FMM(24)}$	$T_{FMM(36)}$	$E_{FMM(36)}$	T_{FFT}	E_{FFT}
32	0.034	$.59 \cdot 10^{-1}$	0.057	$.59 \cdot 10^{-1}$	0.002	$.41 \cdot 10^{-3}$
64	0.132	$.21 \cdot 10^{-2}$	0.232	$.21 \cdot 10^{-2}$	0.006	$.17 \cdot 10^{-10}$
128	0.548	$.12 \cdot 10^{-3}$	0.889	$.12 \cdot 10^{-3}$	0.025	$.28 \cdot 10^{-13}$
256	1.912	$.12 \cdot 10^{-4}$	3.599	$.10 \cdot 10^{-4}$	0.133	$.28 \cdot 10^{-13}$
512	7.527	$.73 \cdot 10^{-5}$	14.240	$.10 \cdot 10^{-5}$	0.704	$.28 \cdot 10^{-13}$

TABLE 6

Performance measures for the generalized FMM. The first column shows the value of N_1 *defining the grid.* P_{24} *and* P_{36} *denote the number of grid points processed per second by the generalized FMM using* 24 *and* 36 *singular functions, respectively.* P_{FFT} *denotes the number of grid points processed per second by the FFT.*

N_1	P_{24}	P_{36}	P_{FFT}	P_{24}/P_{FFT}	P_{36}/P_{FFT}
64	31030	17655	682667	22.0	38.7
128	29898	18430	655360	21.9	35.6
256	34276	18210	492752	14.4	27.1
512	34827	18409	372364	10.7	20.2

generalized FMM with 36 singular functions, and the FFT. To compare the performance of the FMM with the FFT, we also compute the number of points processed per second by both schemes and a ratio of their timings (Table 6).

The following observations can be made from the data:

1. The generalized FMMs converge (more or less) as expected for a fourth-order accurate method. The 24 singular function and 36 singular function versions yield the same accuracy until $N_1 = 512$, at which point the 24 singular function FMM is dominated by the FMM precision rather than the error in polynomial approximation. The FFT is, of course, rapidly convergent for this right-hand side, since it is effectively smooth and periodic.

TABLE 7

Performance of the adaptive FMM using 24 singular functions for Example 3. The first column shows the tolerance in discretizing the right-hand side. The second, third, and fourth columns show the maximum number of levels, the total number of boxes, and the total number of discretization points used in the FMM at that tolerance. The fifth column shows the number of points that would be required by a uniform grid at the finest level using an FFT-based scheme. The sixth column shows the time required (secs.), and the last column shows the number of points processed per second.

Tol	$Level$	N_{boxes}	N_{pts}	N_{uni}	T_{alg}	P_{24}
10^{-3}	7	889	10672	(262144)	0.328	32537
10^{-4}	7	1489	17872	(262144)	0.523	34172
10^{-5}	8	3265	39184	(1048576)	1.128	34736
10^{-6}	9	5369	64432	(4194304)	1.868	34493
10^{-7}	9	12977	155728	(4194304)	4.414	35280

2. The timings for the FMMs grow linearly with the number of unknowns, while the timings for the FFT grow like $N_1^2 \log N_1$. By the time there are 250,000 unknowns, the 5-digit FMM (24 singular functions) is about 10 times as costly as the FFT. The 7-digit FMM (36 singular functions) is about 20 times as costly.

Example 3. We next consider a more complex distribution of charge

$$f(x,y) = g(x-0.25, y-0.25) + g(x+.15, y-.15) + g(x-.05, y+.25),$$

where

$$g(x,y) = xy\,\exp(-2000x^2)\,\exp(-2000y^2).$$

There are three sharply peaked contributions to the total charge, and the right-hand side is discretized adaptively. Our refinement strategy is straightforward. Let B be a leaf node with 16 grid points, as discussed in section 2.1, and let $f_B(x,y)$ denote the fourth-order polynomial used to approximate the right-hand side on B. We then evaluate $f_B(x,y)$ on an 8×8 grid covering B and compute the discrete L_2 error $E_2 = \|f(x,y) - f_B(x,y)\|_2$ over these target points. If $E_2 > tol$, the leaf node B is subdivided. Our results are summarized in Table 7.

Example 4. In our last example, we consider a discontinuous right-hand side

$$f(x,y) = \begin{cases} 3 & |x+y| > 1/2, \\ -1 & \text{otherwise.} \end{cases}$$

Figure 4 shows the function $f(x,y)$ and the adaptive mesh generated by the discretization technique outlined in Example 3. Figure 5 shows the computed solution. Timing results are summarized in Table 8.

4. Conclusions. We have developed an adaptive direct solver for inverting the square root of the Laplacian in two dimensions. While our first implementation relied on a classical spherical-harmonic-based FMM, a faster scheme uses a generalized FMM [8], which constructs optimal representations for the far field using an SVD. The method can be used in free space or with periodic boundary conditions, and the amount of work scales linearly with the number of grid points in the computational domain. The method is an order of magnitude slower than an FFT-based scheme on uniform grids, but quickly surpasses such schemes once adaptive refinement is required. Applications of the method to some problems in fluid dynamics will be reported at a later date.

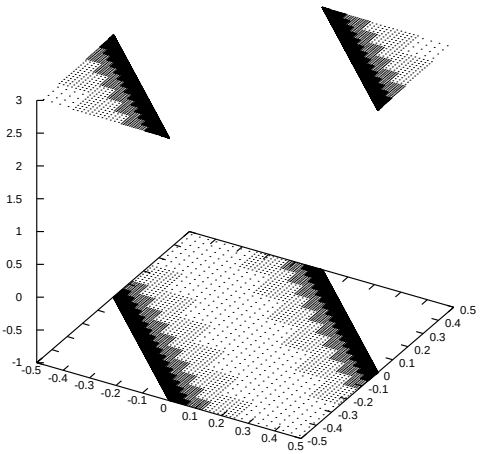


FIG. 4. *The discontinuous right-hand side f for Example* 4.

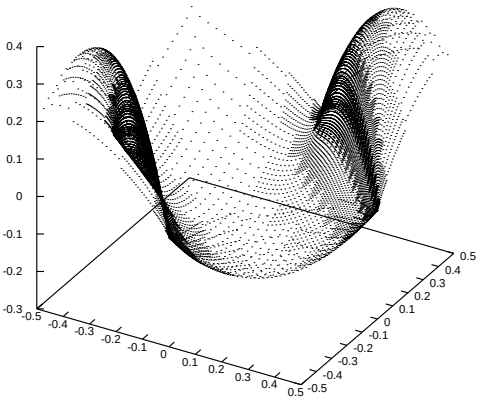


FIG. 5. *The potential induced by inverting the square root of the Laplacian using the right-hand side f in Figure* 4.

TABLE 8
Performance of the adaptive FMM using 24 *singular functions for Example* 4. *The columns are defined as in Table* 7.

Tol	$Level$	N_{boxes}	N_{pts}	N_{uni}	T_{alg}	P_{24}
10^{-3}	7	1205	14464	(262144)	0.434	33327
10^{-4}	9	5045	60544	(4194304)	1.725	35098
10^{-5}	10	10165	121984	(16777216)	3.473	35124

REFERENCES

[1] G. Beylkin, R. Coifman, and V. Rokhlin, *Fast wavelet transforms and numerical algorithms* I, Comm. Pure Appl. Math., 44 (1991), pp. 141–183.

[2] A. Brandt and A. A. Lubrecht, *Multilevel matrix multiplication and fast solution of integral equations*, J. Comput. Phys., 90 (1990) pp. 348–370.

[3] W. Choi and R. Camassa, *Weakly nonlinear internal waves in a two-fluid system*, J. Fluid Mech., 313 (1996), pp. 83–103.

[4] J. Carrier, L. Greengard, and V. Rokhlin, *A fast adaptive multipole algorithm for particle simulations*, SIAM J. Sci. Statist. Comput., 9 (1988), pp. 669–686.

[5] B. Cichocki and B. U. Felderhof, *Electrostatic interactions in thin-film Coulomb systems with periodic boundary conditions*, Molecular Phys., 67 (1989), pp. 1373–1384.

[6] P. Constantin, A. J. Majda, and E. Tabak, *Formation of strong fronts in the* 2*-D quasi-geostrophic thermal active scalar*, Nonlinearity, 7 (1994), pp. 1495–1533.

[7] A. Faridani, D. V. Finch, E. L. Ritman, and K. T. Smith, *Local Tomography.* II, SIAM J. Appl. Math., 57 (1997), pp. 1095–1127.

[8] Z. Gimbutas, *A Generalized Fast Multipole Method for Non-Oscillatory Kernels*, Ph.D. dissertation, Yale University, New Haven, CT, 1999.

[9] L. Greengard, *The Rapid Evaluation of Potential Fields in Particle Systems*, MIT Press, Cambridge, MA, 1988.

[10] C. L. Berman and L. Greengard, *A renormalization method for the evaluation of lattice sums*, J. Math. Phys., 35 (1994), pp. 6036–6048.

[11] L. Greengard and V. Rokhlin, *A fast algorithm for particle simulations*, J. Comput. Phys., 73 (1987), pp. 325–348.

[12] L. Greengard and V. Rokhlin, *A new version of the fast multipole method for the Laplace equation in three dimensions*, Acta Numer., 6 (1997), pp. 229–269.

[13] T. Hrycak and V. Rokhlin, *An improved fast multipole algorithm for potential fields*, SIAM J. Sci. Comput., 19 (1998), pp. 1804–1826.

[14] S. Kapur and D. E. Long, *IES*3*: Efficient electrostatic and electromagnetic simulation*, IEEE Comput. Sci. Eng., 5 (1998), pp. 60–67.

[15] J. D. Jackson, *Classical Electrodynamics*, Wiley, New York, 1975.

[16] J. R. Phillips and J. K. White, *A precorrected-FFT method for electrostatic analysis of complicated* 3*-D structures*, IEEE Trans. Comput. Aid. D., 16 (1997), pp. 1059–1072.

[17] N. R. Smith, *Ocean modeling in a global ocean observing system*, Reviews of Geophysics, 31 (1993), pp. 281–317.

[18] P. R. Wallace, *Mathematical Analysis of Physical Problems*, Dover, New York, 1984.

[19] H. Weber, M. Wallin, and H. J. Jensen, *Monte Carlo calculation of the current-voltage characteristics of a two-dimensional lattice Coulomb gas*, Phys. Rev. B, 53 (1996), pp. 8566–8574.

[20] A. H. Widmann and D. B. Adolf, *A comparison of Ewald summation techniques for planar surfaces*, Comput. Phys. Comm., 107 (1997), pp. 167–186.

SIAM J. SCI. COMPUT.
Vol. 22, No. 6, pp. 2109–2112
© 2001 Society for Industrial and Applied Mathematics

CGNR IS AN ERROR REDUCING ALGORITHM*

CHUNGUANG LI†

Abstract. The CGNR algorithm is a robust algorithm solving large nonsymmetric linear systems. In this short note, we show that the norm of the error vector of CGNR approximation decreases monotonically during its iteration. Few iterative algorithms have this error reducing property. The result indicates that the simple CGNR algorithm is still a potential one in practical use.

Key words. linear systems, CGNR algorithm, norm, residual vector, error vector

AMS subject classification. 65F10

PII. S1064827500375990

1. Introduction.

Consider large linear system

$$Ax = b, \tag{1.1}$$

where A and b are given, A is a nonsingular $n \times n$ real matrix, and b is an n-vector. The CGNR algorithm is the conjugate gradients (CG) method applied to the equivalent system–normal equations–

$$A^T Ax = A^T b, \tag{1.2}$$

which is an s.p.d. system. The CGNR algorithm can be described as follows [6].

ALGORITHM 1.1 (CGNR algorithm).

1. *Compute* $r_0 = b - Ax_0, z_0 = A^T r_0, p_0 = z_0$;
2. *For* $i = 0, 1, \ldots,$ *until convergence Do*
3. $w_i = Ap_i$;
4. $\alpha_i = \|z_i\|_2^2 / \|w_i\|_2^2$;
5. $x_{i+1} = x_i + \alpha_i p_i$;
6. $r_{i+1} = r_i - \alpha_i w_i$;
7. $z_{i+1} = A^T r_{i+1}$;
8. $\beta_i = \|z_{i+1}\|_2^2 / \|z_i\|_2^2$;
9. $p_{i+1} = z_{i+1} + \beta_i p_i$;
10. *EndDo*

This algorithm is quite robust since the CG algorithm is robust for an s.p.d. linear system. The main difficulty in using the CGNR algorithm is that the condition number is squared, i.e., $\kappa(A^T A) = \kappa^2(A)$.

For a long time, the CGNR algorithm has had a bad reputation because of the squaring of the condition number. However, in our numerical tests we often see that the CGNR algorithm is a successful method in many difficult situations. The fact was also observed by other authors; see Saad [5], Saylor [7], and Nachtigal [4]. By a careful analysis, we can find that the CGNR algorithm is not only a residual minimizing method but also an error reducing method. Few algorithms among the Krylov subspace iterative methods have this useful property.

*Received by the editors August 1, 2000; accepted for publication (in revised form) December 18, 2000; published electronically April 12, 2001. This work was partly supported by the Youth Science Foundation of Ningxia, People's Republic of China.

http://www.siam.org/journals/sisc/22-6/37599.html

†Department of Mathematics, Zhengzhou University, Daxue Road No. 75, Zhengzhou, Henan, 450052, People's Republic of China (cglizd@public2.zz.ha.cn).

The error reducing property of the CGNR algorithm seems to be not generally known. Our major aim here is to prove this property strictly.

2. Main results. The CGNR algorithm is mathematically equivalent to the CG method applied to the s.p.d. linear system (1.2). It has the following optimal property concerning the residuals.

PROPOSITION 2.1. *Let x_k be the approximation produced at the k*th *step by the CGNR algorithm (k*th *iterate). Then x_k minimizes function $\|b-Ax\|_2^2$ over all vectors in the affine Krylov subspace*

$$x_0 + \mathcal{K}_k(A^TA, A^Tr_0) = x_0 + \ span\ \left\{A^Tr_0, A^TAA^Tr_0, \ldots, (A^TA)^{k-1}A^Tr_0\right\}, \tag{2.1}$$

in which $r_0 = b-Ax_0$ is the initial residual with respect to the original system $Ax = b$.

This property is a direct consequence of the optimal property of the CG method. Its proof can be found in Saad [6, p. 238].

PROPOSITION 2.2. *Let x^* be the exact solution of the system $Ax = b$, and let $e_k = x^* - x_k$ be the error vector of the CGNR iterate x_k. Then the 2-norm of e_k decreases monotonically, i.e.,*

$$\|e_{k+1}\|_2 \le \|e_k\|_2, k = 0, 1, \ldots. \tag{2.2}$$

This proposition exhibits that the CGNR algorithm has an error reducing property. We give its proof here. At first, we establish two lemmas.

LEMMA 2.3. *The residuals with respect to* (1.2), $z_0, z_1, \ldots, z_k, \ldots$, *are orthogonal to each other, i.e.,*

$$z_i^H z_j = \begin{cases} 0, & i \ne j, \\ \|z_i\|_2^2, & i = j, \end{cases} \tag{2.3}$$

in which $z_i = A^Tb - A^TAx_i$ is the residual vector with respect to the linear system (1.2).

This is a generally known result of the CG method for the s.p.d. linear system. Its proof can be found in any text book concerning the CG method; see for example, Saad [6, p. 178].

LEMMA 2.4. *The direction vector p_k in the CGNR algorithm is a linear combination of the residuals $z_0, z_1, \ldots, z_k$,*

$$p_k = \xi_k \left(\frac{z_0}{\xi_0} + \frac{z_1}{\xi_1} + \cdots + \frac{z_k}{\xi_k}\right), k = 0, 1 \ldots, \tag{2.4}$$

where

$$\xi_j = \|z_j\|_2^2,\ z_j = A^Tb - A^TAx_j,\ j = 0, 1, \ldots, k.$$

Proof. From lines 9 and 8 of the CGNR algorithm, we have

$$p_{i+1} = z_{i+1} + \beta_i p_i, \quad \beta_i = \|z_{i+1}\|_2^2/\|z_i\|_2^2.$$

So, we get

$$\begin{aligned} p_{i+1} = &\ z_{i+1} + \beta_i p_i \\ = &\ \xi_{i+1}\left(\frac{p_i}{\xi_i} + \frac{z_{i+1}}{\xi_{i+1}}\right). \end{aligned}$$

We see that

$$\begin{aligned} p_0 &= z_0, \\ p_1 &= \xi_1 \left(\frac{p_0}{\xi_0} + \frac{z_1}{\xi_1} \right) = \xi_1 \left(\frac{z_0}{\xi_0} + \frac{z_1}{\xi_1} \right), \\ p_2 &= \xi_2 \left(\frac{p_1}{\xi_1} + \frac{z_2}{\xi_2} \right) = \xi_2 \left(\frac{z_0}{\xi_0} + \frac{z_1}{\xi_1} + \frac{z_2}{\xi_2} \right). \end{aligned}$$

In general, we have

$$p_k = \xi_k \left(\frac{p_{k-1}}{\xi_{k-1}} + \frac{z_k}{\xi_k} \right) = \xi_k \left(\frac{z_0}{\xi_0} + \frac{z_1}{\xi_1} + \cdots + \frac{z_{k-1}}{\xi_{k-1}} + \frac{z_k}{\xi_k} \right), k = 0, 1 \ldots .$$

This is just (2.4). □

Now, we turn to prove Proposition 2.2.

Proof of Proposition 2.2. From the CGNR algorithm, line 5 of Algorithm 1.1, we have

$$\begin{aligned} \|e_k\|_2^2 &= \|x^* - x_k\|_2^2 \\ &= \|x^* - x_{k+1} + \alpha_k p_k\|_2^2 \\ &= \|x^* - x_{k+1}\|_2^2 + 2\alpha_k p_k^H (x^* - x_{k+1}) + \|\alpha_k p_k\|_2^2 \\ &= \|e_{k+1}\|_2^2 + 2\alpha_k p_k^H (x^* - x_{k+1}) + \|\alpha_k p_k\|_2^2. \end{aligned}$$

Inequality (2.2) holds if we show that

$$p_k^H (x^* - x_{k+1}) \geq 0, \quad k = 0, 1, \ldots . \tag{2.5}$$

It is well known that the CGNR algorithm will be terminated at most n steps for the s.p.d. linear system (1.2) in exact arithmetic (in the absence of round-off errors), where n is the dimension of the linear system. So, we have $x^* = x_n$. The inequality (2.5) holds when $k = n - 1$. If $k < n - 1$, from the update of x_k in the CGNR algorithm, we have

$$x^* = x_n = x_{k+1} + \alpha_{k+1} p_{k+1} + \alpha_{k+2} p_{k+2} + \cdots + \alpha_{n-1} p_{n-1}.$$

Consequently,

$$p_k^H (x^* - x_{k+1}) = \alpha_{k+1} p_k^H p_{k+1} + \cdots + \alpha_{n-1} p_k^H p_{n-1}. \tag{2.6}$$

By virtue of Lemmas 2.3 and 2.4, for $j = k+1, k+2, \ldots, n-1$, we have

$$\begin{aligned} p_k^H p_j = \;& \xi_k \xi_j \left(\frac{z_0}{\xi_0} + \cdots + \frac{z_k}{\xi_k} \right)^H \left(\frac{z_0}{\xi_0} + \cdots + \frac{z_j}{\xi_j} \right) \\ = \;& \xi_k \xi_j \left(\frac{1}{\xi_0} + \cdots + \frac{1}{\xi_k} \right) \geq 0. \end{aligned}$$

In (2.6), $\alpha_j, j = k+1, k+2, \ldots, n-1$ are nonnegative real numbers. Now, we get to know that the right-hand side of (2.6) is nonnegative. This establishes inequality (2.5) and hence inequality (2.2). □

Note. Proposition 2.2 had been pointed out by Ehrig and Deuflhard in an unpublished report [2]. However, their proof given in [2] is too technical to be understood. The new proof here is likely more clear and direct.

Combining the above two propositions, we get the following important theorem.

THEOREM 2.5. *CGNR is both residual minimizing and error reducing.*

3. Further remarks. In the Krylov subspace iterative methods, the relationships between the norm of the residual vector and the norm of the error vector are

$$\|r_k\| = \|Ae_k\| \leq \|A\| \cdot \|e_k\|,$$

and

$$\|e_k\| = \|A^{-1}r_k\| \leq \|A^{-1}\| \cdot \|r_k\|.$$

However, curves of the error norm and the residual norm are not always in the same tendency. Examples were found when the norm of residual was still decreasing but the norm of error was stagnating. See Bruaset [1, p. 63, Example 3.2] and Weiss [8, Example 1]. The generalized minimal error (GMERR) algorithm was introduced by Weiss [8] in order to find the "optimal" approximation with the minimal norm of the error in the corresponding affine Krylov subspace. However, the implementation of GMERR is quite complicated in computing. So, at this point, Theorem 2.5 indicates that the simple CGNR algorithm is a potential method in practical use.

We would like to mention that even if the CGNR algorithm has the error reducing property, the residual norm should be still used for monitoring the convergence in practical computing since we cannot estimate the error norm during its iteration process.

At last, we mention that the condition number of the normal equations (1.2) could be reduced by using preconditioning techniques. Limited numerical test results using the polynomial preconditioning techniques were reported in Li [3]. Further numerical tests using the other preconditioners, which might have better performance, are being undertaken.

Acknowledgments. The author is grateful to Professors Chengxian Xu and Wenyu Sun for their kindly instructions and useful discussions.

REFERENCES

[1] A. M. Bruaset, *A Survey of Preconditioned Iterative Methods*, Pitman Res. Notes Math. Ser. 328, Longman, Harlow, UK, 1995.
[2] R. Ehrig and P. Deuflhard, *GMERR—An Error Minimizing Variant of GMRES*, Technical report, Konrad-Zuse-Zentrum für Informationstechnik Berlin, Berlin, 1997, http://www.zib.de/.
[3] C. Li, *Some Researches on the Iterative Methods for Solving Large Sparse Linear Systems*, Ph.D. thesis, Xi'an Jiaotong University, Xi'an, Shaanxi province, People's Republic of China, 1999.
[4] N. M. Nachtigal, S. C. Reddy, and L. N. Trefethen, *How fast are nonsymmetric matrix iterations?*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 778–795.
[5] Y. Saad, *Preconditioning techniques for nonsymmetric and indefinite linear systems*, J. Comput. Appl. Math., 24 (1988), pp. 89–105.
[6] Y. Saad, *Iterative Methods for Sparse Linear Systems*, PWS Publishing, Boston, 1996.
[7] P. E. Saylor and D. C. Smolarski, *Implementation of an adaptive algorithm for Richardson's method*, Linear Algebra Appl., 154–156 (1991), pp. 615–646.
[8] R. Weiss, *Error-minimizing Krylov subspace methods*, SIAM J. Sci. Comput., 15 (1994), pp. 511–527.

SIAM J. SCI. COMPUT.
Vol. 22, No. 6, pp. 2113–2134
© 2001 Society for Industrial and Applied Mathematics

FAST BIT-REVERSALS ON UNIPROCESSORS AND SHARED-MEMORY MULTIPROCESSORS*

ZHAO ZHANG† AND XIAODONG ZHANG†

Abstract. In this paper, we examine different methods using techniques of blocking, buffering, and padding for efficient implementations of bit-reversals. We evaluate the merits and limits of each technique and its application and architecture-dependent conditions for developing cache-optimal methods. Besides testing the methods on different uniprocessors, we conducted both simulation and measurements on two commercial symmetric multiprocessors (SMP) to provide architectural insights into the methods and their implementations. We present two contributions in this paper: (1) Our integrated blocking methods, which match cache associativity and translation-lookaside buffer (TLB) cache size and which fully use the available registers, are cache-optimal and fast. (2) We show that our padding methods outperform other software-oriented methods, and we believe they are the fastest in terms of minimizing both CPU and memory access cycles. Since the padding methods are almost independent of hardware, they could be widely used on many uniprocessor workstations and multiprocessors.

Key words. cache optimizations, memory hierarchy, bit-reversals, shared-memory multiprocessors, parallel computing

AMS subject classifications. 68P05, 65Y20, 65Y05

PII. S1064827599359709

1. Introduction.

Many FFT algorithms require data reordering operations of *bit-reversal*. If the bit-reversal operations are not implemented properly, those FFT operations can slow down significantly. On the other hand, it is easy to improperly implement bit-reversals on uniprocessors and multiprocessors. This is because the performance of bit-reversals is highly sensitive to how caches and memory hierarchies are used in the implementations. In other words, a fast bit-reversal implementation must be cache effective. Several papers have well addressed the significance and effects of considering memory hierarchy to bit-reversals (e.g., [2], [11], and [15]). Besides the important usage for FFT, different versions of bit-reversal implementations can also be used as benchmark programs to evaluate the memory hierarchy of various computer systems.

With the rapid development of RISC and VLSI technology, the speed of processors has increased dramatically in the past decade. Processor clock rates have doubled every 1–2 years. Nevertheless, the memory speed has increased at a much slower pace. Therefore we have seen and will continue to see an increasing gap in speed between processor and memory, and this gap makes performance of application programs on both uniprocessor and multiprocessor systems rely more and more on effective usage of caches. Performance degradation of bit-reversals is mainly caused by cache conflict misses. Bit-reversals are often repeatedly used as fundamental subroutines for scientific programs, such as FFT. Thus, in order to gain the best performance, cache-

*Received by the editors September 17, 1999; accepted for publication (in revised form) November 20, 2000; published electronically April 12, 2001. This work is supported in part by the National Science Foundation under grants CCR-9400719 and CCR-9812187, by the Air Force Office of Scientific Research under grant AFOSR-95-1-0215, and by Sun Microsystems under grant EDUE-NAFO-980405. Preliminary results of this work were presented in the 1999 Supercomputing Conference, Portland, OR.

http://www.siam.org/journals/sisc/22-6/35970.html

†Department of Computer Science, College of William and Mary, Williamsburg, VA 23187-8795 (zzhang@cs.wm.edu, zhang@cs.wm.edu).

optimal methods and their implementations should be carefully and precisely done at the programming level. This type of performance programming for some special programs, such as bit-reversals, may significantly outperform an optimization from an automatic tool, such as a compiler.

A standard bit-reversal program is described as follows:

```
for i = 1, N
    Y[i'] = X[i]
```

The values of array X in their sequential positions i are copied to array Y in their bit-reversal positions, i' for $i = 1, \ldots, N$, where $N = 2^n$. The above program says that X is a bit-reversal reordering of Y. The indices of i and i' of X and Y are represented by a sequence of n binary digits. Positions i and its bit-reversal i' are defined in [11] as

$$i = \sum_{j=0}^{n-1} a_j 2^j \quad \text{and} \quad i' = \sum_{j=0}^{n-1} a_j 2^{n-1-j},$$

where a_j is either 0 or 1. For example, a 5-bit reversal of $i = 10010$ is $i' = 01001$.

The bit-reversal operations have following unique characteristics: First, in many implementations, each element in an array is used (read or written) only once for its copy operation. Thus, the reorderings have only spatial locality but no temporal locality for elements. Second, the loops follow certain sequences with high spatial locality. Bit-reversals are highly sensitive to problem sizes, cache sizes, and cache line sizes. Since the data array sizes are a power of 2, multiple elements stored in different memory locations could map to the same cache line, causing severe cache conflict misses and cache thrashing. The reason is simple. Most commercial computers use direct-mapped or n-way associative caches where the mapping functions of cache sizes are also related to powers of 2.

We use an identical unit, called an "element," to represent the sizes of data arrays, caches, and others such as buffers and blocking. One element may represent a 4-byte integer, a 4-byte floating point number, or an 8-byte double floating point number. Because the sizes of caches and cache lines are always a multiple of an element in practice, this identical unit for all sizes is practically meaningful for both architects and application programmers and makes the discussions straightforward. Here are the algorithmic and architectural parameters we will use to describe cache-optimal methods of bit-reversals.

- C: data cache size, which could be further defined as C_{L1} and C_{L2} for data cache sizes of L1 and L2, respectively.
- L: the size of a cache line, which could be further defined as L_{L1} and L_{L2} for cache lines of L1 and L2, respectively.
- K: cache associativity, which could be further defined as K_{L1} and K_{L2} for cache associativity of L1 and L2, respectively.
- K_{TLB}: translation-lookaside buffer (TLB) cache associativity. (A TLB cache is a small buffer that holds most recent memory page mappings. The concept will be discussed in detail later in the paper.)
- T_s: number of entries in the TLB cache.
- N: the data size for the bit-reversal vector of size $N = 2^n$, where n is the number bits used in the vector index.
- B_{cache}: blocking size of a $B \times B$ submatrix for cache.
- B_{TLB}: blocking size for TLB.
- P_s: a memory page size.

In this paper, we examine different methods using techniques of blocking, buffering, and padding for efficient implementations. We evaluate the merits and limits of each technique and its application and architecture-dependent conditions for developing cache-optimal methods. Although our methods are developed for out-of-place bit-reversals, they are also applicable to in-place bit-reversals where X and Y are the same array.

Symmetric multiprocessor (SMP) systems have become practical and cost-effective servers for scientific computing and other applications. Although parallel efficiency and communication latency reduction are major performance concerns, computations on an SMP share many common considerations with uniprocessors. The most important one is the effective usage of memory hierarchies. When the cache locality of each processor is effectively exploited, the memory accesses to the shared-memory will be reduced, and so will be the memory access contention. People have studied parallel data reordering algorithms on distributed-memory systems with special networks, such as hypercubes (see, e.g., [6] and [9]). In this study, we target parallel bit-reversals on SMPs and show the significant impact of the cache and TLB considerations for efficient method development and implementations. We also evaluate the performance impact of SMP interconnection networks.

Our algorithm designs and implementations are optimized by considering several nontraditional but practical and performance-effective factors, namely, the programming complexity, memory space requirement, instruction count, cross interference among the data arrays, and program portability. We will summarize the limits and merits of different bit-reversal methods based on these considerations after we have discussed the designs and presented the performance results, aiming at providing a guideline for performance programming and memory performance optimization for other scientific computing applications.

We present two contributions in this paper: (1) Our integrated blocking methods, which match cache associativity and TLB cache size and which fully use the available registers, are cache-optimal and fast. (2) We show that our padding methods outperform other software-oriented methods and believe they are the fastest in terms of minimizing both CPU and memory access cycles. Since the padding methods are almost independent of hardware, they could be widely used on many uniprocessor workstations and SMP multiprocessors.

The rest of the paper is organized as follows. We discuss the inherently blocking nature of bit-reverse operations and the effectiveness and limits of blocking techniques for solving the problems in section 2. In section 3, we evaluate a software buffering technique and our methods using existing hardware components for implementing the data reordering. Our new method integrating blocking and padding will be presented in section 4. We discuss blocking and padding techniques for TLB in section 5. The experimental measurements and analyses for evaluating different methods on uniprocessor workstations and SMP multiprocessors will be reported in sections 6 and 7. We summarize the work in section 8.

2. Blocking for bit-reversals. The blocked memory access patterns of bit-reversals can be easily viewed when we convert the one-dimensional vector to a two-dimensional equivalent array in Figure 1. All the reordering elements and elements in other groups will be allocated along the column in the two-dimensional equivalent array forming a block.

In this blocking method, the bit-reversal reordering is performed block by block, where the operations for each block are implemented similarly to the Evans method

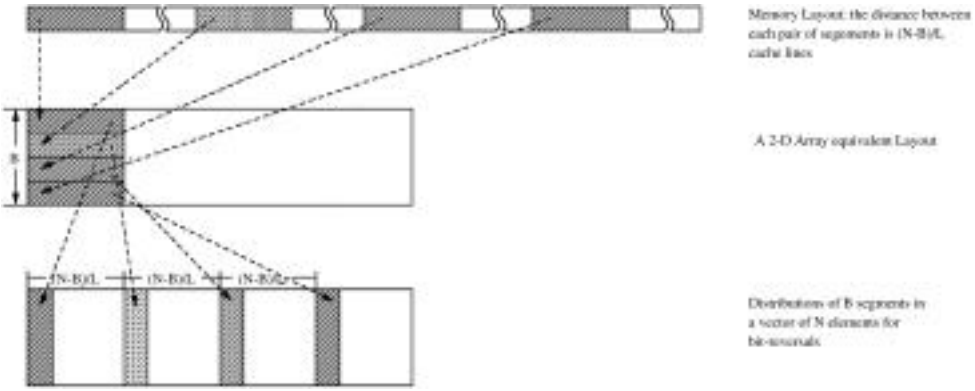


FIG. 1. *Memory layout of a blocked bit-reversals, where* $B = B_{cache}$.

[7]. (The Evans method is used to construct a hybrid method in [11].) The program in the appendix presents such an implementation along with padding technique. (The padding technique will be discussed in section 4.) The blocking algorithm we have used can be classified as a hybrid method.

In general, for a bit-reversal vector of $N = 2^n$ elements, the block size B_{cache} is a power of 2, denoted by $B_{cache} = 2^b$. Each of the B_{cache} elements in X has the address format of fg, where g is B_{cache} bits and f has $n - b$ bits. Each of the corresponding B_{cache} elements in Y has the address format of $g'f'$. Therefore, the distance between two nearest elements in the same group in Y is $2^{n-b} = N/B_{cache}$.

Choosing the cache line size as the minimum blocking size ($B_{cache} = L$), we can easily calculate the maximum N's for the bit-reversal vector based on different data cache sizes. For example, for a large cache of 2 MB, the blocking technique is effective up to an 18-bit-reversal reordering which represents 268,144 data elements, where each element is an 8-byte double type, and the cache line is 32 bytes. In practice, the data size of bit-reversals could easily be larger than $n = 20$ [11].

3. Blocking with buffers. As we have shown, the effectiveness of blocking is limited by the size of the data arrays. In theory, the smallest blocking size could be 2×2. A cache line in a modern processor usually holds more than 2 elements, i.e., is larger than 16 bytes. If we choose a 2×2 block, the data in a cache line will not be fully used before their replacement, causing more cache misses in the reorderings. The bit-reversal reordering demands large cache space to make blocking effective. In order to effectively use limited cache space, Gatlin and Carter [8] present an effective method using an additional buffer to first hold the conflict-missed elements of a block in one array temporarily and then copy the block to their reordered positions in the other array. In this section, we discuss implementations of blocking methods supported by both software and hardware buffers.

3.1. Blocking with a software buffer and its limits. Because this buffer is defined in a reordering program, we call it "software buffer." This buffer shares the allocation space with the data arrays X and Y in the cache.

There are two major limits in this approach. First, the buffer itself may interfere with arrays of X and Y, causing additional access conflicts. This interference is certain when the sizes of X and Y are larger than the size of the cache, C. Each cache block or set is mapped from arrays X and Y more than once. No matter where the buffer is

located in the cache, it will interfere with them. The larger the buffer size, the more interference will occur.

The second limit is the additional copy overhead time involved in moving data from the array X to the buffer and then in moving them to the target array in their reordered positions. This overhead exactly doubles the instruction cycles for data copying. The data copy through a buffer is a worthy investment if the number of cycles lost from cache misses is much higher than the additional CPU cycles for the data copy.

To overcome the two limits, we propose several alternatives to eliminate cache interference caused by the software and to reduce or eliminate the data copy time.

3.2. Cache structure dependent blocking. We will present several blocking methods which depend on the cache organization of the running machine. These methods can be implemented at the user programming level.

Blocking based on set associativity. The cache associativity, K, is an important factor to consider for blocking. If $K \geq L$, an $L \times L$ or a $K \times K$ blocking method for bit-reversals would effectively avoid conflict misses. Because the hit time is a less sensitive performance factor than the cache misses in the L2 cache, a higher associativity of the L2 cache is more effective than that of L1. If a cache line holds 4 double floating point elements ($L = 4$ elements of 32 bytes in Pentium processors), a 4×4 blocking method without any data buffer is able to fully use the cache associativity. The blocking method would gain more benefit from caches of associativity higher than 4, such as a design in [20].

What would we do if the associativity is not sufficiently high for the blocking, or $K < L$? One solution is to make a $K \times L$ rectangular blocking. Unfortunately bit-reversals require an $L \times L$ blocking.

Supplement with registers. We may also consider using the available registers to supplement a low associativity cache. The number of registers available to a user program is limited. Normally, a uniprocessor provides up to 16 registers to users. For example, for a 2-way associative cache, we need 8 registers to buffer 2 additional cache lines so that we could effectively make a 4×4 blocking as if we ran the program on a 4-way associative cache.

We develop a more efficient blocking method for bit-reversals, which requires only $(L-K) \times (L-K)$ registers. The operation sequence of this method is in three steps: (1) The $L - K$ cache lines of X are stored in K cache lines of Y and accessed by copying its $(L - K) \times K$ elements to Y in the reordered positions and copying the rest of $(L-K) \times (L-K)$ elements to a buffer consisting $(L-K) \times (L-K)$ registers. (2) The rest of K lines of X are brought to the cache set, and its $K \times K$ elements are copied to Y in the reordered positions. (3) Finally, the $(L - K) \times (L - K)$ elements in the register buffer and the rest of the $(L - K) \times K$ elements are copied to Y in their reordered positions. A cache set will be used more than twice if $K < L/2$.

Besides the advantage of no access conflicts between the register buffer and the arrays of X and Y, there is another advantage of using registers to buffer the data in a load/store processor. A data copy through the registers from X to Y is equivalent to the two-step process of load and store, and thus there will be no additional overhead. We will show our experimental performance in section 5.

Using registers as the buffer. If the cache is direct-mapped, we have to fully rely on a buffer for blocking. Here we discuss some ways to use registers to serve the buffer in order to eliminate the potential cache conflicts and eliminate extra data

copying by taking advantage of the load/store operations. The number of registers for a buffer of $L \times L$ elements is determined by the number of elements a cache line can hold. The length of a cache line of the L1 cache in some processors, such as Sun SPARC Micro I and II, is $L = 2$ of 16 bytes, which holds only two floating point elements. The blocking size could be as small as 2×2 using a buffer of 4 registers.

The cache line length of the L1 cache in many advanced workstations is 32 bytes, such as the Sun Ultra and Intel Pentium processors, each of which holds 4 double floating point elements. In this case, we need a buffer of $4 \times 4 = 16$ registers for a blocking. This would be difficult due to the limited number of available registers. We have two solutions for this. First, we use only the number of registers available to form a smaller buffer than it should be, which will not make each cache line fully used and will cause additional cache misses. Our experiments show that this blocking method of using a buffer of insufficient number of registers still achieves a reasonable performance improvement and outperforms the implementation using a software buffer.

The second method is to further reduce the size of the buffer, which reduces the required number of registers by using our $(L - K) \times (L - K)$ blocking method.

L1 cache versus L2 cache. The main objective of building two-level caches is to make the L1 cache small enough to catch up to the cycle time of the fast CPU and to make the L2 cache large enough to capture as many accesses as possible [12]. In practice, the data size of a bit-reversal is larger than the size of the L2 cache. L1 and L2 caches offer different sizes of the cache line, L, and the associativity, K. Both of the following alternatives are effective for blocking. (1) Taking advantage of a short cache line and fast hit time of the L1 cache, we could effectively use limited registers as the buffer and make a small $L \times L$ blocking effective. (2) Taking advantage of high associativity of the L2 cache, we could effectively use both associativity and supplemental registers as the buffer and make a large $L \times L$ blocking effective.

3.3. Victim-cache-aided blocking. Victim cache [13] is a small fully associative cache serving as the buffer containing only cache blocks due to conflict misses from L1 cache. This is an on-chip cache connected between L1 and the next level cache or memory. On a miss in L1, the victim cache is first checked before going to the next level. If the missed block is found there, the victim cache block and the L1 cache block are swapped and then the block is delivered to CPU from the L1 cache. Victim cache has been available in some commercial workstations, such as HP7200.

The minimum number of victim cache lines required for $L \times L$ blockings of transpose and bit-reversal reorderings is $L - K$. In the execution, $L \times L$ elements of each blocking are allocated in a set of K lines in L1 cache, and the rest of the elements are allocated in the $L - K$ lines of the victim cache. The victim cache is able to hold all the conflict misses in the reorderings by an $L \times L$ blocking. In addition, a conflict miss in the L1 cache that hits in the victim cache has only one additional cycle miss penalty. Thus, a simple $L \times L$ blocking method would be effective if such a victim cache is available.

However, the victim cache does not have a direct connection with the CPU. When a data hit happens in the victim cache, it has to be first swapped to the L1 cache and then delivered to CPU. This swapping operation is unnecessary for our reordering algorithms. Without counting the cold misses of bringing the elements in the first column for an $L \times L$ blocking and considering the LRU replacement policy, the entire blocking will have $L \times (L - 1)$ conflict misses in the L1 cache, which are then found in the victim cache. This also means that each of such a blocking needs $L \times (L - 1)$ additional swapping cycles between the L1 cache and the victim

cache, which is independent of the associativity, K. In contrast with the blocking method based on the associativity supplemented by registers, the swapping cycles in the victim cache are additional overhead. Despite this, a victim-cache-aided blocking is more efficient than a blocking method with a software buffer because there are no cross interference conflicts between the victim buffer and arrays of X and Y.

4. Blocking with padding. Padding is a technique that modifies the data layout of a program so that the conflict misses are reduced or eliminated. The data layout modification can be done at run-time by system software [3, 19] or at compile-time by complier optimization [16]. Sharing the same objective of compiler optimization to change the base addresses of potentially conflicting cache blocks in the reorderings, we insert padding variables inside the data array. For example, the padding can be done as part of the last butterfly for the decimation in an FFT computation without additional cost, and the output is not padded.

However, we notice that this free padding opportunity may not be easily found, and the bit-reversal result may be padded in some cases. For example, the padding of a recursive implementation of the Cooley–Tukey FFT algorithm [5] is more complex than the padding in our implementations. The padding method produces padded results in a vector if the bit-reversals are done in an inplaced fashion. The accesses to the padded results need to go through a simple address converting process with additional CPU cycles. In addition, our methods target bit-reversals based on the data size of powers of 2. However, FFT algorithms are not limited to this data size. If the data size is not a power of 2, the padding method will be more complex to implement. Poor memory performance of bit-reversals has been reported even for nonpower of 2 data sizes (see, e.g., [2]).

Since the data arrays of bit-reversals form a vector whose size is power of 2, the padding is highly regular, inserting L elements or a cache line space starting at the vector positions of N/L, $2 \times N/L$, ..., and $(L-1) \times N/L$. Using L elements or a section data of a cache line to separate the vector at these L points can completely eliminate the cache conflicts caused by the address mapping based on powers of 2. Again during execution, the reordering data copies are directly conducted between the arrays X and Y without going through a data buffer. Another advantage is that the number of padding elements needed is only $L \times L$ or L cache lines and is independent of the data array size, N. Compared with the data size of bit-reversals, the number of padding elements is insignificant. Figure 2 shows how the data layout of a bit-reversal vector is modified by padding so that conflict misses are eliminated.

Compiler optimization targets a large range of application programs and automatically inserts padding variables in the programs for users. An optimal padding is application program dependent. For example, padding positions are different from different applications in order to effectively change base addresses of conflicting cache blocks [18]. Based on the unique nature of the data reordering, the optimal padding unit used by our methods for bit-reversals is a cache line with L elements. In contrast, a compiler optimization normally uses an element as the basic padding unit. How many padding units to use and where to pad in the data arrays are determined by some approximation models which may not precisely fit the unique memory access patterns of each case. In addition, applying the padding technique to bit-reversals embedded in applications would not increase complexity in the entire computation. For example, when a padded bit-reversal is performed in an FFT computation, it has little effect on the neighboring butterfly operations.

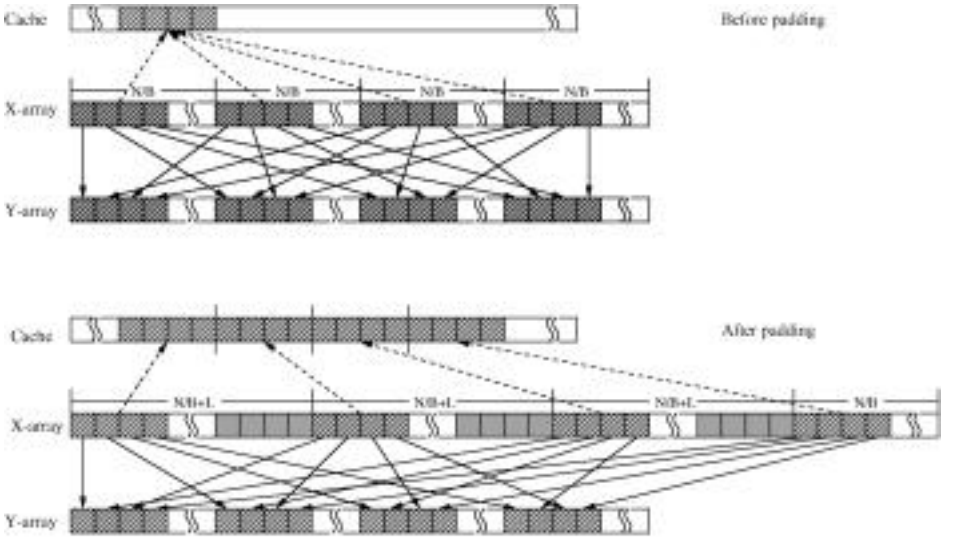


FIG. 2. *Data layout of a bit-reversal is modified by padding, where* $B = B_{cache} = L$.

5. Blocking and padding for TLB. The TLB is a special cache that stores the most recently used virtual-physical page translations for memory accesses. The TLB is a small and usually fully associative cache. Each entry points to a memory page of 4 KB to 64 KB. The page size is normally fixed at the level of operating systems and cannot be changed by user programs. A TLB cache miss will make the system retrieve the missing translation from the page table in memory and then select a TLB entry to replace. When the data to be accessed in our blocking method is larger than the amount of data of all the memory pages that the TLB can hold, we will have TLB thrashing. In this section, we will discuss and present blocking and padding methods for TLB cache optimizations.

5.1. Blocking for a fully associative TLB. Before giving a general model to show how the blocking size is affected by the TLB size, let's go through an example to show that a moderate N for bit-reversals would easily lead to TLB cache thrashing. The 64 pages in the TLB of the Sun UltraSparc-II processor hold $64 \times 1024 = 65536$ elements, which represents a 16-bit-reversal of $N = 2^{16}$. Since we have two vectors X and Y, the TLB can hold a 15-bit-reversal of $N = 2^{15}$ elements. This is also consistent with our experiments on this machine, where execution time per element was a constant until $n = 15$, but sharply increased at $n = 16$ bit-reversals caused by the TLB misses.

In our cache-optimal methods, we include an outer loop to form a blocking for TLB, whose size is denoted as B_{TLB}. The blocking size of B_{TLB} for bit-reversals when $N \geq T_s \times P_s$ is

$$B_{TLB} \leq T_s,$$

where P_s is the page size in elements, and T_s is the number of entries of the TLB. On the other hand, the B_{TLB} should be chosen as large as possible to make effective use of the page space. When $N < T_s \times P_s$, the data size of a bit-reversal will be less than the data size covered by the TLB. Thus there is no need for TLB optimizations.

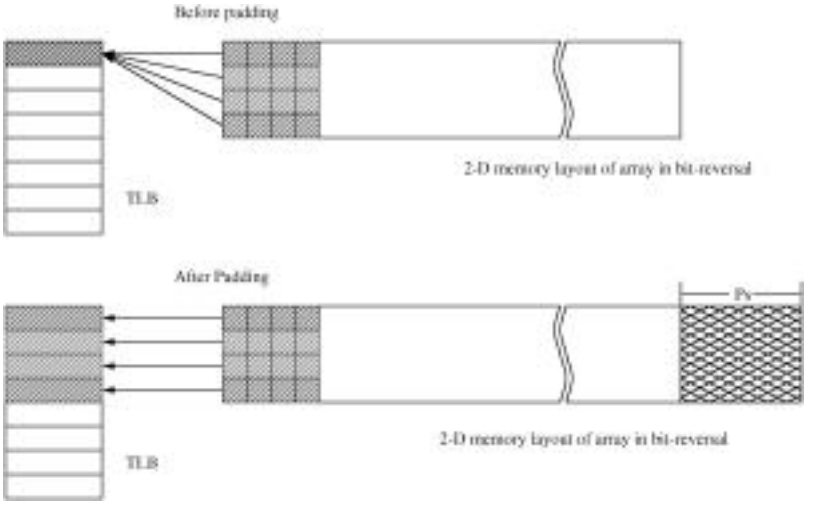


FIG. 3. *Padding for TLB: the data layout is modified by inserting a page space at multiple locations, where* $B_{TLB} = 4$, $K_{TLB} = 1$, $T_s = 8$.

5.2. Padding for a set-associative TLB. Some processors' TLBs are not fully associative, but set-associative. For example, the TLB in the Pentium-II 400 processor is 4-way associative ($K_{TLB} = 4$). A simple blocking based on the number of TLB entries is not cache-optimal, because multiple pages within a TLB-size-based blocking may map to the same TLB cache set and cause TLB cache conflict misses.

If the size N of a bit-reversal vector is a multiple of $T_s \times P_s$, where T_s is the number of TLB entries and P_s is the page size in elements, and if $K_{TLB} < B_{TLB}$, then TLB cache conflict misses will occur. This could easily happen in practice. For example, on the Pentium-II 400, N is equal to $128K$ elements (one element = 8 bytes) for a 17-bit-reversal, and this N is two times the value $T_s \times P_s$ of the machine, where $T_s = 64$, and $P_s = 1024$ elements.

In a way similar to the technique of padding for the data cache, we insert a page of elements or a page of space starting at the vector positions of N/L, $2 \times N/L, \ldots$ and $(L-1) \times N/L$ to eliminate the conflict of TLB cache misses. Figure 3 gives an example of the padding for TLB, where the TLB is a direct-mapped cache of 8 entries, blocking size is $B_{TLB} = 4$, and the number of elements of a row is a multiple of 8 page elements. Before padding, each of blocking row is mapped to the same cache line of the TLB. After padding, these rows are mapped to different cache lines of the TLB.

Combining padding for data cache and padding for TLB cache, we are inserting $L+P_s$ elements or a page plus a cache line space in L locations separated by a distance of N/L elements.

In practice, we selected more than N/L points to insert the padding variables to eliminate both data cache and TLB conflict misses. This approach could effectively merge two nested paddings (one for data cache and the other one for TLB) into a single one. An optimal number of inserting points can be easily determined experimentally based on the size of the TLB cache. The padding optimizations are all based on L2 cache in our experiments.

Partial index mapping addresses of bit-reversals are precalculated and stored in a small table as shown in the program in the appendix. This approach further improves

TABLE 1
Architectural parameters of the 5 workstations we have used for the experiments. All specifications on L1 cache refer to the L1 data cache, and all L2s are uniform. Each L2 cache block on UltraSPARC-IIi consists of 2 16-byte subblocks. The hit times of L1, L2 and the main memory are measured by lmbench [14], and their units are converted from nanosecond (ns) to their CPU cycles.

Workstations	SGI O2	Sun Ultra 5	Sun E-450	Pentium	XP1000
Processor type	R10000	UltraSparc-IIi	UltraSparc II	P-II 400	Alpha 21264
Clock rate (MHz)	150	270	300	400	500
L1 cache (KBytes)	32	16	16	16	64
L1 block size (Bytes)	32	32	32	32	64
L1 associativity	2	1	1	4	2
L1 hit time (cycles)	2	2	2	2	3
L2 cache (KBytes)	64	256	2048	256	4096
L2 block size (Bytes)	64	64	64	32	64
L2 associativity	2	2	2	4	1
L2 hit time (cycles)	13	14	10	21	15
TLB size (entries)	64	64	64	64	128
TLB associativity	64	64	64	4	128
Memory latency (cycles)	208	76	73	68	92

the performance because the table will be accessed in the cache during the computation, and the precalculation overhead is trivial. The time for the precalculation is included in the total execution time.

6. Experimental results and performance evaluation. We have implemented and tested all the bit-reversal methods discussed in the previous sections on an SGI O2 workstation, a Sun Ultra-5 workstation, a Sun SMP server E-450, a Pentium PC, and a Compaq XP1000 workstation. We will present and evaluate the performance of different methods on different machines.

6.1. Experimental environment and evaluation methodology. We used "lmbench" [14] to measure the latencies of memory hierarchies at different levels on each machine. The architectural parameters of the 5 machines are listed in Table 1.

We focus the performance evaluation on methods and implementations of bit-reversals in this paper. We compared all our methods with the method of blocking with a software buffer which was recently published in [8]. We denote this method as "bbuf-br"—blocking with buffer for bit-reversals. Two of our methods are experimentally compared: "breg-br"—blocking with associativity and registers for bit-reversals, and "bpad-br"—blocking with padding for bit-reversals. We have also applied blocking or padding technique for the TLB in these two methods based on the TLB associativity.

All the programs use a standard subroutine to calculate the bit-reversal value for a given address. The execution times were collected by "gettimeofday()," a standard Unix timing function. The resolution of this function is 1 μs on the machines being measured, which is significantly smaller than the execution times of any programs we have measured. A small bit-reversal table is precalculated, and we exclude this calculation time. The reported time unit is cycles per element (CPE):

$$CPE = \frac{execution\ time \times clock\ rate}{N},$$

where *execution time* is the measured time in seconds, *clock rate* is the CPU speed (cycles/second) of the machine where the program is run, and N is the number of elements of the bit-reversal program. Besides the different methods of bit-reversals, we also measured the execution time of a program copying elements between X and Y. This program has the same number of data copying operations with a continuous memory access pattern. We use the execution time of this program to provide a base line reference for bit-reversal programs and show how close a bit-reversal execution is to its ideal time. We denote this reference program as "base." Each method is further divided into "float" data type using 4 bytes to represent an element, and "double" type using 8 bytes to represent an element. The data type divisions will show the performance impact of the cache line length.

For all experiments on different machines, the bit-reversal programs first call a routine to flush the cache to make sure that all the data are allocated only in the memory. All experiments were repeated multiple times.

6.2. Effects of TLB and virtual memory. Before measuring and comparing the performance of different bit-reversal methods, we experimentally evaluated the effects of TLB and virtual memory to confirm our assumptions and analyses.

Selection of TLB blocking size. The TLB blocking size is a sensitive performance parameter to be selected, which is determined by the size of the TLB if it is fully associative. We executed program "bpad-br" (blocking with padding for bit-reversals) with $n = 20$ on a single node of Sun E-450 by changing the blocking sizes for TLB from 8 to 128. The TLB of the E-450 is a fully associative cache with 64 entries. Figure 4 shows the measured cycles per element of the program of different blocking sizes on the node. Our experimental results are consistent with our analyses in the previous section. When the blocking size for TLB was 64, the execution time curve increased sharply. This is because arrays X and Y together demanded more than 64 pages and caused TLB thrashing.

Virtual memory versus physical memory addresses. All our analyses are based on cache mappings between memory pages in the virtual address space and cache blocks in the physical memory address space. This assumes that contiguous memory pages will be contiguously mapped to the cache. This assumption is guaranteed for the virtual-address caches [4]. However, all our experiments have been performed on machines with physical address L2 caches. Since the virtual-physical translations for L2 caches are handled by operating systems, our assumptions may sometimes be inaccurate. In order to show that many operating systems attempt to map contiguous virtual pages to cache blocks contiguously so that our virtual-address-based study is practically meaningful and effective, we conducted a simulation by using the SimOS [17] and measurements on different workstations to observe how an operating system makes translations from virtual memory addresses to their physical addresses.

The SimOS simulates a complete hardware of SGI machines and runs the IRIX 5.3 operating system in the simulation. We executed a blocking-only program of bit-reversals using the cache line L as the blocking size. The bit-reversal vector size was changed from $n = 15$ to $n = 22$. We measured the miss rates on array X. The cache size was set to 2 MB holding two double type arrays up to $n = 18$ in the virtual memory space. Figure 5 gives consistent results from the SimOS simulation: when $n > 18$, the miss rate on array X was sharply increased to 100% from 12.5%.

From this experiment, we have observed that virtual-physical translations from

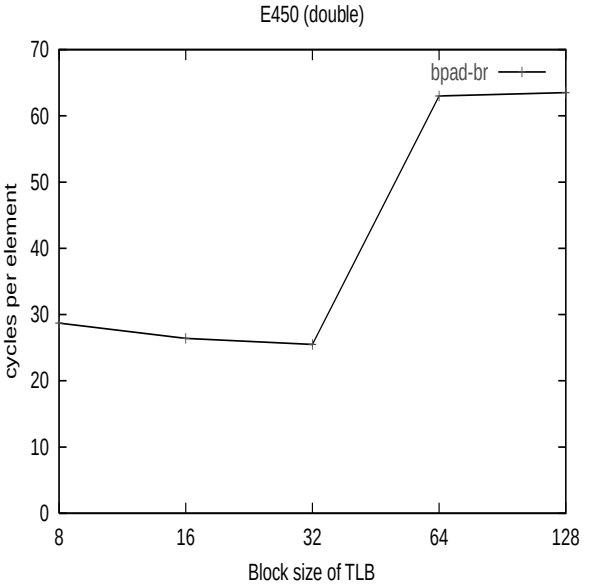


FIG. 4. *Changing the TLB blocking sizes on a single node of the Sun E-450: when the blocking size for TLB was larger than* 32*, the execution time curve was sharply increased.*

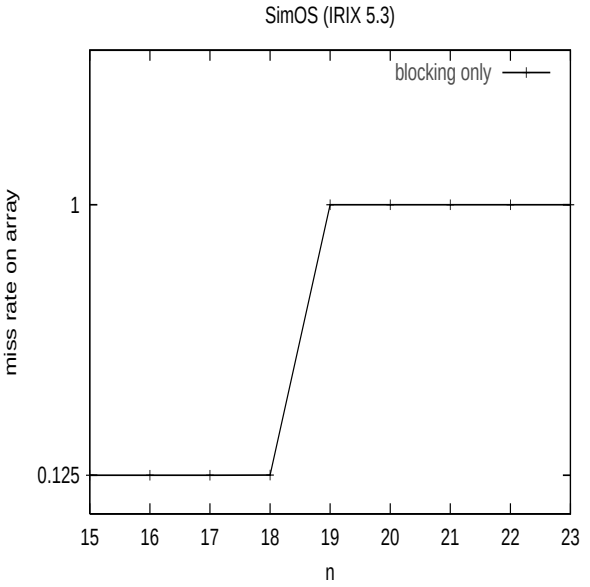


FIG. 5. *Using the SimOS to observe the miss rates by changing the size of the bit-reversal arrays of a blocking-only program: when* $n > 18$*, the miss rate was sharply increased to* 100%.

the IRIX 5.3 operating system are quite consistent with our assumption of "contiguous allocations."

We have also run the similar experiments on different targeted workstations with different operating systems, such as Linux and Solaris, to measure the changes of execution times when the data size is changed. Our measurements are also consistent

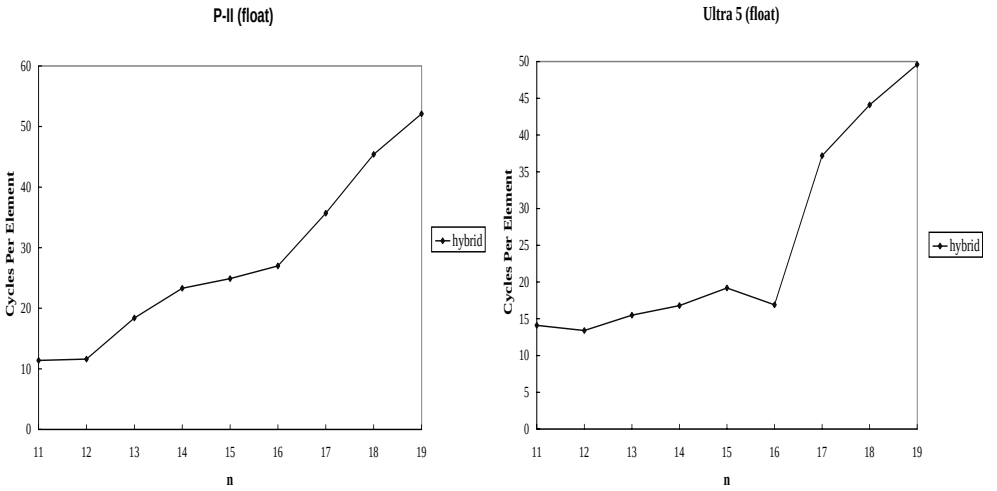


FIG. 6. *Execution times of the hybrid method on the Pentium*-II *(left figure) and on the Ultra*-5 *machine (right figure).*

to the SimOS results and indicate that the larger the data arrays to be used, the more likely an operating system will allocate the pages contiguously. Because our study targets large data sets, our analyses based on the virtual memory space is reasonably accurate. In addition, our methods assume that the operating system uses a uniform page size for page allocation, which is consistent with most commercial and commonly used operating systems.

6.3. Performance of the hybrid method for bit-reversals. In order to show the effectiveness of our cache optimizations, we first plot the measured execution times of the hybrid method[1] in "float" data types on the Pentium-II and the Ultra-5 machines in Figure 6. Although the hybrid method did reasonably well for $n \leq 16$ on Pentium-II and $n \leq 12$ on Ultra-5, the execution times significantly increased due to limited cache performance after the data size was further increased.

6.4. Performance comparisons on the SGI O2. The SGI O2 is a 1995 product using an R10000 processor of 150 MHz, 32 KB 2-way associative L1 cache, and 64 KB 2-way associative L2 cache. The cache line of L2 is 64 bytes. Since the associativity of L2 is low, and the cache line of L2 is relatively long, it is difficult to do blocking with associativity and available registers. We implemented only the blocking with padding method to compare with blocking with software buffer and the base reference.

We scaled bit-reversal methods from $n = 16$ to $n = 21$. Figure 7 shows the comparisons of CPE among the three programs of both "float" type and "double" type on the SGI O2 machine. The measurements show that the padding method slightly reduced the execution time compared with the method of blocking with software buffer. The time reduction was up to 6%. The reason for the small performance improvement comes from the extremely long memory latency (208 cycles) of the O2 machine. The reduction and saving of instruction cycles for data copies from padding became less significant because memory latencies caused by the required cold misses in both methods were dominant in execution.

[1]The program was written in Fortran by Alan Karp.

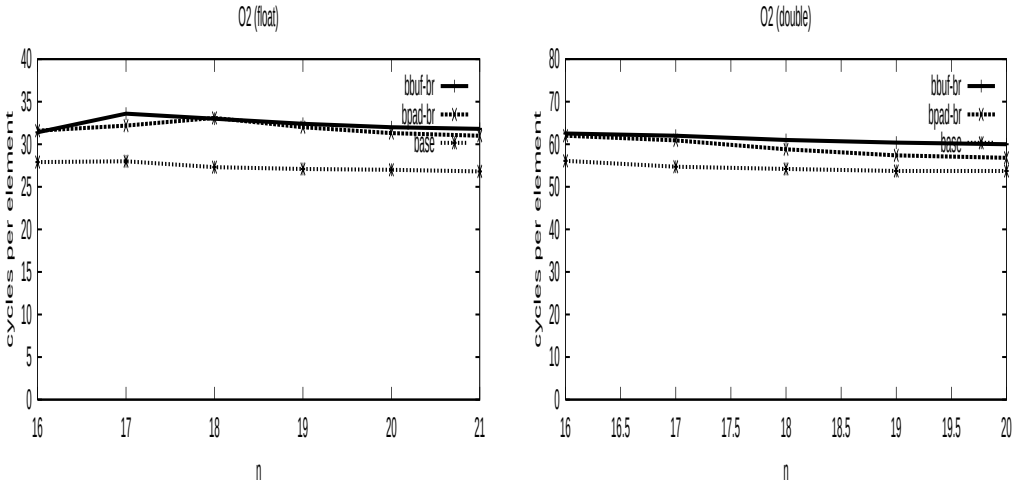


FIG. 7. *Execution comparisons on the SGI O2 workstation: "bbuf-br" represents the method of blocking with software buffer; "bpad-br" represents the method of blocking with padding; and "base" represents the ideal base line reference.*

6.5. Performance comparisons on the Sun Ultra-5. The Sun Ultra-5 is a 1998 product using an UltraSparc-IIi processor of 275 MHz, 16 KB direct-mapped L1 cache, and 256 KB 2-way associative L2 cache. The cache line of L1 is 32 bytes consisting of two 16-byte subblocks, and L2 is 64 bytes long. Similar to the SGI O2, the associativity of L2 on the Ultra-5 is low, and the cache line of L2 is relatively long, so it is difficult to do blocking with associativity and available registers. We implemented only the blocking with padding method to compare with blocking with software buffer and the base reference.

We scaled the bit-reversal methods from $n = 16$ to $n = 23$. Figure 8 shows the comparisons of cycles per element among the three programs of both "float" type and "double" type on the Ultra-5. The memory latency of the Ultra-5 (76 cycles) is significantly lower than that of the O2. We observed a more significant performance improvement from the method of blocking with padding over that of blocking with software buffer. For example, using "float" type, the padding program is 14% faster than that of blocking with buffer for $n = 20$ or larger. A L2 cache line of the Ultra-5 holds 16 "float" type elements ($L = 16$), and 8 "double" type elements ($L = 8$). The larger the L, the higher overhead the blocking with software buffer will have. This has been confirmed by our comparative experiments between the "float" and "double" types on the Ultra-5 shown in Figure 8.

6.6. Performance comparisons on the Sun E-450. The Sun E-450 is a 1998 4-processor SMP product. Each of the 4 nodes is an UltraSparc-2 processor of 300 MHz, 16 KB direct-mapped L1 cache, and 2 MB 2-way associative L2 cache. The cache line of L1 is 32 bytes consisting of two 16-byte subblocks, and L2 is 64 bytes long. Due to the limited associativity and a relatively long L2 cache line, we implemented only the blocking with padding method to compare with blocking with software buffer and the base reference.

We scaled the bit-reversal methods from $n = 16$ to $n = 25$. Figure 9 shows the comparisons of CPE among blocking with software buffer, blocking with padding, and the base program on a single node of E-450, each of which has both "float" type and

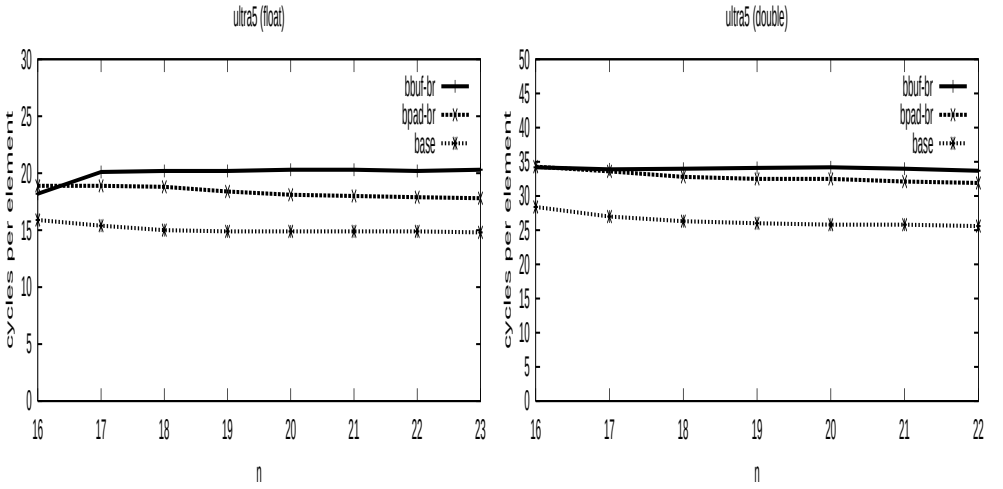


FIG. 8. *Execution comparisons on the Sun Ultra-5 workstation: "bbuf-br" represents the method of blocking with software buffer; "bpad-br" represents the method of blocking with padding; and "base" represents the ideal base line reference.*

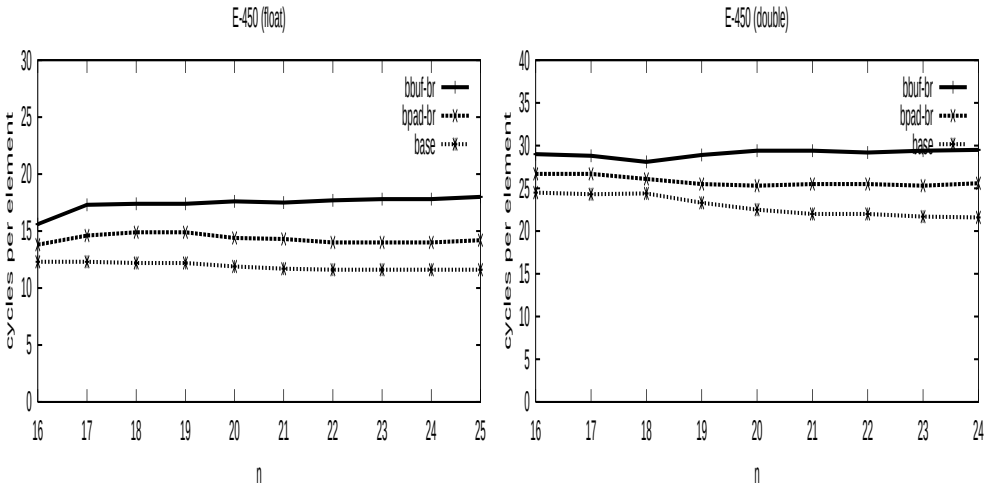


FIG. 9. *Execution comparisons on the Sun E-450 SMP: "bbuf-br" represents the method of blocking with software buffer; "bpad-br" represents the method of blocking with padding; and "base" represents the ideal base line reference.*

"double" type. The memory latency of the Ultra-5 (73 cycles) is slightly lower than that of Ultra-5. On this machine, we observed higher performance improvement from the method of blocking with padding over that of blocking with software buffer. For example, using "float" type, the padding program is 22% faster than that of blocking with buffer for $n = 20$ or larger. Our comparative experiments between the "float" and "double" types on E-450 in Figure 9 also confirms that the larger the L, the higher performance the padding method would achieve.

6.7. Performance comparisons on the Pentium-II 400. The Pentium PC we used is a 1998 product using a Pentium-II 400 processor of 400 MHz, 8 KB direct-mapped L1 cache, and 256 KB 4-way associative L2 cache. The cache lines of both

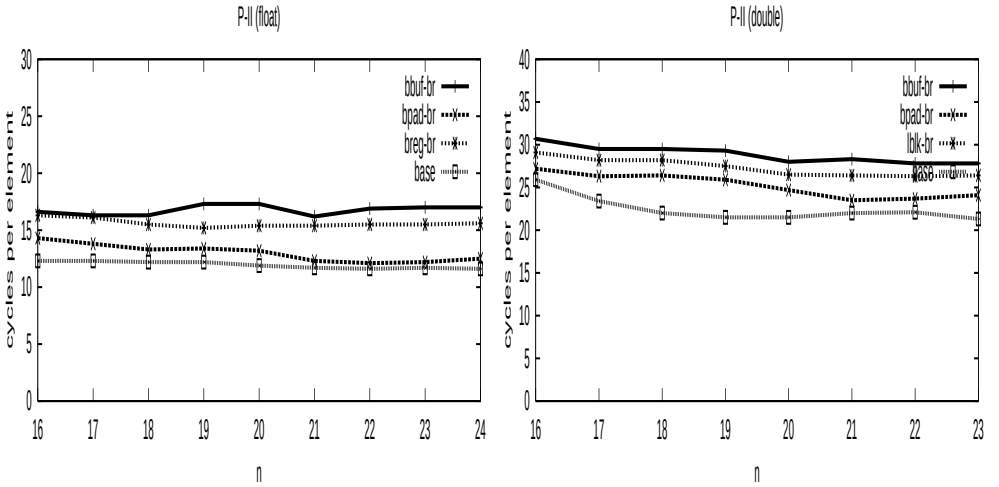


FIG. 10. *Execution comparisons on the Pentium*-II 4000 *PC: "bbuf-br" represents the method of blocking with software buffer; "bpad-br" represents the method of blocking with padding; "breg-br" represents the method of blocking with associativity and registers; and "base" represents the ideal base line reference.*

L1 and L2 are 32 bytes. Since the L2 associativity is high, we are able to implement the method of blocking with associativity and available registers, L2 cache line $L = 8$ elements for a "float" type, and we need $(L-K)(L-K) = 16$ registers to supplement the 4-way associative cache. An L2 cache line holds 4 "double" type elements ($L = 4$). Thus, we do not need any registers to supplement but simply make a 4×4 blocking. The TLB of the Pentium processor is a 4-way associative cache of 64 entries. We used our padding for the TLB technique to avoid TLB misses. We implemented the blocking with padding method and the blocking with associativity and registers to compare with blocking with software buffer and the base reference.

We scaled the bit-reversal methods from $n = 16$ to $n = 24$. Figure 10 shows the comparisons of cycles per element among the four programs. As we expected, the paddings for both cache and TLB were highly effective, and the padding program performed the best. For example, using "float" type, the padding program is about 40% faster than that of blocking with buffer for $n = 22$ or larger. We also show that the method using available registers to supplement associativity is effective. Although it is not as good as the padding program due to the increase of the instruction counts for additional data copies, it still achieved up to 12% execution reduction over the blocking with software buffer program. As we expected, the execution time of the method using the 4-way associative L2 cache without the supplement of registers to form a 4×4 blocking was delayed mainly by the longer L2 cache hit time. The performance of this method still outperformed the method of blocking with a software buffer.

6.8. Performance comparisons on the Compaq XP-1000. The Compaq XP-1000 is a 1999 product using an Alpha 21264 processor of 500 MHz, 64 KB 2-way associative L1 cache, and 4 MB 2-way associative L2 cache. The cache lines of both L1 and L2 are 64 bytes long. Similar to the SGI and Sun machines, the associativity of L2 on the XP 1000 is low, and the cache line of L2 is relatively long, so it is difficult to do blocking with associativity and available registers. We implemented only the

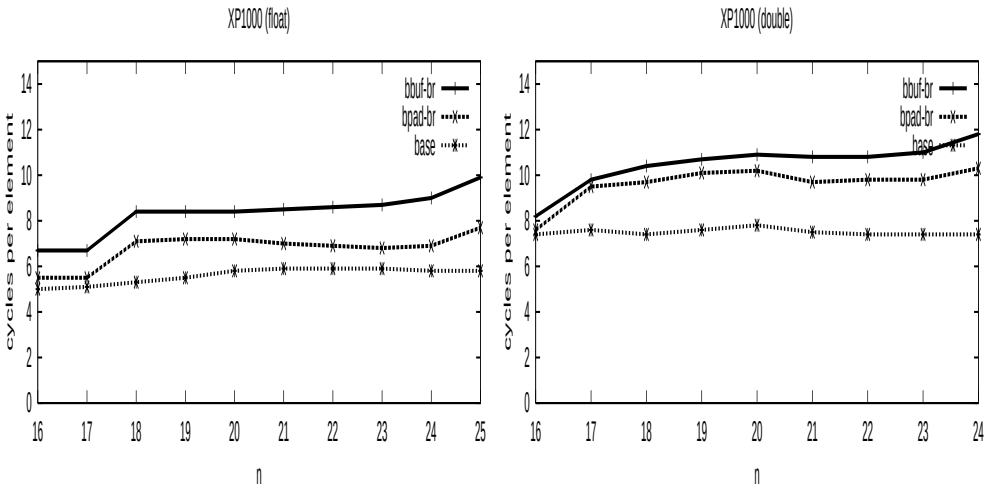


FIG. 11. *Execution comparisons on the Compaq XP-*1000 *workstation: "bbuf-br" represents the method of blocking with software buffer; "bpad-br" represents the method of blocking with padding; and "base" represents the ideal base line reference.*

blocking with padding method to compare with blocking with software buffer and the base reference.

We scaled the bit-reversal methods from $n = 16$ to $n = 25$. Figure 11 shows the comparisons of CPE among the three programs of both "float" type and "double" type on the XP-1000 machine. As we expected, we achieved better or comparable performance to the ones on the Sun machines. For example, using "float" type, for $n = 24$ or larger, the padding program is 30% faster than that of blocking with buffer, and 15% faster for "double" type.

7. Performance evaluation on SMP multiprocessors. We implemented the bit-reversal methods on two SMP multiprocessors: the Sun E-450 and the HP 9000 V2200. The parallel bit-reversal program on an SMP with M processors is described using POSIX thread primitives [10] as follows:

```
bit_reversal(id)
  my_start = id*(N/M);
  my_end = (id-1)*(N/M);
  for i = 1, N
    Y[i'] = X[i];
```

The bit-reversal operations are evenly distributed among M processors.

7.1. Performance comparisons on the Sun E-450. The Sun E-450 is a 1998 4-processor SMP product. Each of the 4 nodes is an UltraSparc-2 processor of 300 MHz, 16 KB direct-mapped L1 cache, and 2 MB 2-way associative L2 cache. The cache line of L1 is 32 bytes consisting of two 16-byte subblocks, and L2 cache line is 64 bytes. Due to the limited associativity and a relatively long L2 cache line, we implemented only the blocking with padding algorithm to compare with blocking with software buffer and the base reference.

We scaled the bit-reversal algorithms from $n = 16$ to $n = 24$. Figure 12 shows the comparisons of CPE among blocking with software buffer, blocking with padding, and the base program on the E-450 of 4 nodes, each of which has both "float" type and "double" type. On this machine, we observed some performance improvement

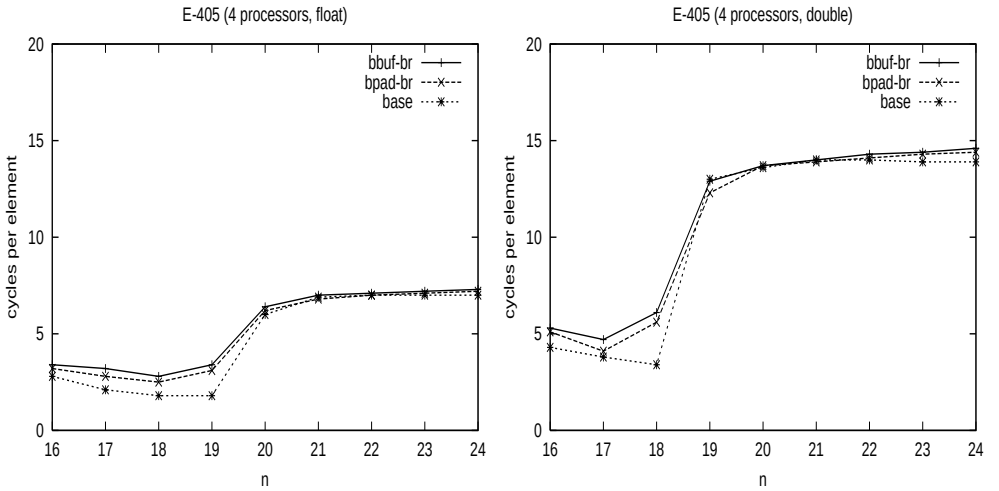


FIG. 12. *Execution comparisons on Sun E-450 SMP of 4 processors: "bbuf-br" represents the algorithm of blocking with software buffer; "bpad-br" represents the algorithm of blocking with padding; and "base" represents the ideal base line reference.*

when $n \leq 18$ from the algorithm of blocking with padding over that of blocking with software buffer.

However, when $n > 18$ of double type or $n > 19$ of float type, each processor has to process a data set larger than its cache capacity. Multiple processors simultaneously access the memory through a shared data link would cause the contention to degrade the performance. Since the data to be accessed from different processors are distributed in different locations, a crossbar interconnection network to link each processor to all the memory modules would significantly reduce the contention. The E-450 does have a 5×5 crossbar to connect 2 pairs of processors, 2 I/O ports, and the memory. The communications between the 4 processors the memory modules are connected through the single memory data link. Figure 13 shows the crossbar interconnections of the E-450 among the processors, the shared-memory modules, and the 2 I/O ports. The contention occurs in the memory data link when the multiple processors request memory accesses simultaneously.

We have observed severe performance degradation caused by the memory access contention. Figure 12 shows that this contention makes the execution time curves of the three programs jump sharply and merge together when $n > 18$ of double type and $n > 19$ of float type. In contrast, on a single processor of E-450, accesses to the memory through the memory bus have no contention so that the algorithms were scaled well.

7.2. Performance comparisons on the HP 9000 V2200. HP 9000 V2200 is a 1997 SMP product with up to 16 processors. We used 4 processors for performance comparisons. Each node is a HP PA-8200 processor of 200 MHz with a 2 MB direct-mapped L1 data cache. The cache line is 32 bytes. Due to limited associativity, we implemented only the blocking with padding algorithm to compare with blocking with software buffer and the base reference.

The HP SMP has a crossbar interconnection network, the HyperPlane crossbar, to connect up to 8 pairs of processors to 8 memory modules. Multiple pairs of processors can access different memory modules simultaneously. Each pair of the processors is

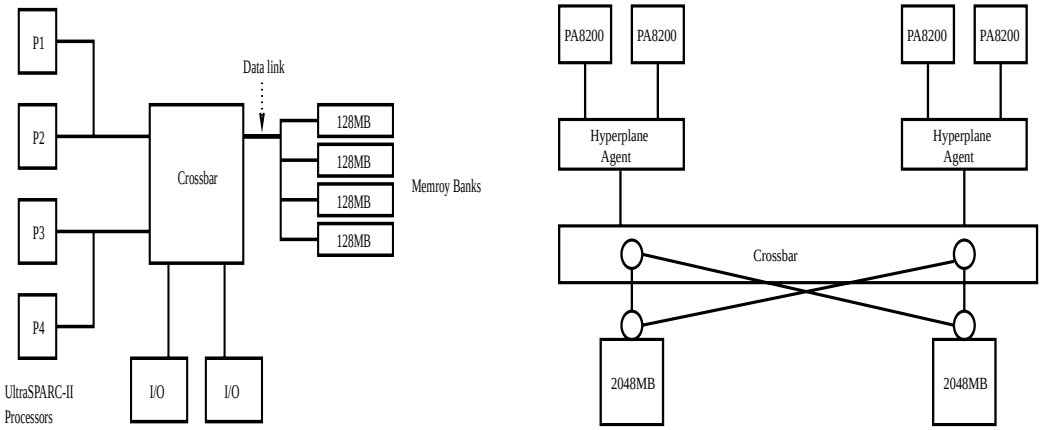


FIG. 13. *Architecture comparisons between Sun E-*450 *SMP (left) and HP* 9000 *V*2200 *SMP (right): the memory data link of the E-*450 *may become a bottleneck when simultaneous memory access requests from multiple processors; the HyperPlane crossbar connected between the memory modules and the processors on the HP* 9000 *V*2200 *can effectively reduce the contention.*

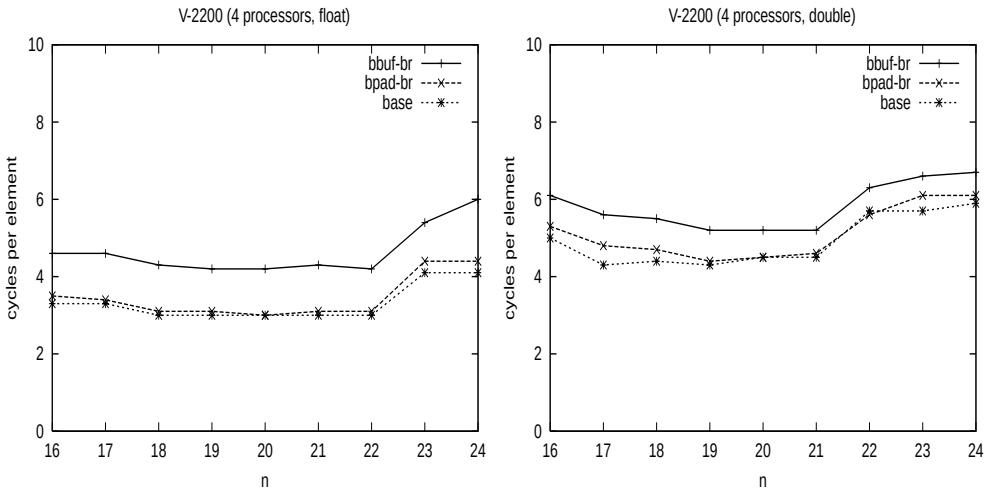


FIG. 14. *Execution comparisons on HP* 9000 *V*2200*: "bbuf-br" represents the algorithm of blocking with software buffer; "bpad-br" represents the algorithm of blocking with padding; and "base" represents the ideal base line reference.*

connected to the crossbar through an adaptor called HyperPlane Runway Agent. Figure 13 gives the interconnection structure of the HP 9000 V2200 of 4 processors.

In our experiments, the 4 processors are divided into 2 pairs which are connected to 2 memory modules by a 2×2 hyperplane crossbar. Each pair of processors may have contention to compete the adaptor, but the crossbar is able to allow simultaneous data accesses among the memory modules. The negative performance effect due to the data link contention observed on Sun E-450 was significantly reduced on the HP SMP, which shows the effectiveness of the crossbar. Figure 14 shows comparative execution time curves between the "float" and "double" types on E-450 in Figure 14. The execution times of the 3 programs are quite stable and independent of the size of n. Both the padding programs of the float type and of the double type outperformed

TABLE 2

Summary of the blocking methods and their impact on the three aspects of performance (cross interference, instruction count, and memory space) and on the program complexity. The performance of "blocking only" method is the base line for comparisons. Note: + means that the method quantitatively increases the factor and hurts the performance, and blank means it has no impact. The program complicity is subjective and compared with the "block only" method, with 1 *being a slightly more complex, and* 2 *a moderately more complex.*

Methods	Cross interference	Instruction count	Memory space	Program complexity	Comments
Blocking only				0	limited by data sizes.
Blocking with software buffer	+	+	+	1	system independent.
Blocking with register buffer				1	limited by the number of available registers.
Blocking with associativity and registers				2	works well on high associativity caches.
Blocking with padding			+	1	works well on all systems.
TLB blocking				0	a TLB size dependent outer loop, effective for fully associative TLBs.
TLB padding			+	1	paddings by using L pages, effective for set associative TLBs.

the blocking methods with buffer up to 40% and 18%, respectively. Their execution curves almost merge together with the base reference curve.

8. Conclusion. We have examined and developed cache-optimal methods for bit-reversal data reorderings. These methods have been tested on 5 representative uniprocessor workstations of 1995 to 1999 products to show their effectiveness. Different methods have their own merits and limits. The blocking-only method is limited by data sizes. Although the blocking-with-software-buffer method is architecture independent, it increases cross interference and instruction count and needs additional memory space. The blocking-with-a-register-buffer method is fast but is limited by the number of available registers. Blocking with associativity and with registers work well on high associativity caches. We have shown that the methods of blocking with padding, blocking for TLB, and padding for TLB can effectively exploit cache locality and are almost independent on hardware. Thus, they could be widely used on many uniprocessors workstations and SMP multiprocessors. We summarize different techniques and their merits and limits in Table 2, which gives a guideline for application users to choose a technique based on the size of the problem and the machines available.

The methods have also tested on two commercial SMP multiprocessors. By exploiting cache locality of each processor, we have effectively eliminated the conflict misses so that accesses to the shared memory and contention are minimized. However, another potential bottleneck on SMPs is the data access contention to the shared-memory. We show that crossbar interconnections between processors and memory modules play an important role to parallel bit-reversal data reorderings.

Appendix.

```
/* This is a padded bit-reversal program for cache optimization. */
void bit_reversal()
{
   int blk, blk_rev, i, i_rev, j, jump = PAD_LENGTH, k;
   int D = N >> 2*b, d = n - 2*b;
   DATA_TYPE *Xp[B];
   DATA_TYPE *Yp, f0, f1, f2, f3;

   for (i = 0; i < B; i ++)
      Xp[i] = &X[bitrev_tbl[i]*jump];

   for (blk = 0; blk < D; blk ++) {
      bitrev(blk, blk_rev, d);
      for (i = 0; i < B; i ++) {
         i_rev = bitrev_tbl[i];
         k = (blk << b) + i;
         Yp = &Y[(blk_rev<<b) + (i_rev<<(n-b))];
         for (j = 0; j < B; j += 4) {
            f0 = Xp[j][k];
            f1 = Xp[j+1][k];
            f2 = Xp[j+2][k];
            f3 = Xp[j+3][k];
            Yp[j] = f0;
            Yp[j+1] = f1;
            Yp[j+2] = f2;
            Yp[j+3] = f3;
         }
      }
   }
}
```

Acknowledgments. We feel fortunate to have had Alan Karp's expert views and comments on this work. We have also exchanged bit-reversal programs of different methods to compare the performance. The comments from the anonymous referees were helpful. We thank Kang Su Gatlin for his constructive suggestions on a preliminary version of this paper. Neal Wagner carefully read the manuscript and made useful comments.

REFERENCES

[1] D. F. Bacon, S. L. Graham, and O. J. Sharp, *Compiler transformations for high performance computing*, ACM Computing Surveys, 26 (1994), pp. 345–420.

[2] D. H. Bailey, *FFTs in external or hierarchical memory*, J. Supercomputing, 4 (1990), pp. 23–35.

[3] B. Bershad, D. Lee, T. Romer, and B. Chen, *Avoiding conflict misses dynamically in large direct-mapped caches*, in Proceedings of the Sixth International Conference on Architectural Support for Programming Languages and Operating Systems, 1994, pp. 158–170.

[4] M. Cekleov and M. Dubois, *Virtual-address caches*, IEEE Micro, 17 (1997), pp. 64–71.

[5] J. W. Cooley and J. W. Tukey, *An algorithm for the machine calculation of complex Fourier series*, Math. Comp., 19 (1965), pp. 297–301.

[6] A. Edelman, *Optimal matrix transpose and bit reversal on hypercube: All-to-all personalized communication*, J. Parallel Distrib. Comput., 11 (1991), pp. 328–331.

[7] D. M. W. Evans, *An improved digit-reversal permutation algorithm for the fast Fourier and Hartley transforms*, IEEE Trans. Acoust. Speech Signal Process., 35 (1987), pp. 1120–1125.

[8] K. S. Gatlin and L. Carter, *Memory hierarchy considerations for fast transpose and bit-reversals*, in Proceedings of Fifth International Symposium on High-Performance Computer Architecture, 1999, pp. 33–44.

[9] S. L. Johnsson and C.-T. Ho, *Algorithms for matrix transposition on Boolean N-cube configured ensemble architectures*, SIAM J. Matrix Anal. Appl., 9 (1988), pp. 419–454.

[10] IEEE, *POSIX P*1003.4*a: Threads Extension for Portable Operating Systems*, IEEE Press, Piscataway, NJ, 1994.

[11] A. H. Karp, *Bit reversal on uniprocessors*, SIAM Rev., 38 (1996), pp. 1–26.

[12] J. L. Hennessy and D. A. Patterson, *Computer Architecture: A Quantitative Approach*, Morgan-Kaufmann, San Francisco, 1996.

[13] N. P. Jouppi, *Improving direct-mapped cache performance by the addition of a small fully-associative cache and prefetch buffers*, in Proceedings of 17th Annual International Symposium on Computer Architecture, 1990, pp. 364–373.

[14] L. McVoy and C. Staelin, *lmbench: Portable tools for performance analysis*, in Proceedings of the 1996 USENIX Technical Conference, 1996, pp. 279–295.

[15] P. N. Swarztrauber, *FFT algorithms for vector computers*, Parallel Comput., 1 (1984), pp. 45–63.

[16] C. Rivera and C.-W. Tseng, *Data transformations for eliminating conflict misses*, in Proceedings of the ACM SIGPLAN'98 Conference on Programming Language Design and Implementation, 1998, pp. 38–49.

[17] M. Rosenblum, E. Bugnion, S. Devine, and S. A. Herrod, *Using the SimOS machine simulator to study complex computer systems*, ACM Transactions on Modeling and Computer Simulation, 7 (1997), pp. 78–103.

[18] L. Xiao, X. Zhang, and S. A. Kubricht, *Improving memory performance of sorting algorithms*, 5 (2000), pp. 1–23.

[19] Y. Yan, X. Zhang, and Z. Zhang, *Cacheminer: A runtime approach to exploit cache locality on SMP*, IEEE Trans. Parallel Distrib. Systems, 11 (2000), pp. 357–374.

[20] C. Zhang, X. Zhang, and Y. Yan, *Two fast and high-associativity cache schemes*, IEEE Micro, 17 (1997), pp. 40–49.

SIAM J. SCI. COMPUT.
Vol. 22, No. 6, pp. 2135–2151
© 2001 Society for Industrial and Applied Mathematics

CHECKPOINTING SCHEMES FOR ADJOINT CODES: APPLICATION TO THE METEOROLOGICAL MODEL MESO-NH*

I. CHARPENTIER†

Abstract. The adjoint code of a nonlinear computer model calculates gradients along a trajectory that has to be known at integration time. When the storage of the whole trajectory requires too large an amount of memory, the calculation of the adjoint code is split and is done part by part from restart points called checkpoints. Griewank proposed a checkpointing method named *Revolve*, which provides an optimal logarithmic behavior with respect to time and memory requirement. In this work, some checkpointing schedules are proposed. Some of them correspond to special cases of *Revolve*.

The user's preference is essential to choose between time and memory requirements. This is a key point for adjoint codes of temporal models such as the meteorological model Meso-NH that may be used for weather forecasts. When the computational time is the top priority, a particular checkpointing scheme allows computation of the adjoint code with at most one extra integration of the model. The memory requirement behaves then as the square root of the number of iterations of the model. Checkpointing schemes are tested on adjoint simulations of Meso-NH.

Key words. checkpointing algorithms, adjoint codes, inverse problems, nonlinear least squares, three-dimensional (3D) meteorological simulations, leap-frog schemes

AMS subject classifications. 65D25, 65K10, 68-04, 86-04, 86A10, 65-05

PII. S1064827598343735

1. Introduction.

Computational methods using derivatives are now classical for solving inverse problems through optimal control theory [14]. From a scientific computing point of view, the latter are nonlinear least square calculations on a vector function that is defined as a solution of a partial differential equation problem. Such optimization problems may be solved using gradients that are calculated via the differentiation of the numerical code representing the original model.

According to theoretical bounds (see, for example, [17] and [6]), the reverse mode of differentiation allows the generation of an adjoint code involving at most five times the number of operations of the original model. However, this surprisingly low operation count requires the storage of the full trajectory. The *trajectory* is formed by the values of the variables of the original model that may be used for the evaluation of linearized statements. The calculation of the trajectory has to be performed before (or during) the backward integration of the adjoint code. When the differentiated codes require too large an amount of memory, a possible solution is to implement algorithms such as Griewank's *Revolve* checkpointing method (named *Treeverse* in [7]), which provides an optimal logarithmic behavior in terms of time and memory requirement.

More generally, the strategy to solve the trajectory problem arising in adjoint computations is based on the classical idea of "divide and conquer." Since the whole trajectory cannot be stored, the calculation of the adjoint code is split and done part by part from restart points called *checkpoints*. Extra forward iterations of the original

*Received by the editors August 21, 1998; accepted for publication (in revised form) November 28, 2000; published electronically April 12, 2001. This work was supported by the INRIA action for Operative Inverse Mode, the CEMRACS'97, and the IDRIS computational center.

http://www.siam.org/journals/sisc/22-6/34373.html

†Projet IDOPT, LMC-IMAG, 51 Rue des Mathématiques, BP 53, F-38041 Grenoble Cedex 9, France (Isabelle.Charpentier@imag.fr).

model are then necessary to progress between checkpoints. The main problem is to find a partition of the computations that minimizes the number of extra forward iterations. The paper describes and analyzes several algorithms in order to facilitate the user's preference between computational time and memory requirement. Some of them correspond to particular parameter choices in *Revolve*. Particular attention is devoted to evolutionary problems solved using leap-frog schemes [1]. Numerical tests are performed on the meteorological model Meso-NH [12] that may be used for operational (real time) weather forecasts. The differentiation work that generates the adjoint of Meso-NH was published in [2] and [1].

The layout of the paper is as follows. Section 2 introduces optimal control methods suitable for the meteorological model Meso-NH and points out the problem of the size of the trajectory. Section 3 presents checkpointing algorithms that allow the solution of the trajectory problem, whereas the case of leap-frog schemes is discussed in section 4. Section 5 reports numerical results, and section 6 concludes on some discussion about the efficiency of checkpointing schemes.

2. Computation of gradients. Derivatives are very helpful for a large number of numerical methods in very different fields. For example, derivatives are usable for performing sensitivity analysis of a numerical model with respect to perturbations of its initial conditions, or for the validation of a model with respect to some of its parameters. Likewise, inverse problems may be solved through gradient methods. These are nowadays in use for weather forecasts [18].

2.1. Variational data assimilation process. Variational data assimilation techniques [3], [13] are used in most of the operational meteorological models. Let Ω be an open bounded domain of the atmosphere, let $[0,T]$ be the time interval, and let X be the *state variable* belonging to the set $\mathcal{X}$ of admissible states of the atmosphere. For the sake of simplicity, the governing equations that define the *original model* are written in the following form:

$$\begin{cases} \dfrac{dX}{dt} = F(X) \quad \text{in } \Omega \times (0,T), \\ \\ X(0) = X_0 \quad \text{in } \Omega, \\ + \text{ boundary conditions.} \end{cases} \tag{2.1}$$

In the previous system, F is a nonlinear differentiable operator from $\mathcal{X}$ to $\mathcal{X}$ that describes the dynamic behavior of the model, and X_0 is the initial state. System (2.1) is supposed to have a unique solution in $\mathcal{X}$ for all $t \in [0,T]$.

When studying the atmospheric circulation, the main goal to achieve has always been weather forecasting, i.e., one wants to predict the state of the atmosphere after time T. This requires a good numerical model and the knowledge of a "good" approximation of the state of the atmosphere at time T.

The aim of the data assimilation process is to find this good approximation by including observed data in the model. Observed data $X_{obs} \in \mathcal{X}_{obs}$ that are acquired during the interval [0,T], for example, by surface stations and/or pilot balloons are introduced into the numerical model by means of a cost function $J : \mathcal{X} \longrightarrow \mathbb{R}$ such that

$$J(X_0) = \frac{1}{2} \int_0^T ||H(X) - X_{obs}||^2 dt. \tag{2.2}$$

The observation operator H maps $\mathcal{X}$ to $\mathcal{X}_{obs}$. Let X_0^{opt} be the solution of the nonlinear least square problem

$$\textit{Find } X_0 \textit{ such that } J(X_0) \textit{ is minimal.} \tag{2.3}$$

A well-known necessary condition for the local optimality is then the stationarity equation

$$\nabla J(X_0) = 0.$$

As a consequence, the optimal state X_0^{opt} is the initial state such that the solution X of (2.1) fits at best the observed data. Meteorologists use the resulting solution $X(T)$ as a fair initial condition for forecasting experiments.

Algorithmic differentiation [8] proposes two methods for the calculation of the gradient of J: the direct mode that generates tangent linear codes and the reverse mode that generates adjoint codes. This paper is only concerned with the second one. The reader is referred to [8] for more information on algorithmic differentiation.

2.2. Reverse mode of differentiation. Let the *adjoint variable* P be the solution of the *adjoint system*

$$\begin{cases} \dfrac{dP}{dt} + \left[\dfrac{\partial F}{\partial X}(X)\right]^* P = H^*(H(X) - X_{obs}) \quad \text{in } \Omega \times (0,T), \\ P(T) = 0 \quad \text{in } \Omega, \\ + \text{ boundary conditions}, \end{cases} \tag{2.4}$$

where $[\frac{\partial F}{\partial X}(X_0)]^*$ denotes the transposed Jacobian of F. One may prove [13] that the gradient of J is given by

$$\nabla J = -P(0). \tag{2.5}$$

According to (2.5), the computation of all the components of ∇J requires only a single evaluation of the adjoint system. Moreover, theoretical results indicate that the ratio between the number of operations of the adjoint statement and the number of operations of the original statement is lower than 5 for any statement.

In the case of nonlinear equations, it is necessary to solve the original system (2.1) to compute and to record the trajectory before the evaluation of the adjoint system (2.4). This constitutes what one calls the *forward sweep*. The trajectory is then restored to evaluate adjoint statements during the *reverse sweep*.

The management of the trajectory is often the key point of the adjoint code. There exist essentially two manners for saving trajectory information in an adjoint code. In the first one, parts of the trajectory are locally recomputed and stored using local variables, while in the second one, the trajectory is stored on a last-in-first-out stack. These options are discussed in [4]. Nevertheless, the size of the trajectory of a model sometimes exceeds the memory capacities of a given computer.

2.3. The meteorological model Meso-NH. As an example, we study the trajectory problem on the nonhydrostatic meso-scale meteorological model Meso-NH (30,000 lines) developed by the Centre National de Recherches en Météorologie and the Laboratoire d'Aérologie of Toulouse (France). In a summary, the governing equations of the atmosphere are discretized using explicit finite difference methods in both

time and space. The time-stepping is coded using a leap-frog scheme. An extensive description of the Meso-NH package is given in [15] and [16].

Meso-NH has been differentiated [2] with respect to the state variables. Defined on a large spatial grid, these are wind velocity components, dry potential temperature, turbulent kinetic energy, six mixing ratios of water, N passive source terms (chemical pollutants, for example), and pressure.

The differentiation work is carried out using the automatic differentiation tool Odyssée [5] and appropriate posttreatments [2]. The resulting codes are gathered in the MesODiF package [1]. The management of the trajectory is realized according to the following two principles.

- Only state variables are stored at the beginning of each time-step (in the main routine).
- A few local variables are recomputed at each iteration.

The original adjoint code, called *OAC-MesODiF*, performs a full forward sweep before the full reverse one. The storage is realized on a fast disk of 1.2 giga-word space.

A first example is chosen in order to evaluate the performances of *OAC-MesODiF*. The two-dimensional (2D) simulation trial takes place on a computational domain (180 km$\times$ 15 km) containing a bell shaped mountain with an altitude of 500 m and a base of 10 km. The domain is discretized by 90 points in the horizontal plane (x-axis) that are duplicated 63 times in the vertical grid (z-axis). One notices that the same code Meso-NH is used to perform 2D and three-dimensional (3D) simulations. This is why three grid points are used in the y-direction even though one deals with a 2D simulation. One calculates wind components and temperature only. The physical duration T of the simulation is 2,000 seconds with a time-step of 20 seconds. Finally, the state vector is represented using 62,000 reals in that simulation.

Table 2.1 shows measurements of the execution time for an evaluation of *OAC-MesODiF*.

TABLE 2.1
Runtime measurements and ratios of both Meso-NH and OAC-MesODiF.

	Original code Meso-NH	Adjoint code *OAC-MesODiF*
CPU time	12.97	39.53
Ratio	1	3.05
Theoretical bound for the ratio	1	5

As far as computational time is concerned, *OAC-MesODiF* is efficient since the ratio between the CPU time of *OAC-MesODiF* and the CPU time of the original model, rather equal to 3, is lower than the theoretical bound. Such performances result from the trajectory management we have implemented. The other side of the coin is that this management induces the construction of a trajectory that takes up 20% of the 1.2 giga-words of memory allowed by the computer we use.

2.4. Current meteorological simulations and trajectory problems. Usually the size of the trajectory is small when the original code is such that the number of statements (written as a straight-line program) and the number of iterations are small quantities. Iterative schemes do not always satisfy these constraints since the size of the trajectory strongly depends on the number of iterations. One may distinguish evolutionary problems that have predicted costs, since the number of iterations

is known, from fixed point algorithms like conjugate gradient or quasi-Newton methods. For that second class the number of iterations, related to the verification of a stopping criterion, is unknown. This paper is devoted only to evolutionary problems. The number of iterations is denoted by P.

A set of trial simulations with a physical meaning [12] is given in Table 2.2.

TABLE 2.2
Trial simulations for a meso-scale model.

Simulation	Discretization		Memory	CPU
	# grid points	# time-steps	mega-words	seconds
Mountain waves (2D)	$90 \times 3 \times 63$	3000	2.4	167
Mountain waves	$180 \times 3 \times 63$	100	3.3	200
Mountain waves (3D)	$128 \times 96 \times 180$	889	110	7400
Baroclinic waves	$96 \times 48 \times 33$	4320	8.5	1886

Memory information (given in the 4th column) takes into account both state variables and local variables required to perform one time-step. For the sake of simplicity, one assumes that this amount of memory is used at each iteration even though local variables could be counted once.

Multiplying the amount of memory by the number of time-steps indicates that trajectories are sometimes larger than the upper limit of 1.2 giga-words. Thus *OAC-MesODiF* cannot be run.

3. Checkpointing algorithms. Many problems in computer sciences or numerical analysis may be tackled by the so-called basic idea of "divide and conquer." This applies to the evaluation of adjoint codes: one divides the computations (i.e., the trajectory), and one performs recalculations into adjoint codes.

Griewank [7] proposed to save the state of the system from time to time during runs of the original code. These are called *checkpoints*, and they allow for the computation of parts of the trajectory without systematically coming back to the initial point. In the following, the word "checkpoint" refers to the moment it is set, the recorded data, and the number of the checkpoint. Checkpoints are stored on a stack in a last-in-first-out manner. Forward sweeps and reverse sweeps are then done, part by part, from checkpoints. A simple checkpointing algorithm reads as follows.

run the original code and **store** some checkpoints
(as described in sections 3.3.2 (*Periodis* scheme) and 3.4 (*Twice* scheme))
for all the checkpoints (taken in the reverse order)
restore the checkpoint
perform a forward sweep from the checkpoint to the previously removed one (or to the Pth iteration)
perform a reverse sweep down to the checkpoint
remove the checkpoint from the stack.

Checkpoints can be set at any point in a computational graph, but the remaining problem is to know the active variables to be taped. Checkpointing is easy to implement for time-stepping problems, where a natural point is the beginning of a time-step. It is even easier for the adjoint of Meso-NH because the trajectory management already defines points (section 2.3) where state variables values are stored.

Using checkpointing algorithms introduces *extra forward steps* of the original code. Regarding the computational time, this number of iterations N_{it} is chosen as a mea-

sure for the performances of the algorithms. Forward sweeps that integrate the model and store the trajectory are not counted in N_{it} since they are always performed once during the adjoint code evaluation. One notices that checkpoints are not set during a forward sweep; they are only set during evaluations of the original code. A second measure is concerned with the maximal number K of checkpoints that need to be kept in memory at any time.

From the user's point of view, the choice of a checkpointing scheme essentially depends on the particular code and the particular computer he deals with. An interesting problem is then to find a partition for the calculations that minimizes the number of extra iterations and/or the number of checkpoints. A large variety of schedules is discussed in the following; some of them are derived from Griewank's *Revolve* [7].

3.1. Notations. The main iterative loop of the original model one deals with solves an explicit finite difference scheme. For the sake of simplicity, one assumes that the P steps of the loop have the same execution time and generate a trajectory of the same size S.

A trajectory decomposes into two parts: on one hand state variables, and on the other hand, local variables. The variables of the second part are local to an iteration, i.e., the amount of memory needed for their storage is independent of the number of iterations. For the sake of simplicity, in complexity estimates, trajectories are nevertheless assumed to be of size S.

One denotes by

- R_O the execution time used to perform one time-step of the original model,
- R_T the execution time used to perform one time-step during a forward sweep,
- R_A the execution time used to perform one time-step during a reverse sweep, and
- R_C the execution time and S_C the amount of memory required to store a checkpoint.

One observes that the storage of a checkpoint is generally far less time-consuming than the execution of one time-step of a reverse sweep ($R_C << R_A$). Furthermore, S_C is usually much smaller than S because the trajectory generated during one iteration includes many intermediate values that are saved for the reverse sweep.

The adjoint code of Meso-NH behaves differently since one may assume that $S = S_C$. This is due to the fact that state variables are read at the beginning of each time-step of the reverse sweep, a few local recomputations depending on these variables (section 2.3) being added to perform the backward integration.

In order to describe the schemes, one introduces the following three integer parameters:

- the maximal number K of checkpoints in memory,
- the maximal number times τ any particular time-step is repeated, and
- the maximal length ν of a forward sweep during which no checkpoints are set.

3.2. Griewank's *Revolve* routine. Griewank [7] proposes to split the evaluation of the adjoint code with respect to a binomial law β that depends on parameters

K and τ:

$$\begin{aligned}\beta(K,\tau) &= \frac{(K+\tau)!}{K!\tau!} \\ &= \frac{(K-1+\tau)!}{(K-1)!\tau!} + \frac{(K+\tau-1)!}{K!(\tau-1)!} = \beta(K-1,\tau) + \beta(K,\tau-1).\end{aligned} \tag{3.1}$$

The checkpointing schedule *Revolve* may be written as follows:

compute or choose adequate parameters $K \geq 0$ and $\tau \geq 0$
CALL *Revolve* $(1, P, K, \tau)$

RECURSIVE ROUTINE *Revolve* (B, E, k, t)
IF ($(E - B > t)$ and $(k > 0)$) **THEN**
    store the checkpoint at time B (if it differs from the current checkpoint)
    run the original model from B to $I = B + \beta(k, t-1)$
    CALL *Revolve* $(I, E, k-1, t)$
    restore the checkpoint (time B)
    CALL *Revolve* $(B, I, k, t-1)$
ELSE
    restore the checkpoint (time B)
    compute trajectories and **evaluate** the adjoint from E down to B
    remove the checkpoint (time B) from the stack
END IF

The last-in-first-out process is well adapted to this recursive scheme:

- one stores a checkpoint if it differs from the current one,
- one always reads the last checkpoint of the stack, and
- one removes the last checkpoint after the adjoint evaluation.

One notices that the checkpoint at time B is stored once and restored twice at least. In fact, the evaluation of the adjoint code may require several readings of the same checkpoint, especially when trajectories are locally recomputed.

Assuming that the underlying iteration is a single step recurrence ($\nu = 1$), one proves the following results.

THEOREM 3.1. *Let P be the number of forward iterations.*

1. *The adjoint code is calculable by Revolve if and only if parameters $K \in \mathbb{N}^*$ and $\tau \in \mathbb{N}^*$ are such that*

$$P \;\leq\; \beta(K,\tau).$$

2. *The minimal number of forward steps that are required to revert a sequence of P time-steps storing up to K checkpoints at any time satisfies*

$$N_{it} = \tau P - \beta(K+1, \tau-1), \tag{3.2}$$

where τ is chosen such that

$$\beta(K, \tau-1) < P \leq \beta(K,\tau). \tag{3.3}$$

3. *From among the checkpointing schedules satisfying the first two assertions, there exists a scheme that minimizes the number of checkpoints settings.*

Proof. Assertions of Theorem 3.1 are proved in [7], [11], and [9]. ∎

One notices that the scheme described in [11] is similar to *Revolve*. It has been incorporated into the automatic differentiation tool Odyssée [5] for generating adjoint codes.

The following temporal and spatial costs, $R_{Revolve}$ and $S_{Revolve}$, for the evaluation of the adjoint code incurred

$$\left\{ \begin{array}{l} R_{Revolve} \simeq N_{it} R_O + P R_T + P R_A, \\ S_{Revolve} = \nu S + K S_C. \end{array} \right. \tag{3.4}$$

For the choice $K \simeq \tau$ one deduces from Stirling's formula (see [7]) that the temporal complexity, which is related to N_{it}, is of the order of $P \log(P)$. The corresponding spatial complexity is of the order of $\log(P)$.

Leap-frog schemes are not single step recurrences. Hence *Revolve* as described in [7] cannot be fairly compared to other schemes [1]. Fortunately, the assumption "$\nu = 1$" has been recently relaxed in [19] and [10]. To prove the optimality of *Revolve* in such a case, A. Walther [19] introduces "mega-steps" that treat a few successive iterates as a unique one.

3.3. Other checkpointing algorithms. The iterative problem with P iterations is split into $K \geq 1$ subproblems $[t_{k-1}, t_k]_{k=1,\ldots,K}$ of $\nu_k \in \mathbb{N}$ time-steps that are supposed to be performed in one sweep. The partition is assumed to satisfy

$$\left\{ \begin{array}{ll} 0 = t_0 < t_1 < \cdots < t_{K-1} < t_K = P, & \\ t_k - t_{k-1} = \nu_k & \forall k \in \{1, \ldots, K\}, \\ \nu_k \geq \nu_{k+1} & \forall k \in \{1, \ldots, K-1\}, \\ \displaystyle\sum_{k=1}^{K} \nu_k = P. & \end{array} \right. \tag{3.5}$$

3.3.1. Computation from the initial state (*FISC*). For the sake of simplicity, intervals are supposed to be of the same length ν, i.e.,

$$t_k = k\nu \quad \forall\ k \in \{0, \ldots, K\}.$$

The parameter K must obviously be chosen such that $P/\nu \leq K < P/\nu + 1$. A unique checkpoint, the initial state, is saved once. The original code reads the checkpoint and runs without recording intermediate variables until the kth interval. A forward sweep is then executed so as to generate the trajectory from iteration t_k to iteration t_{k+1}; this trajectory is immediately used for the corresponding reverse sweep. The process loops until the adjoint system is completely evaluated. When ν is equal to 1, this schedule corresponds to *Revolve* $(1, P, 1, P)$: *FISC* is the upper bound of *Revolve* in terms of time requirement and its lower bound in terms of memory requirement.

In Figure 3.1, the number of extra forward steps is clearly equal to the area (computed using a rectangle method) of the triangle limited by forward and reverse sweeps. This algorithm then induces $N_{it} = K(K-1)\nu/2$ extra forward iterations: the temporal complexity is of the order of P^2. As a consequence, the expensive FISC scheme must be avoided when the number of intervals is large. However, this simple scheme could be used for a partition containing a few intervals (typically, 2 or 3).

3.3.2. Periodic distribution of checkpoints (*Periodis*). As in section 3.3.1, the iterative problem is divided into K subproblems of length ν. A first forward integration of the P iterations sets regularly the checkpoints that are used in reverse

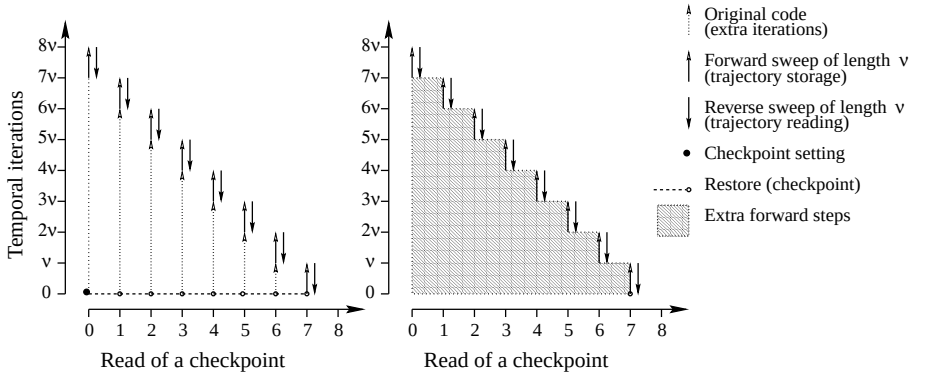


FIG. 3.1. *Computation from the initial state (FISC).*

order to evaluate, part by part, the adjoint code. This process is drawn in Figure 3.2. If the amount of memory is sufficiently large, the original problem is solved twice:

- once to set the checkpoints (original code),
- once for the construction of each of the K parts of the trajectory (forward sweeps).

This two levels scheme, which is called *Periodis*, induces $N_{it} = (K-1)\nu \simeq P$ extra forward steps. Such a partition is well adapted to simulations in which computational time is the main concern. On the other hand, *Periodis* requires a potentially large amount of memory which varies linearly with respect to the number of iterations P. This constraint is relaxed in section 3.4.

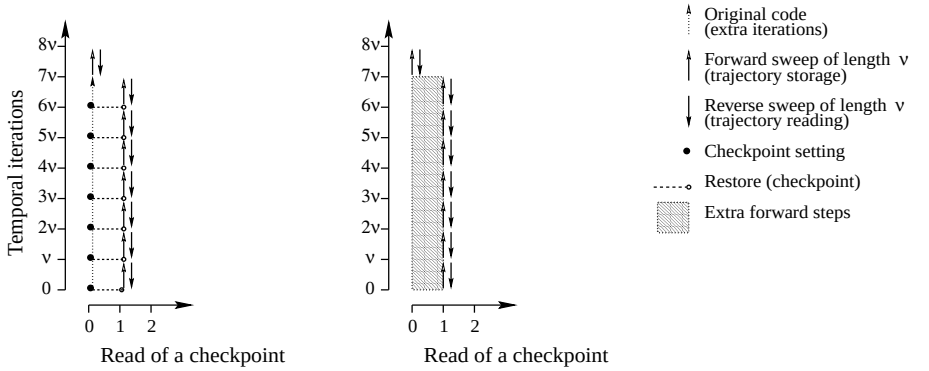


FIG. 3.2. *Periodic distribution of checkpoints (Periodis).*

When $\nu = 1$, this schedule is equivalent to *Revolve* $(1, P, P, 1)$. In that case, checkpoints are stored and restored once at every time-step. Afterward small computations are performed to supply the remaining part of the trajectory (local variables). This user-friendly strategy is used at the European Centre for Medium-Range Weather Forecast (ECMWF, Reading, UK) for the evaluation of the adjoint code of the operational meteorological model IFS (global circulation model). Although it uses a large amount of memory, the top priority at the European Center is given to the CPU time because the purpose of any operational model is to provide a forecast in time.

3.4. Distribution through arithmetic regression (*Twice*). The size of the checkpoint stack and the size of the trajectory one can store between two successive checkpoints may be managed with respect to the fixed amount of memory Q of a computer.

As a consequence, one chooses parameters K and ν such that KS_C and νS are both smaller than Q. This is not always possible. A solution is then to partition the calculations such that ν_k varies with respect to the number of checkpoints that are already stored at time t_k. Checkpoints are no longer assumed to be equidistant.

One searches couples $(k, \nu_k)_{k=1,\dots,K}$ such that

$$\begin{cases} \nu_k \geq 1 \quad \forall\ k \in \{1, \dots, K\}, \\ \nu_k S + k S_C \leq Q. \end{cases} \tag{3.6}$$

Once the partition is determined, *Twice* runs as *Periodis*: the original model is integrated twice. An integration of the original model is done to store the checkpoints that are then used in reverse order to evaluate, part by part, the adjoint code. Drawn in Figure 3.3, the *Twice* process is easy to implement. It may be viewed as *Revolve* $(1, P, K, 2)$, which first sets the K checkpoints and then performs forward sweeps. The amount of memory released when removing a checkpoint is reused for trajectories computations, i.e.,

$$\nu_k \geq \nu_{k+1} \quad \forall\ k \in \{1, \dots, K-1\}.$$

In some sense *Twice* is the lower bound of *Revolve* in terms of computational time and its upper bound in terms of memory requirement.

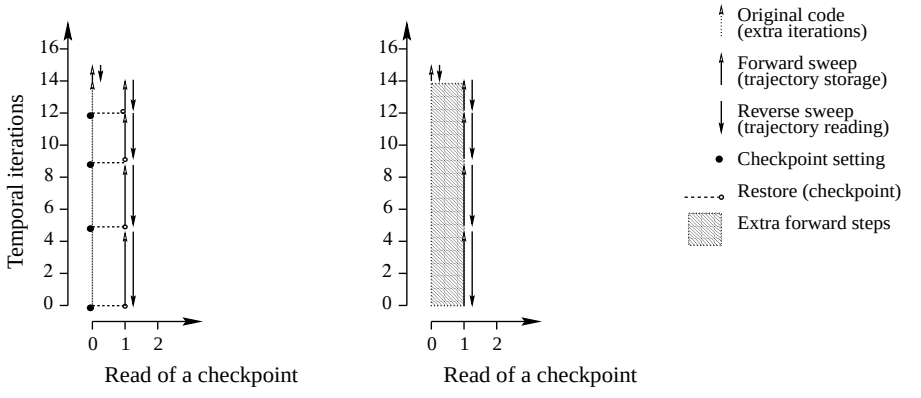


FIG. 3.3. *Reverse integration in a constant space (Twice)* $\{P = 15,\ K = 4,\ and\ \tau = 1\}$.

When $S_C = S$, which is true for Meso-NH (section 3.1), one proves the following.

THEOREM 3.2. *Let P be the number of forward iterations, and let Q be the quantity of available memory.*

1. *The adjoint code is calculable by Twice if and only if integers ν and K are positive and such that*

$$\begin{cases} P \leq \dfrac{\nu(\nu-1)}{2}, \\ \nu S < Q, \\ \dfrac{2\nu - 1 - \sqrt{(2\nu-1)^2 - 8P}}{2} \leq K \leq \nu - 1. \end{cases} \tag{3.7}$$

2. *The computational costs are equal to*

$$\text{(3.8)} \qquad \begin{cases} N_{it} = \displaystyle\sum_{k=1}^{K-1} \nu_k \simeq P, \\ R_{Twice} = N_{it} R_O + P(R_T + R_A) + K R_C, \\ S_{Twice} = \nu S \simeq Q. \end{cases}$$

Proof. Assume that $S = S_C$. Let ν be the largest positive integer such that

$$\nu < \frac{Q}{S}.$$

If $\nu \geq P$, then the adjoint code is computable without the use of a checkpointing algorithm. Otherwise, the construction of ν_k and t_k according to (3.6) obviously implies that

$$\nu_k = \nu - k \quad \forall\ k,\ 0 \leq k < \nu.$$

This means that intervals $[t_{k-1}, t_k]$ (for all $k,\ 0 \leq k < \nu$) are such that

$$\text{(3.9)} \qquad \begin{cases} t_0 = 0, \\ t_k = t_{k-1} + \nu - k = \displaystyle\sum_{l=1}^{k-1} \nu_l + \nu_k = \sum_{l=1}^{k} \nu_l \qquad \forall\ k,\ 0 \leq k < \nu. \end{cases}$$

One observes that *Twice* can perform the backward iterations if and only if there always exists a sufficient amount of memory for the storage of a trajectory of length ν_k, i.e., ν_k needs to be a positive integer for all $k \in \{0, \ldots, \nu - 1\}$. Thus a sequence of P time-steps can be reverted if and only if ν satisfies

$$\text{(3.10)} \qquad P \leq \sum_{l=1}^{\nu-1} (\nu - l) = \sum_{l=1}^{\nu-1} l = \frac{\nu(\nu-1)}{2}.$$

According to ν, the minimum number of checkpoints $K < \nu$ has to satisfy

$$\text{(3.11)} \qquad P \leq \sum_{l=1}^{K} \nu_l = \sum_{l=1}^{K} \nu - l = K\nu - \frac{K(K+1)}{2},$$

i.e.,

$$\text{(3.12)} \qquad \frac{2\nu - 1 - \sqrt{(2\nu-1)^2 - 8P}}{2} \leq K \leq \nu - 1.$$

One derives assertion 2 from the previous results. ∎

It follows that the temporal complexity is independent of P as for *Periodis*, whereas the spatial complexity decreases to $O(\sqrt{P})$.

3.4.1. Bisection method (*Bisection*). For many numerical purposes such as a search in an ordered table, a root finding, or the "all-to-all" parallel communication scheme, it is hard to do better than bisection. For example, it will find the place of any element, in a table of size P, in about $\log_2 P$ tries. One may use a bisection scheme (Figure 3.4) for adjoint calculations, too.

In this section, checkpoints are at least lagged by ν iterations. The number K of checkpoints which are required to revert the sequence of P iterations, and the number of extra forward iterations, are such that

$$\left\{ \begin{array}{l} \log_2\left(\dfrac{P}{\nu}\right) \leq K < \log_2\left(\dfrac{P}{\nu}\right) + 1, \\ N_{it} = \dfrac{KP}{2}. \end{array} \right. \tag{3.13}$$

It follows that the temporal complexity of the *Bisection* schedule is of the order of $P\log(P)$ for a spatial complexity of the order of $\log(P)$. The complexities are equivalent to those of *Revolve*. Nevertheless, due to constants, *Revolve* is better than *Bisection* as far as actual extra forward steps are considered.

A bisection routine may be written as follows.

RECURSIVE ROUTINE *Bisection*(B, E, ν)
IF $(E - B > \nu)$ **THEN**
  $M = (E - B)/2$
  store the checkpoint at time B (if it differs from the current checkpoint)
  run the original model from B to M
  CALL *Bisection*(M, E, ν)
  restore the checkpoint (time B)
  CALL *Bisection*(B, M, ν)
ELSE
  restore the checkpoint (time B)
  compute trajectories and **evaluate** the adjoint from E down to B
  remove the checkpoint (time B) from the stack.
ENDIF

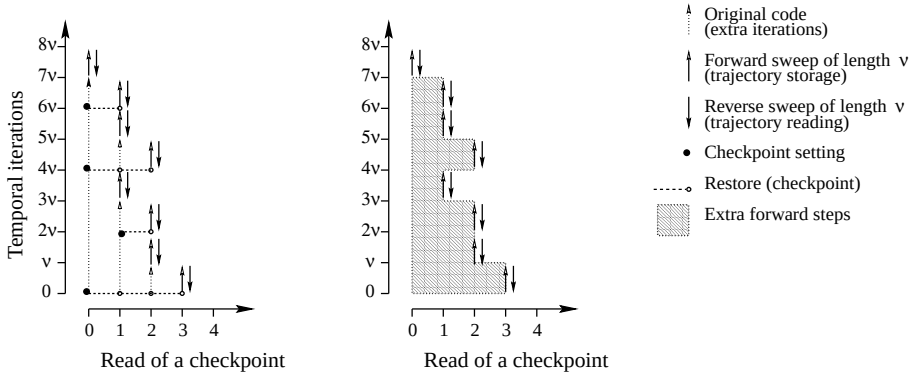


FIG. 3.4. *Bisection scheme.*

4. Leap-frog schemes and differentiation. Let F be a nonlinear differentiable function. The staggered leap-frog method, which allows for solving the equation

$$\frac{\partial X}{\partial t} = F(X) \text{ in } \Omega \times (0, T),$$

is defined as follows. Use the values of X^p at iteration p to compute $(F_j^p)_j = F(X^p)$ on the spatial grid that discretizes Ω. Then compute new values X^{p+1} using the time

centered values of the function F:

$$\left\{\begin{array}{l} X^0 \text{ is given,} \\ X_j^1 - X_j^0 = \Delta t F_j^0, \\ X_j^{p+1} - X_j^{p-1} = 2\Delta t F_j^p \qquad \forall p \in \{1, \ldots, P-1\}. \end{array}\right. \tag{4.1}$$

In the former equations Δt denotes the length of the time-step.

A convenient way to code a leap-frog scheme is to use three variables that, respectively, contain the values of the solution at iterations $p-1$, p, and $p+1$. These variables are denoted, respectively, by X_M, X_T, and X_R. When $p \in \{1, \ldots, P-1\}$, (4.1) may be solved using the following sequence of statements:

$$\left\{\begin{array}{ll} X_R = 2\Delta t F(X_T) + X_M, & \text{computation of the model,} \\ X_M = X_T, & \text{temporal advances,} \\ X_T = X_R. & \end{array}\right. \tag{4.2}$$

The computation of X_R requires the knowledge of X_M and X_T. In particular, a correct restart of a simulation from a checkpoint implies that one has saved the state variables at the previous two iterations.

4.1. Differentiation of a leap-frog scheme. The adjoint model that corresponds to the sequence of statements (4.2) is written as follows:

$$\left\{\begin{array}{l} \widehat{X_R} = \widehat{X_R} + \widehat{X_T}, \\ \widehat{X_T} = 0, \\ \widehat{X_T} = \widehat{X_T} + \widehat{X_M}, \\ \widehat{X_M} = 0, \\ \widehat{X_M} = \widehat{X_M} + \widehat{X_R}, \\ \widehat{X_T} = \widehat{X_T} + 2\Delta t \left[\dfrac{\partial F}{\partial X_T}(X_T)\right]^* \widehat{X_R}, \\ \widehat{X_R} = 0. \end{array}\right. \tag{4.3}$$

Let us assume that the nonlinear term $F(X)$ at iteration p depends only on values of X^p, which is locally equal to X_T in (4.2). Variable X_M enters linearly in the right-hand side of the first equation of (4.2). Therefore, the useful trajectory is limited to X_T. As a consequence, the memory required for the storage of the trajectory (X_T) is twice smaller than the memory required for the storage of the state solution (X_T, X_M) at a checkpoint.

Remark. Systems (4.2) and (4.3) were introduced in order to explain the underlying property of the adjoint of some leap-frog schemes. However, the useful trajectory is limited to X_T even if function F is nonlinear in both X_M and X_T. This is due to the fact that temporal advances assign X_M to X_T, and one can read the trajectory in advance to get X_M: storing X_M and X_T is a redundant task.

4.2. Leap-frog scheme of Meso-NH. In meteorological and oceanic models, time-stepping is achieved by a basic leap-frog scheme associated with an Asselin filter applied at each time-step in order to avoid time splitting. Variables X_M and X_T are filtered using linear combinations of values of X_M, X_T, and X_R:

$$\left\{\begin{array}{ll} X_R = 2\Delta t F(X_T) + X_M, & \text{computation of the model,} \\ X_M = \eta X_M + (1-2\eta) X_T + \eta X_R, & \text{temporal advance with a filter,} \\ X_T = X_R. & \end{array}\right. \tag{4.4}$$

The coefficient η is positive real and $\eta < 0.5$. Checkpoints are again formed by values of X_M and X_T. On the other hand, trajectories require

- X_T if function F is nonlinear in X_T only, and
- (X_M, X_T) if function F is nonlinear in the two variables. This occurs when a numerical horizontal diffusion is applied on the fluctuations of state variables.

The trial simulation presented in section 2.3 satisfies $F = F(X_T)$. Let U, W, Θ, and Φ denote the variables used for the wind fields, the temperature, and the pressure, respectively. These are 3D arrays. Indexes $_M$, $_T$, and $_R$ are used to distinguish iterates. Since the model equations are nonlinear in $X_T = \{U_T,\ W_T,\ \Theta_T\}$ only, the knowledge of X_T is sufficient to evaluate one time-step of the adjoint integration. On the other hand, a restart of the Meso-NH model from a given checkpoint requires the knowledge of the two sets of variables $X_M = \{U_M,\ W_M,\ \Theta_M,\ \Phi_M\}$ and X_T. As a consequence, one has the following ratio:

$$\frac{S_C}{S} = \frac{\text{memory for one checkpoint}}{\text{memory for a trajectory of one step}} = \frac{\#X_M + \#X_T}{\#X_T} = \frac{7}{3},$$

where $\#X_M$ is the cardinal number of the set X_M.

Remark. The variable Φ_T does not belong to X_T because of the anelastic approximation used to solve the pressure equation.

When the code is governed by a leap-frog scheme such as defined before, the size of the useful trajectory may be reduced up to a factor 2. Larger simulations may be tackled without a checkpointing scheme. Otherwise, *Periodis* and *Twice* can easily take into account this property. This allows us to store more time-steps between two successive checkpoints and to run larger simulations. *Revolve* also takes into account the gain in memory related to leap-frog schemes as soon as one implements the original model with mega-steps of length $\nu = 2$ (see [19] for details).

Whatever the chosen scheme is, the gain in memory can be changed into a gain in CPU time since one may run simulations twice longer in terms of iterations.

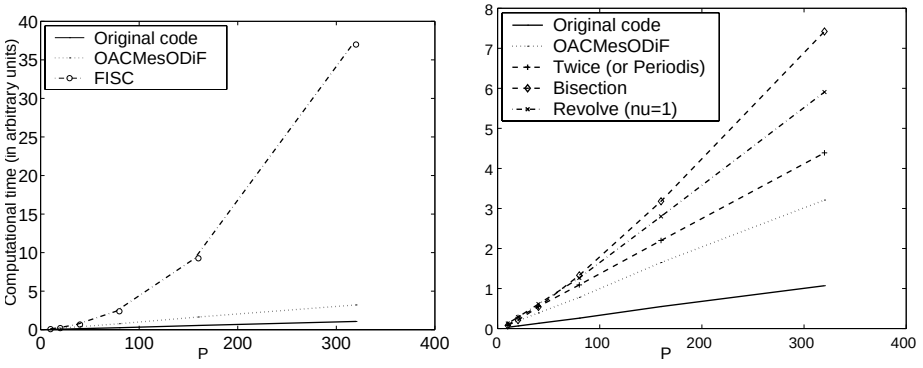
5. Numerical results. The Meso-NH model is used to compare the schemes. The results indicate only the general behavior in CPU time of adjoint computations performed with checkpointing schemes. This choice is relevant when one deals with an optimization process that integrates the adjoint code several times. The reader is referred to [9] for details on the memory requirement of *Revolve*.

5.1. Temporal cost of checkpointing schemes. The following two graphs indicate the computational time in an arbitrary unit spent to evaluate the adjoint code of Meso-NH with respect to the number of iterations $P \in \{10, 20, 40, 80, 160, 320\}$. Checkpoints are set every ν =10 steps for *FISC*, *Periodis*, and *Bisection*.

In order to allow fair comparisons, the same quantity of memory is used for *Bisection* ($\nu_B = 10$, $K_B = \log_2(P/\nu)$) and *Revolve* ($\nu_R = 1$, $K_R = \nu_B + K_B$). *Twice* uses the same amount of memory as *Bisection* as far as $P \leq 160$, whereas $K_T = 18$ checkpoints and a maximal length $\nu_T = 36$ are required to reverse 320 iterations.

The quadratic behavior of *FISC*, computations from the initial state, appears clearly in Figure 5.1. In Figure 5.1 one observes the affine behavior of the two levels scheme *Twice*. The numbers of iterations of *Periodis* and *Twice* differ only by a few units. They almost lead to the same curve because computations were done using a fast access disc. As expected, one obtains the CPU time of these schemes when one adds the CPU time of *OAC-MesODiF* to the CPU time of the original model.

Revolve results are almost similar to those obtained with *Periodis* when using mega-steps of length $\nu = 2$ that are adapted to the leap-frog process. One notices



(a) Quadratic behavior of FISC.

(b) Schemes with a linear or logarithmic behavior.

FIG. 5.1. *Computational time of checkpointing algorithms.*

that the improvement proposed in [19] is important since the results are really better than those obtained using *Revolve* with $\nu = 1$. From a theoretical point of view, the logarithmic growth applies only when parameters K and τ of *Revolve* are chosen according to formula (3.3). One notices that choosing a larger parameter K allows one to reduce the number of extra iterations.

The numerical logarithmic behavior of the bisection scheme is in good agreement with the theory. One observes that *Revolve* ($\nu = 1$) reaches a better compromise between time and memory requirements than *Bisection* ($\nu = 10$). This is a general result (depending on constants) when comparing the two schemes.

5.2. Trial meteorological simulations. Each of the trial meteorological simulations described in section 2.3 can be run with a suitable algorithm. The priority is again given to the CPU time at the expense of the use of a large amount of memory (1.2 giga-words). In Table 5.1 the performances are measured with respect to the number of extra forward steps N_{it}.

TABLE 5.1
Checkpointing scheme for large meteorological simulations.

Simulation	P	Memory (S_C) mega-words	Scheme	(ν, K)	N_{it}
Mount. waves (2D)	3000	2.4	*Twice*	(200,111)	2889
Mount. waves	100	3.3	not necessary		0
Mount. waves (3D)	889	110	*Revolve* ($\nu = 1$)	(4,10)	3192
Baroclin. waves	4320	8.5	*Twice*	(140,140)	4180

6. Conclusions. *Periodis* and *Twice* schemes are easy to implement and are good in terms of computational time since the evaluation of the adjoint code requires only two integrations of the original code: one for setting the checkpoints and another one for the construction of each of the K parts of the trajectory.

It appears that the evaluation of the adjoint code of the meteorological model Meso-NH can usually be realized with the *Twice* scheme. This is especially true when the iterative scheme of the model is governed by a leap-frog method as described in section 4. However, one notices that *Revolve* [19] always offers an optimal solution.

In the examples it appears that the ratio between the CPU time of the calculation of one gradient with the adjoint code of Meso-NH and the CPU time of the evaluation of the original model is equal to

- 3 when *OAC-MesODiF* applies (the amount of memory is sufficient to store all the trajectory),
- 4 when one has enough memory to run *Twice* or *Revolve* $(1, P, K, 2)$ $(\nu = 2)$,
- 5.6 for the 3D mountain waves simulation described in Tables 2.2 and 5.1 when using Griewank's *Revolve* subroutine $(\nu = 1)$.

In conclusion, checkpointing algorithms allow for gradient computations of applications that require the storage of large trajectories.

REFERENCES

[1] I. Charpentier, *The MesODiF package for gradients computations with the atmospheric model Meso-NH*, in Proceedings of the Air Pollution and Modeling Systems Conference (Paris, 1998), Environmental Modeling and Software, to appear.

[2] I. Charpentier and M. Ghémirès, *Efficient adjoint derivatives: Application to the atmospheric model Meso-NH*, Optim. Methods Softw., 13 (2000), pp. 35–63.

[3] J. Derber, *Variational four-dimensional analysis using quasi-geostrophic constraints*, Mon. Wea. Rev., 115 (1987), pp. 998–1008.

[4] C. Faure and I. Charpentier, *Comparing strategies for coding adjoints*, Scientific Programming, to appear.

[5] C. Faure and Y. Papegay, *Odyssée Version* 1.6, *the User's Reference Manual*, Rapport technique INRIA RT-0211, INRIA, Le Chesnay, France, 1997.

[6] A. Griewank, *On automatic differentiation*, in Mathematical Programming: Recent Developments and Applications, M. M. Iri and K. Tanabe, eds., Kluwer Academic Publishers, Dordrecht, The Netherlands, 1989, pp. 83–108.

[7] A. Griewank, *Achieving logarithmic growth of temporal and spatial complexity in reverse automatic differentiation*, Optim. Methods Softw., 1 (1992), pp. 35–54.

[8] A. Griewank, *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*, Frontiers Appl. Math. 19, SIAM, Philadelphia, 2000.

[9] A. Griewank and A. Walther, *Revolve: An Implementation of the Checkpointing for the Reverse or Adjoint Mode of Differentiation*, preprint IOKOMO-04-1997, Technische Universität Dresden, Dresden, Germany, 1997; ACM Trans. Math. Software, to appear.

[10] A. Griewank and A. Walther, *Optimal Program Execution Reversal*, in preparation.

[11] J. Grimm, L. Pottier, and N. Rostaing-Schmidt, *Optimal time and minimum space-time product for reversing a certain class of programs*, in Computational Differentiation: Techniques, Applications, and Tools, M. Berz, C. Bischof, G. Corliss, and A. Griewank, eds., SIAM, Philadelphia, 1996, pp. 95–106.

[12] J. P. Lafore, J. Stein, N. Ascencio, P. Bougeault, V. Ducrocq, J. Duron, C. Fischer, P. Héreil, P. Mascart, V. Masson, J. P. Pinty, J. L. Redelsperger, E. Richard, and J. Vilà-Guerau de Arellano, *The Meso-NH atmospheric simulation system. Part* I: *Adiabatic formulation and control system*, Ann. Geophysicae, 16 (1998), pp. 90–109.

[13] F.-X. Le Dimet and O. Talagrand, *Variational algorithms for analysis and assimilation of meteorological observations: Theoretical aspects*, Tellus, 38A (1986), pp. 97–110.

[14] J.-L. Lions, *Optimal Control of Systems Governed by Partial Differential Equations*, Springer-Verlag, Berlin, 1971.

[15] J. P. Lafore, J. Stein, N. Ascencio, P. Bougeault, V. Ducrocq, J. Duron, C. Fischer, P. Héreil, P. Mascart, V. Masson, J. P. Pinty, J. L. Redelsperger, E. Richard, and J. Vilà-Guerau de Arellano, *The Meso-NH Atmospheric Simulation System: Scientific Documentation*, Météo-France and CNRS, Toulouse, France, 1995.

[16] J. P. Lafore, J. Stein, N. Ascencio, P. Bougeault, V. Ducrocq, J. Duron, C. Fischer, P. Héreil, P. Mascart, V. Masson, J. P. Pinty, J. L. Redelsperger, E. Richard, and J.

Vilà-Guerau de Arellano, *The Meso-NH Atmospheric Simulation System: Algorithmic Documentation*, Météo-France and CNRS, Toulouse, France, 1996.

[17] J. Morgenstern, *How to compute fast a function and all its derivatives, a variation on the theorem of Baur-Strassen*, SIGACT News, 16 (1985), pp. 60–62.

[18] F. Rabier, H. Järvinen, E. Klinker, J-F. Mahfouf, and A. Simmons, *The ECMWF Operational Implementation of Four Dimensional Variational Assimilation. Part* I: *Experimental Results with Simplified Physics*, ECMWF Research Department Tech. Memo. 271, ECMWF, Reading, UK, 1999.

[19] A. Walther, *Program Reversal Schedules for Single- and Multi-Processor Machines*, Thesis, Institute of Scientific Computing, Dresden University of Technology, Dresden, Germany, 1999.

SIAM J. SCI. COMPUT.
Vol. 22, No. 6, pp. 2152–2177
© 2001 Society for Industrial and Applied Mathematics

A SEMI-LAGRANGIAN FINITE VOLUME METHOD FOR NEWTONIAN CONTRACTION FLOWS *

T. N. PHILLIPS† AND A. J. WILLIAMS‡

Abstract. A new finite volume method for solving the incompressible Navier–Stokes equations is presented. The main features of this method are the location of the velocity components and pressure on different staggered grids and a semi-Lagrangian method for the treatment of convection. An interpolation procedure based on area-weighting is used for the convection part of the computation. The method is applied to flow through a constricted channel, and results are obtained for Reynolds numbers, based on half the flow rate, up to 1000. The behavior of the vortex in the salient corner is investigated qualitatively and quantitatively, and excellent agreement is found with the numerical results of Dennis and Smith [*Proc. Roy. Soc. London A*, 372 (1980), pp. 393–414] and the asymptotic theory of Smith [*J. Fluid Mech.*, 90 (1979), pp. 725–754].

Key words. semi-Lagrangian approach, Navier–Stokes, finite volume method, contraction geometry, staggered mesh, SIMPLER algorithm

AMS subject classifications. 65N06, 65N22, 76D05, 76M25

PII. S1064827599365288

1. Introduction.

In this paper a semi-Lagrangian finite volume method is presented for solving incompressible flows of Newtonian fluids. This approach is shown to be particularly well suited to solving the Navier–Stokes equations for which the discretization of the governing equations is well known to be crucial when the flow is convection dominated. Numerical difficulties can occur for large values of the Reynolds number, a nondimensional quantity which measures the relative importance of convection compared to diffusion. Central difference schemes, which perform well for low Reynolds number flows, are prone to numerical difficulties for higher Reynolds numbers. For low Reynolds numbers central difference schemes produce a diagonally dominant system of equations which can be solved without any difficulty using standard relaxation techniques. At higher Reynolds numbers diagonal dominance is lost with the result that one can encounter problems using the same schemes. These difficulties can manifest themselves in several ways. For example, relaxation techniques may fail to converge, and if a solution is obtained for the steady problem, it may exhibit physically unrealistic oscillations.

A popular finite volume approach for overcoming the difficulties associated with the treatment of convection uses interpolation biased towards the upwind direction when the flow is convection dominated. Low-order interpolation invariably results in artificial or false diffusion being added to the scheme. The effect of this is a degradation in accuracy. Higher-order interpolation schemes such as quadratic upwind interpolation for convection kinematics (QUICK) [6] and SMART [9] increase the complexity of the scheme and create difficulties near boundaries. In the presence of high gradients, they may additionally produce overshoot or undershoot values. Nonlinear flux limiting functions may be used in conjunction with high-order upwinding

*Received by the editors January 5, 2000; accepted for publication (in revised form) November 8, 2000; published electronically April 26, 2001.
http://www.siam.org/journals/sisc/22-6/36528.html
†Department of Mathematics, University of Wales, Aberystwyth SY23 3BZ, UK (tnp@aber.ac.uk).
‡Department of Mathematics, University of Wales, Aberystwyth SY23 3BZ, United Kingdom. Present address: School of Computing and Mathematical Sciences, University of Greenwich, London SE10 9LS, United Kingdom (A.J.Williams@greenwich.ac.uk)

techniques to prevent the appearance of over- or undershoots. The underlying idea behind these techniques is to control the gradients of the computed solution through the use of total variation diminishing schemes (TVD) schemes (see Hirsch [1] for a comprehensive review of this subject). It is the treatment of convection together with the positioning of the mesh points that distinguish one finite volume method from another.

In time-dependent calculations standard finite volume schemes possess time-step restrictions due to stability. These may be more severe than the conditions imposed by accuracy considerations alone. When particle following techniques are used the stability conditions are much less restrictive. However, the unrestricted movement of the points used in Lagrangian methods, which involves following a fixed set of particles throughout the flow, introduces other difficulties. For example, a set of fluid particles which is initially regularly distributed soon becomes greatly deformed, in general, and is thus rendered unsuitable for numerical integration. Semi-Lagrangian methods avoid this difficulty while still following particles. A semi-Lagrangian method for treating the convection term, in which particles on a regular grid of points are traced backwards over a single time-step to their departure points, provides the focus of this paper. Although semi-Lagrangian finite volume methods have been developed for advection problems [22, 24, 25], this paper describes their application to problems in Newtonian computational fluid dynamics. This scheme circumvents the problems associated with upwind biased interpolation schemes, possesses less restrictive stability requirements, and combines the advantages of fixed grids inherent in Eulerian methods with modifications to the location of grid points at previous time-steps based on the Lagrangian approach.

The remaining terms in the governing equations are treated implicitly and are discretized by integrating over an appropriate control volume. The discrete equations are solved using the semi-implicit method for pressure linked equations revised (SIMPLER) [19] algorithm. Therefore, this approach may be viewed as a time-splitting scheme in which the different operators in the governing equations are discretized by appropriate techniques.

The emphasis in this paper is on the semi-Lagrangian treatment of convection. This can only be accomplished within the framework of time-splitting schemes for time-dependent problems. Therefore, the proposed semi-Lagrangian finite volume method is described for time-dependent problems even though, in this paper, it is only used as a means of reaching the steady state solution. The temporal accuracy of the scheme depends on the discretization of the characteristic paths as well as the temporal discretization of the governing equations. Since a first-order scheme is used for the latter, there is no advantage in using a higher-order scheme for the characteristic calculation. The development of high-order discretizations for all the component parts of the proposed scheme is an area of ongoing activity.

The important features of the method are illustrated on a pure convection problem before it is applied to the flow through an abruptly contracting channel in which the ratio of the channel widths before and after the contraction is 2:1. Numerical results are presented for Reynolds numbers in the range [0,1000]. The behavior of the vortex in the salient corner is investigated, and comparisons are made with other results in the literature. Smith [12] developed an asymptotic theory for the flow upstream of the contraction. Good agreement is shown with this theory for the location of the upstream separation point for Reynolds numbers greater than about 100. The size of the salient corner vortex decreases as the Reynolds number increases from 0 to around

50. As the Reynolds number is increased further, the vortex grows slowly.

This paper is organized as follows. In section 2 the governing equations and flow geometry are described. The location of the dependent variables on a staggered grid and the conventional finite volume discretization of the governing equations are described in section 3. In section 4 we describe the semi-Lagrangian treatment of the convection terms. In section 5 we show how this is incorporated into a solution procedure for solving the Navier–Stokes equations based on the SIMPLER algorithm. Numerical results and comparisons with other work are presented in section 6. Finally, concluding remarks are made in section 7.

2. Governing equations. We consider the laminar flow of an incompressible fluid of viscosity η through an abruptly contracting channel with walls at $y = \pm 1$ for $x < 0$, $y = \pm\frac{1}{2}$ for $x > 0$, and $\frac{1}{2} \leq |y| \leq 1$ for $x = 0$. A schematic diagram of the lower half of this geometry is shown in Figure 1. Upstream of the contraction we impose parabolic Poiseuille flow, and we suppose that the flow is parabolic again far enough downstream. Since the flow is symmetric about $y = 0$, it is only necessary to seek a solution for $y \leq 0$.

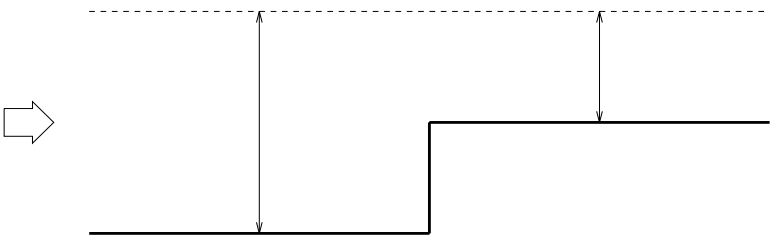


FIG. 1. *Geometry for the* 2:1 *planar contraction.*

The time-dependent Navier–Stokes equations are

$$\rho\left[\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u}.\nabla\boldsymbol{u}\right] = \eta\nabla^2\boldsymbol{u} - \nabla p, \tag{2.1}$$

$$\nabla.\boldsymbol{u} = 0, \tag{2.2}$$

where $\boldsymbol{u} = (u, v)$ is the velocity vector, p is the pressure, and ρ is the density. Nondimensional quantities are defined as

$$\boldsymbol{x}^* = \frac{\boldsymbol{x}}{L}, \quad t^* = \frac{Ut}{L}, \quad \boldsymbol{u}^* = \frac{\boldsymbol{u}}{U}, \quad p^* = \frac{p}{\rho U^2},$$

where U is a characteristic flow speed and L is a characteristic length scale of the flow. With the Reynolds number, Re, defined by

$$Re = \frac{\rho U L}{\eta},$$

we may write the Navier–Stokes equations in dimensionless form. In two-dimensional

component form they are

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} = \frac{1}{Re}\left[\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right] - \frac{\partial p}{\partial x}, \tag{2.3}$$

$$\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} = \frac{1}{Re}\left[\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right] - \frac{\partial p}{\partial y}, \tag{2.4}$$

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0, \tag{2.5}$$

where we have dropped the $*$ notation for ease of notation.

For the contraction problem the characteristic length scale (L) is chosen to be the downstream half-channel width, and the characteristic flow speed (U) is chosen to be the mean velocity across half of the downstream channel. Therefore, the boundary conditions are given by

$$\boldsymbol{u} = 0 \quad \text{on} \quad y = \begin{cases} -1 & \text{for } x \le 0, \\ -\frac{1}{2} & \text{for } x \ge 0, \end{cases}$$

$$\boldsymbol{u} = 0 \quad \text{on} \quad x = 0 \text{ for } -1 \le y \le -\frac{1}{2},$$

$$v = 0, \frac{\partial u}{\partial y} = 0 \quad \text{on} \quad y = 0,$$

$$u \to \frac{3}{2}(1 - y^2),\ v \to 0 \quad \text{as} \quad x \to -\infty,$$

$$u \to 3(1 - 4y^2),\ v \to 0 \quad \text{as} \quad x \to \infty.$$

3. Finite volume discretization. The finite volume method is generally applied to a system of differential equations written in conservation form, e.g.,

$$\frac{\partial \boldsymbol{w}}{\partial t} + \frac{\partial \boldsymbol{f}}{\partial x} + \frac{\partial \boldsymbol{g}}{\partial y} = \boldsymbol{S}, \tag{3.1}$$

where $\boldsymbol{w}$ is the vector of unknowns, $\boldsymbol{f}$ and $\boldsymbol{g}$ are vector functions of $\boldsymbol{x} = (x, y)$, $\boldsymbol{w}$, and $\nabla \boldsymbol{w}$, and $\boldsymbol{S}$ is the source term. In this paper we consider cell center finite volume methods. These methods are closely related to finite difference methods.

A grid is placed on the computational domain, and a control or finite volume is associated with each unknown on the grid. Each component equation of (3.1) is integrated over the appropriate control volume. Finite difference type approximations are then used to approximate line integrals over each side of the control volume. There is a number of ways of doing this, each leading to a numerical scheme satisfying certain properties. In the finite volume formulation mass and momentum are conserved over every control volume and therefore over the whole computational domain. The property of conservation of physical quantities, which is preserved by the discrete system, is one of the attractions and advantages of the finite volume method.

A staggered grid is used in which the different dependent variables are approximated at different mesh points (see Figure 2). This type of mesh arrangement ensures that the solution is not polluted by spurious pressure modes. On a nonstaggered mesh the familiar checkerboard mode is present.

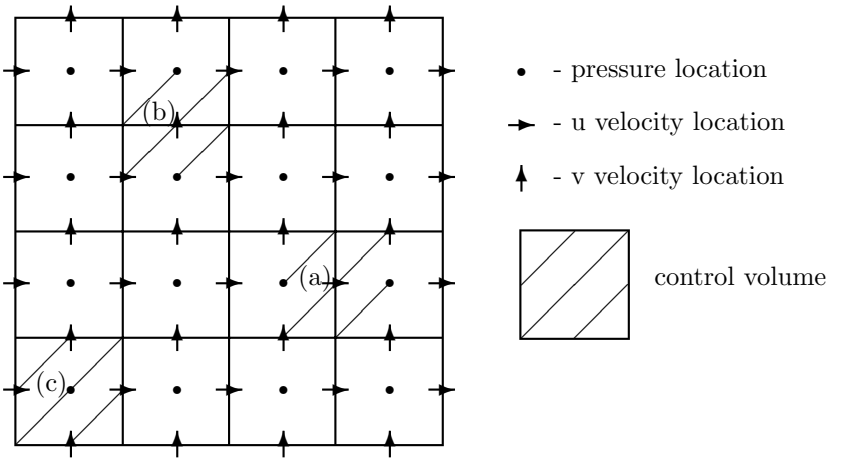


FIG. 2. *A staggered grid depicting the control volumes for* (a) *u-momentum equation,* (b) *v-momentum equation, and* (c) *continuity equation.*

TABLE 1
Definition of the symbols in the general equation (3.2).

Equation	ϕ	δ	Γ	S_ϕ
u-momentum	u	1	$(Re)^{-1}$	$-\frac{\partial p}{\partial x}$
v-momentum	v	1	$(Re)^{-1}$	$-\frac{\partial p}{\partial y}$
continuity	1	0	0	0

Each of the governing equations (2.3)–(2.5) can be cast into the general conservative form

$$\delta\frac{\partial\phi}{\partial t}+\frac{\partial}{\partial x}\left(u\phi-\Gamma\frac{\partial\phi}{\partial x}\right)+\frac{\partial}{\partial y}\left(v\phi-\Gamma\frac{\partial\phi}{\partial y}\right)=S_\phi, \tag{3.2}$$

where δ and Γ are constants and ϕ and S_ϕ are functions that are defined depending on the particular equation under consideration (see Table 1).

Equation (3.2) is discretized in time using the backward Euler method with time-step Δt to give

$$\frac{\partial}{\partial x}\left(u\phi^{n+1}-\Gamma\frac{\partial\phi^{n+1}}{\partial x}\right)+\frac{\partial}{\partial y}\left(u\phi^{n+1}-\Gamma\frac{\partial\phi^{n+1}}{\partial y}\right)=S_\phi^{n+1}-\frac{\delta}{\Delta t}(\phi^{n+1}-\phi^n), \tag{3.3}$$

where ϕ^{n+1} denotes the approximation to the variable ϕ at the $(n+1)$th time-step. Note that this semidiscrete scheme can be used as the basis for determining solutions to transient or steady problems.

The finite volume methodology proceeds by integrating (3.3) over a control volume

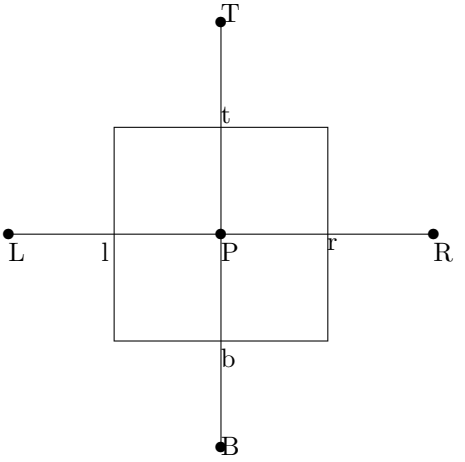


FIG. 3. *A general control volume.*

(see Figure 3, for example) and using the divergence theorem. This yields the equation

$$\begin{aligned}\int_b^t \left[\left(u\phi^{n+1} - \Gamma\frac{\partial\phi^{n+1}}{\partial x}\right)_r - \left(u\phi^{n+1} - \Gamma\frac{\partial\phi^{n+1}}{\partial x}\right)_l\right] dy \\ + \int_l^r \left[\left(v\phi^{n+1} - \Gamma\frac{\partial\phi^{n+1}}{\partial y}\right)_t - \left(v\phi^{n+1} - \Gamma\frac{\partial\phi^{n+1}}{\partial y}\right)_b\right] dx \\ = \int_b^t\int_l^r \left[S_\phi^{n+1} - \frac{\delta}{\Delta t}(\phi^{n+1} - \phi^n)\right] dxdy. \end{aligned} \tag{3.4}$$

Each of the integral terms on the left-hand side of (3.4) represents a transport by convection and diffusion through the relevant control-volume face. All schemes use central differences to approximate the diffusive flux across each face. For example, across the vertical face passing through the point r we have

$$\left(\frac{\partial\phi}{\partial x}\right)_r \simeq \frac{\phi_R - \phi_P}{\Delta x}. \tag{3.5}$$

It is the approximation of the convective flux which differentiates one finite volume scheme from another. We shall review briefly some of the common approaches, many of which are based on some form of upwinding to retain accuracy and stability when convection dominates diffusion.

A central difference approximation to the convective flux would take the form

$$(u\phi^{n+1})_r \simeq u_r^n \frac{(\phi_R^{n+1} + \phi_P^{n+1})}{2}, \tag{3.6}$$

for example. Note that the velocity component u is frozen at its value from the previous time-step. In steady calculations it would be frozen at its value from the previous iteration since iterative methods are generally used to solve the algebraic equation resulting from a finite volume discretization. Therefore, in the case of the u-momentum equation, we arrive at the discretization

$$\begin{aligned} A_P u_P^{n+1} = A_R u_R^{n+1} + A_L u_L^{n+1} + A_T u_T^{n+1} + A_B u_B^{n+1} \\ + \frac{u_P^n \Delta x \Delta y}{\Delta t} + (p_l^{n+1} - p_r^{n+1})\Delta y, \end{aligned} \tag{3.7}$$

where

$$A_R = -\frac{F_r}{2} + D_r,$$
$$A_L = \frac{F_l}{2} + D_l,$$
$$A_T = -\frac{F_t}{2} + D_t,$$
$$A_B = \frac{F_b}{2} + D_b,$$
$$A_P = A_R + A_L + A_T + A_B + F_r - F_l + F_t - F_b + \frac{\Delta x \Delta y}{\Delta t}, \tag{3.8}$$

and $F_r = u_r^n \Delta y$, $D_r = \frac{1}{Re}\frac{\Delta y}{\Delta x}$, etc.

The local truncation error for the central-difference approximation is second-order. An essential requirement for a bounded solution is that all the coefficients A_{nb}, where the subscript nb refers to the neighbors of P, should be of the same sign, usually all positive. If the resulting system of equations is diagonally dominant, then many of the standard iterative methods can be used, and convergence is guaranteed. This property is assured, and oscillatory solutions arising from negative roots to the characteristic equation are prevented, if the following conditions are satisfied:

$$A_{nb} > 0 \quad \text{and} \quad A_P \geq -(A_R + A_L + A_T + A_B). \tag{3.9}$$

However, looking at the coefficients given in (3.8), we see that in some circumstances some coefficients A_{nb} may become negative. Thus for convergence to be guaranteed we require the mesh Peclet number given by

$$Pe = \frac{F}{D},$$

where $F = u\Delta y$ and $D = \frac{1}{Re}\frac{\Delta y}{\Delta x}$, to be less than 2 in order to satisfy this boundedness criterion, i.e.,

$$u\, Re \Delta x < 2,$$

or

$$\Delta x < \frac{2}{u\, Re}. \tag{3.10}$$

Similarly, we also require

$$\Delta y < \frac{2}{v\, Re}. \tag{3.11}$$

For $Pe > 2$ the scheme may converge but to physically unrealistic solutions. This means that the central-difference method is limited to low values of Re unless the mesh is suitably refined so that conditions (3.10) and (3.11) are satisfied. Since this would be computationally expensive, the central-difference technique is not suitable for convection-dominated flow problems.

From a physical point of view central difference schemes are not suitable for convection dominated problems because the direction of the flow is not used in the derivation of the approximation. The upwind scheme takes this into account and

proposes that the value of u at the interface is equal to the value of u at the grid point on the upwind side of the face, i.e.,

$$u_r^{n+1} = \begin{cases} u_P^{n+1} & \text{if } u_r^n > 0, \\ u_R^{n+1} & \text{if } u_r^n < 0. \end{cases} \tag{3.12}$$

This approximation again leads to an equation of the form (3.7), but now the coefficients satisfy the boundedness criterion (3.9) for all Reynolds numbers. The local truncation error for the upwind scheme is first order. Since the upwind scheme is simple to use, it has been widely applied in applications in computational fluid dynamics. It is easily extended to three-dimensional problems. However, a major drawback with the scheme is that it causes the distribution of the transported properties to become smeared, particularly when the flow is not aligned with the grid lines. The error has a diffusion-like appearance and is referred to as artificial diffusion. Refinement of the grid can overcome this problem, but it is expensive. At high Reynolds numbers the error due to artificial diffusion can be large enough to give physically incorrect results.

In an attempt to reduce the amount of false diffusion present in the simple upwind scheme and to improve the overall accuracy of the finite volume scheme Leonard [6] introduced the QUICK scheme to approximate the convective fluxes. This scheme is based on the use of a second degree polynomial biased toward the upstream direction to interpolate the value of the dependent variable, u_r, at each face of the control volume.

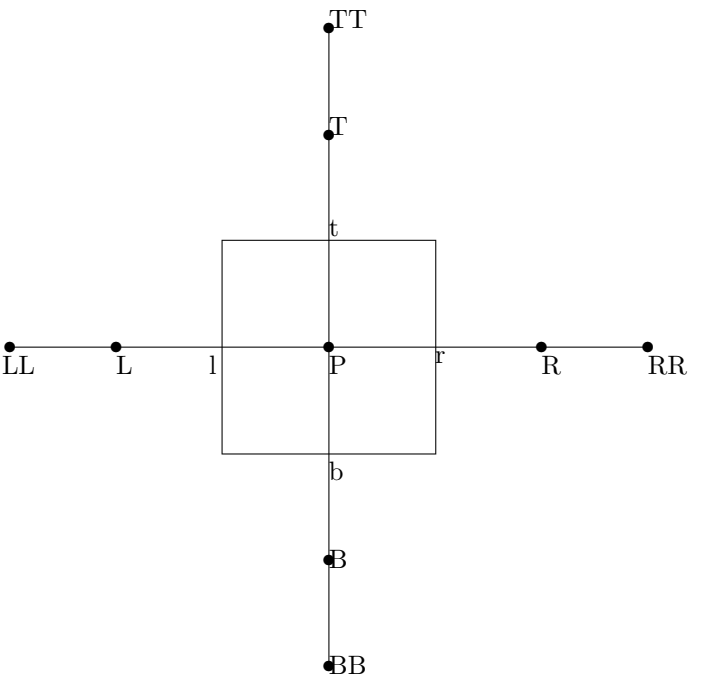


FIG. 4. *A control volume for use with the QUICK scheme.*

This leads to a third order approximation possessing a larger stencil as shown in Figure 4. A drawback of the scheme is that the coefficients corresponding to points R, L, T, and B are not guaranteed to be positive and the coefficients corresponding to points RR, LL, TT, and BB are negative. Thus the QUICK scheme is only

conditionally stable, and there is a tendency for QUICK to give oscillatory results. Furthermore, the conversion of this method to a three-dimensional scheme is likely to cause more difficulties than the upwind method due to the extra terms in the momentum equations.

In this paper we have pursued an entirely different approach to the treatment of convection. Instead of using upwinding to stabilize the scheme at moderate values of the Reynolds number, we have chosen to treat convection in a semi-Lagrangian manner. This is described in the next section.

4. A semi-Lagrangian approach. The underlying problem with the traditional finite volume treatment of convection diffusion problems is that the convection term is discretized using what is essentially a technique that has been constructed and tested for diffusion problems. Upwinding attempts to redress the situation as far as convection is concerned by giving some weighting to the convection part of the problem. The approach adopted in this paper is a time-splitting technique in which we decouple the treatment of convection and diffusion and use appropriate methods of discretization for each subproblem. Over each time interval $[t_n, t_{n+1}]$ we solve a convection equation of the form

$$\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u}^n.\nabla \boldsymbol{u} = 0, \tag{4.1}$$

followed by the solution of an unsteady generalized Stokes problem

$$\frac{\partial \boldsymbol{u}}{\partial t} - \frac{1}{Re}\nabla^2 \boldsymbol{u} + \nabla p = 0, \tag{4.2}$$

$$\nabla.\boldsymbol{u} = 0. \tag{4.3}$$

The natural frame of reference in which to solve (4.1) is defined by the particle following transformation

$$\begin{aligned} \frac{d\xi(t)}{dt} &= u^n(\xi(t), \eta(t), \tau), \\ \frac{d\eta(t)}{dt} &= v^n(\xi(t), \eta(t), \tau), \\ \frac{d\tau(t)}{dt} &= 1, \end{aligned} \tag{4.4}$$

where ξ and η are the spatial variables and τ is the temporal variable. For $\mathbf{x} = (x, y) \in \mathrm{R}^2$ the solution of the system of ordinary differential equations (4.4) for $t \in [t_n, t_{n+1}]$ subject to

$$\xi(t_{n+1}) = x, \quad \eta(t_{n+1}) = y, \quad \tau(t_{n+1}) = t_{n+1}, \tag{4.5}$$

is a continuous curve, known as the trajectory, in space-time passing through the point (x, y, t_{n+1}). After applying a transformation which satisfies (4.4), the governing equation for $\mathbf{u}$ becomes

$$\frac{d\mathbf{u}}{d\tau}(\xi, \eta, \tau) = 0. \tag{4.6}$$

Thus smooth solutions of (4.1) are constant along the characteristic paths [18].

In this section we concentrate on the novel features of our finite volume scheme which is the discretization of (4.1) using a semi-Lagrangian approach. This approach

differs from that proposed by Manson and Wallis [25]. The scheme of Manson and Wallis [25] was developed for pure advection problems in one dimension and is based on a fractional-staged strategy that combines a conventional control volume discretization over a partial time-step $\tau\Delta t$ $(0 < \tau < 1)$ with a semi-Lagrangian treatment over the partial time-step $(1-\tau)\Delta t$. The value of τ is chosen to avoid the need for interpolation in the semi-Lagrangian part of the calculation. The solution of (4.2) and (4.3) follows standard finite volume methodology. The discrete formulation of this problem can be solved using the SIMPLER approach, for example.

The computational domain is partitioned into a number of nonoverlapping control volumes or cells, $C_{i,j}$. This grid, known as the reference grid, remains fixed in space for all time. Consider the mesh associated with one of the dependent variables, ϕ, say, where $\phi = u$ or v. Let $C_{i,j}$ be one such control volume. Let the positions of the corners of cell $C_{i,j}$ be located at the points $\mathbf{X}_{i\pm1/2,j\pm1/2} = (x_{i\pm1/2}, y_{j\pm1/2})$. Particles which arrive at these four corner points at time $t = t_{n+1}$ were located at the vertices of some cell, which is to be determined as part of the solution process, at time $t = t_n$. In general, this will be a deformed control volume which may lie anywhere on the underlying grid or indeed outside the domain if the time-step is sufficiently large. We approximate this cell by a quadrilateral $C^{*n}_{i,j}$, formed by joining the departure points by straight line segments.

Associated with each cell $C_{i,j}$ at each time $t_n = n\Delta t$ an approximation is introduced, denoted by $\bar{\phi}^n_{i,j}$, to the cell average of $\phi(x, y, t_n)$, i.e.,

$$\phi^n_{i,j} \approx \frac{1}{\Delta x_i \Delta y_j} \iint_{C_{i,j}} \phi(x, y, t_n)\, dxdy, \tag{4.7}$$

where

$$\Delta x_i = x_{i+1/2} - x_{i-1/2}, \quad \Delta y_j = y_{j+1/2} - y_{j-1/2}.$$

Note that $\phi^n_{i,j}$ will, in general, be distinct from the pointwise approximation to $\phi(x_i, y_j, t_n)$.

An approximation to the solution of (4.1) is given by

$$\bar{\phi}^{n+1}_{i,j} = \bar{\phi}^{*n}_{i,j}, \tag{4.8}$$

where

$$\bar{\phi}^{*n}_{i,j} \approx \frac{1}{\Delta x_i \Delta y_j} \iint_{C^{*n}_{i,j}} \phi(x, y, t_n)\, dxdy. \tag{4.9}$$

Thus there are two stages to the numerical calculation at each time-step. In the first stage the departure points are determined. These are the vertices of the cells $C^{*n}_{i,j}$. In the second stage the cell average values of ϕ^{*n} are determined from a knowledge of the cell average values of ϕ at time $t = t_n$ on the reference grid. These values are then inserted into the discretized versions of equations (4.2) and (4.3) in order to determine the values of velocity and pressure at the new time-step.

4.1. Calculation of departure points. The departure points at time t_n of each point on the reference grid are determined by solving the particle following transformation (4.4) for $t \in [t_n, t_{n+1}]$, subject to

$$\xi(t_{n+1}) = x_{i+\frac{1}{2},j+\frac{1}{2}}, \quad \eta(t_{n+1}) = y_{i+\frac{1}{2},j+\frac{1}{2}}, \quad \tau(t_{n+1}) = t_{n+1}. \tag{4.10}$$

This computation is performed for each grid point, i.e., for a suitable range of values of i and j in (4.10). The solution of this problem describes the path of the particle that passes through the grid point

$$\mathbf{X}_{i+\frac{1}{2},j+\frac{1}{2}} = (x_{i+\frac{1}{2},j+\frac{1}{2}}, y_{i+\frac{1}{2},j+\frac{1}{2}})$$

at time $t = t_{n+1}$. Each of these paths is traced backwards in time over one time-step. For example, if $C_{i,j}$ represents a control volume for the u component of velocity, then the following one-step method may be used to determine the positions of the corners of $C^{*n}_{i,j}$:

$$\begin{aligned} x^{*n}_{i+\frac{1}{2},j+\frac{1}{2}} &= x^{n+1}_{i+\frac{1}{2},j+\frac{1}{2}} - \frac{\Delta t}{4}(u^n_{i,j} + u^n_{i+1,j} + u^n_{i,j+1} + u^n_{i+1,j+1}), \\ y^{*n}_{i+\frac{1}{2},j+\frac{1}{2}} &= y^{n+1}_{i+\frac{1}{2},j+\frac{1}{2}} - \Delta t v^n_{i+\frac{1}{2},j+\frac{1}{2}}, \end{aligned} \tag{4.11}$$

where subscripts indicate grid locations and superscripts indicate the time-step. The above scheme is first-order in time. Higher-order schemes in time may also be used.

4.2. Calculation of area-weighting coefficients. At the beginning of each time-step the values of $\phi^n_{i,j}$ are known in all control volumes. Given the location of the departure points of the reference grid at time t_n, the values $\phi^{*n}_{i,j}$ must be determined. This approximation is generated by means of an area-weighting technique which uses a weighted sum of the values of ϕ^n over the control volumes on the reference grid which overlap with cell $C^{*n}_{i,j}$. Area-weighting techniques are not new, and they have been demonstrated to possess attractive stability properties. They were originally developed by users of particle-in-cell methods [21, 20]. In the Lagrange–Galerkin finite element method the evaluation of inner products using nonexact integration must be performed with great care [3]. Large classes of well-known quadrature rules lead to conditionally unstable schemes. However, the use of area-weighting can restore the stability properties of the exactly integrated schemes, albeit with a slight degradation in accuracy. In the application of this technique the centroid of each element is tracked, and the whole element is deemed to move without distortion and rotation. However, in the present application of the technique the control volumes are allowed to move with distortion and rotation.

The first-order area-weighting scheme of Scroggs and Semazzi [4] for determining the value of $\phi^{*n}_{i,j}$ is

$$\phi^{*n}_{i,j} = \frac{1}{\Delta x_i \Delta y_j} \sum_{I,J \in Z} \omega^{I,J}_{i,j} \bar{\phi}^n_{I,J}, \tag{4.12}$$

where $\omega^{I,J}_{i,j}$ is the common area between $C^{*n}_{i,j}$ and $C_{I,J}$, i.e., the area of $C_{I,J} \cap C^{*n}_{i,j}$, and Z is the set of indices of all the points in the computational domain. This involves determining how the cell $C^{*n}_{i,j}$ intersects with the control volumes in the fixed grid at time t_n and then to perform an area weighting based on the amount of overlap. Although this procedure is a straightforward exercise in coordinate geometry, it requires careful programming. Details can be found in Williams [17]. This scheme possesses the important property that when it is applied to systems of conservation laws the numerical approximation preserves the discrete conservation property identically, i.e.,

$$\sum_{i,j \in Z} u^{*n}_{i,j} = \sum_{i,j \in Z} u^{n+1}, \tag{4.13}$$

if the boundaries of each control volume $C_{I,J}$ remain inside the computational domain from time t_n to t_{n+1} since

$$\sum_{i,j\in Z} \omega_{i,j}^{I,J} = \text{area of } C_{I,J}. \tag{4.14}$$

Higher-order extensions of this scheme have been derived by Phillips and Williams [16] and have been applied to conservation laws. In particular, Phillips and Williams [16] apply the first-order and higher-order schemes to a rotating disk test problem. The results which have been obtained fully demonstrate the conservation properties of all the schemes.

5. Stability and accuracy. In this section we investigate the stability and convergence properties of the semi-Lagrangian scheme on a pure convection problem in one space dimension. Consider the scalar conservation law

$$\frac{\partial \phi}{\partial t} + a\frac{\partial \phi}{\partial x} = 0, \quad x \in \mathrm{R},\ 0 < t < T, \tag{5.1}$$

where a is a function of x and t, and T is some finite time, with the initial condition

$$\phi(x,0) = \phi_0(x), \quad x \in \mathrm{R}. \tag{5.2}$$

Assume a uniform distribution of grid points $\{x_j : j \in \mathrm{Z}\}$ in the x direction with $x_j = jh$. The discretization of this equation using the semi-Lagrangian scheme (4.8) and (4.12) is

$$\bar{\phi}_j^{n+1} = \bar{\phi}_j^{*n}, \tag{5.3}$$

where

$$\bar{\phi}_j^{*n} = \hat{\alpha}\bar{\phi}_{j-m-1}^{n} + (1-\hat{\alpha})\bar{\phi}_{j-m}^{n}, \tag{5.4}$$

where

$$\hat{\alpha} = \alpha - m,\ \alpha = \frac{a_j^{n+1}\Delta t}{h}.$$

Here we have assumed, without loss of generality, that the departure point at time $t = t_n$ of the particle which is at the point x_j at time $t = t_{n+1}$ lies in the interval $[x_{j-m-1}, x_{j-m}]$. Note that area-weighting in one dimension involves no more than linear interpolation using the information at the two nearest grid points.

Note. If $a(x,t)$ is not constant, then the characteristic path is not a straight line. In this case an error is incurred in locating the departure point. This contributes to the overall global error of the approximation. Furthermore, if $a(x,t)$ varies rapidly in the domain $x \in \mathrm{R}$, $0 < t < T$, then it is necessary to choose Δt sufficiently small so that the computed and the actual departure points of the characteristic passing through the point x_i at time t_{n+1} lie in the same reference cell.

5.1. Stability. The stability analysis is performed in the case when a is a positive constant. If we assume that (5.3)–(5.4) has a solution of the form

$$\bar{\phi}_j^n = \bar{\phi}^0 \lambda^n \exp(ikjh), \tag{5.5}$$

then we can show that λ is given by

$$\lambda = [1 - \hat{\alpha}(1 - \exp(-ikh))] \exp(-ikmh). \tag{5.6}$$

One can show the sufficient condition for stability, i.e., $|\lambda|^2 \leq 1$, is satisfied when

$$0 \leq \hat{\alpha} \leq 1.$$

This condition is satisfied since we have chosen the interval $[x_{j-m-1}, x_{j-m}]$ so that the departure point lies in it. So the scheme is unconditionally stable.

5.2. Convergence. We define the Sobolev space $W^{2,\infty}(\mathrm{R})$ by

$$W^{2,\infty}(\mathrm{R}) = \left\{ \psi : \sup_{x \in \mathrm{R}} \left| \frac{\partial^2 \psi}{\partial x^2} \right| < \infty \right\}$$

with corresponding seminorm

$$\|\psi\|_{2,\infty} \;=\; \sup_{x \in \mathrm{R}} \left| \frac{\partial^2 \psi}{\partial x^2} \right|.$$

Similarly, we define $W^{2,\infty}(\mathrm{R} \times [0,T])$ to be the space of functions with bounded second derivatives in $\mathrm{R} \times [0,T]$. We denote by $L^\infty(0,T;W^{2,\infty}(\mathrm{R}))$ the space of functions $\psi(x,t)$ defined in $\mathrm{R} \times [0,T]$ that belong to $W^{2,\infty}(\mathrm{R})$ for any $t \in [0,T]$ and that satisfy

$$\operatorname{ess} \sup_{t \in [0,T]} \|\psi(\cdot,t)\|_{2,\infty} < \infty.$$

This space is equipped with the norm

$$\|\psi\|_{L^\infty(0,T;W^{2,\infty}(\mathrm{R}))} = \operatorname{ess} \sup_{t \in [0,T]} \|\psi(\cdot,t)\|_{2,\infty}.$$

The following convergence result is proved in Phillips and Williams [16].

THEOREM 5.1. *Let the solution of* (5.1) *belong to*

$$W^{2,\infty}(\mathrm{R} \times [0,T]) \;\cap\; L^\infty(0,T;W^{2,\infty}(\mathrm{R})),$$

and let the numerical approximation be generated by (5.3)– (5.4). *Then the error* $e_j^n = \phi(x_j,t_n) - \bar{\phi}_j^n$ *satisfies the bound*

$$\|e^n\|_\infty = O(\Delta t) + O(\min(h, h^2/\Delta t)). \tag{5.7}$$

One can deduce from this result that for some values of the discretization parameters h and Δt the error will increase as Δt is reduced up to a maximum error which is $O(\Delta t)+O(h)$. This behavior of the error as a function of the time-step Δt is in general agreement with a result of Süli and Ware [7] for the spectral method of characteristics for a similar class of problems.

6. Method of solution. The generalized Stokes problem

$$\frac{\boldsymbol{u}^{n+1} - \boldsymbol{u}^{*n}}{\Delta t} = \frac{1}{Re} \nabla^2 \boldsymbol{u}^{n+1} - \nabla p^{n+1}, \tag{6.1}$$

$$\nabla . \boldsymbol{u}^{n+1} = 0, \tag{6.2}$$

with given boundary conditions on $\boldsymbol{u}^{n+1}$ is discretized using the traditional central difference approach described earlier. Note, however, that now there will be no contribution due to convection. After the governing equations have been discretized in time and space, a suitable solution strategy must be devised to solve the resulting system of algebraic equations. One option is to solve the full problem for velocity and pressure directly at each time-step. For a large number of degrees of freedom this is a very inefficient approach. Instead we have chosen to follow the semi-implicit method for pressure linked equations (SIMPLE) methodology first advocated by Patankar and Spalding [5], which involves decoupling the velocity and pressure computations and iterating between them until convergence is reached at each time-step.

Within the SIMPLE procedure the pressure correction equation is prone to divergence unless some underrelaxation is used. The method recommended by Patankar [19] underrelaxes the velocity components in the momentum equations, with a relaxation factor α approximately equal to 0.5, and to only add a fraction of the pressure correction to the pressure, i.e.,

$$p = p^* + \alpha_p p',$$

where α_p is approximately equal to 0.8. These values for α and α_p are suggested since they have been shown to be satisfactory for a large number of flow problems. They are not necessarily the optimum values and for some problems will not produce a converged solution. It is clear that α and α_p will vary for different flow situations and may also vary for different mesh sizes within the same computational domain. Other relaxation techniques may be applied, and some of these are discussed in [23]. Clearly this algorithm is not robust. In an effort to overcome this problem Patankar [8] devised SIMPLER—SIMPLE Revised.

The argument used in the derivation of SIMPLE is that since the neighbor-point velocity corrections are removed from the velocity correction formula, the pressure correction has the sole responsibility of correcting the velocities. This leads to severe changes in the pressure correction field. Patankar [19] supposes that the pressure correction equation does a reasonable job of correcting the velocities but a poor job of correcting the pressures. The SIMPLER methodology was developed to overcome this deficiency.

To derive a pressure field equation the momentum equation is first written as

$$u_r = \frac{\sum A_{nb} u_{nb} + b_r}{A_r} + d_r(p_P - p_R). \tag{6.3}$$

We define a pseudovelocity of the form

$$\hat{u}_r = \frac{\sum A_{nb} u_{nb} + b_r}{A_r}. \tag{6.4}$$

Substitution of (6.4) into (6.3) gives

$$u_r = \hat{u}_r + d_r(p_P - p_R). \tag{6.5}$$

Similarly, we have

$$v_t = \hat{v}_t + d_t(p_P - p_T). \tag{6.6}$$

The pressure equation is derived in a similar manner to the pressure correction equation. We again integrate the continuity equation over the control volume. If

we substitute (6.5) and (6.6) into the discrete continuity equation and rearrange the terms, we obtain

$$a_P p_P = a_R p_R + a_L p_L + a_T p_T + a_B p_B + b_1 \tag{6.7}$$

with

$$a_R = d_r \Delta y, \tag{6.8}$$

$$a_L = d_l \Delta y, \tag{6.9}$$

$$a_T = d_t \Delta x, \tag{6.10}$$

$$a_B = d_b \Delta x, \tag{6.11}$$

$$a_P = a_R + a_L + a_T + a_B, \tag{6.12}$$

$$b_1 = [\hat{u}_l - \hat{u}_r]\Delta y + [\hat{v}_b - \hat{v}_t]\Delta x. \tag{6.13}$$

No approximations have been used in the derivation of the pressure equation, so if a correct velocity field is used to calculate the pseudovelocities, the pressure equation will give the correct pressure at once.

The algorithm used is based on the SIMPLER algorithm suitably ammended to incorporate the semi-Lagrangian treatment of the convection term and is described below.

The algorithm.

1. Set $n \leftarrow 0$ and define an initial velocity field $\boldsymbol{u}^{(n)}$. Put $\boldsymbol{u} \leftarrow \boldsymbol{u}^{(n)}$.
2. Calculate $u^{*(n)}$ and $v^{*(n)}$ using the semi-Lagrangian approach described in section 4.
3. Calculate pseudovelocities $\hat{u}$, $\hat{v}$ from

$$\begin{aligned} \hat{u}_r &= \tfrac{\sum A_{nb} u_{nb} + b_r}{A_r}, \\ \hat{v}_t &= \tfrac{\sum A_{nb} v_{nb} + b_t}{A_t}, \end{aligned}$$

where

$$\begin{aligned} A_r &= \textstyle\sum_{nb} A_{nb} + \frac{\Delta x \Delta y}{\Delta t}, & A_t &= \textstyle\sum_{nb} A_{nb} + \frac{\Delta x \Delta y}{\Delta t}, \\ b_r &= u_r^{*(n)} \tfrac{\Delta x \Delta y}{\Delta t}, & b_t &= v_t^{*(n)} \tfrac{\Delta x \Delta y}{\Delta t}. \end{aligned}$$

4. Solve the pressure equation for p^*

$$a_P p_P^* = a_R p_R^* + a_L p_L^* + a_T p_T^* + a_B p_B^* + b,$$

where

$$\begin{aligned} a_R &= \tfrac{(\Delta y)^2}{A_r}, & a_L &= \tfrac{(\Delta y)^2}{A_l}, \\ a_T &= \tfrac{(\Delta x)^2}{A_t}, & a_B &= \tfrac{(\Delta x)^2}{A_b}, \\ a_P &= \textstyle\sum_{nb} a_{nb}, \end{aligned}$$

and

$$b = (\hat{u}_l - \hat{u}_r)\Delta y + (\hat{v}_b - \hat{v}_t)\Delta x.$$

5. Solve the momentum equations for u', v'.

$$\begin{aligned} A_r u_r' &= \textstyle\sum_{nb} A_{nb} u_{nb}' + b_r + \Delta y (p_P^* - p_R^*), \\ A_t v_t' &= \textstyle\sum_{nb} A_{nb} v_{nb}' + b_t + \Delta x (p_P^* - p_T^*). \end{aligned}$$

6. Solve a pressure correction equation for p'

$$a_P p'_P = a_R p'_R + a_L p'_L + a_T p'_T + a_B p'_B + b,$$

where

$$b = (u'_l - u'_r)\Delta y + (v'_b - v'_t)\Delta x.$$

7. Correct the velocity field using

$$\begin{aligned} u_r &= u'_r + \tfrac{\Delta y}{A_r}(p'_P - p'_R), \\ v_t &= v'_t + \tfrac{\Delta x}{A_t}(p'_P - p'_T). \end{aligned}$$

8. Return to step 3 and repeat until convergence is obtained.
9. Set $\boldsymbol{u}^{(n+1)} \leftarrow \boldsymbol{u}$ and $n \leftarrow n+1$.
10. Return to step 2, let $\boldsymbol{u} \leftarrow \boldsymbol{u}^{(n)}$, and repeat until a steady state solution is obtained.

Note that step 2 is performed once at the beginning of each time-step and steps 3–8 take place within each time-step. Only when we have a convergent velocity solution within a time-step do we proceed to the next time-step.

7. Numerical results: Conservation law. In this section we perform a numerical experiment to illustrate the important features of the finite volume scheme developed in this paper. In this experiment we demonstrate the accuracy of this scheme by solving a problem possessing an exact solution. We consider uniform meshes only with $h = \Delta x = \Delta y$. Consider the model problem

$$\frac{\partial \phi}{\partial t} + x\frac{\partial \phi}{\partial x} - y\frac{\partial \phi}{\partial y} = 0,\ \mathbf{x} \in [1,2] \times [1,2],\ t \geq 0, \tag{7.1}$$

in which the velocity field, $\boldsymbol{u} = (x, -y)$, is divergence-free. On the two inflow boundaries we prescribe

$$\begin{aligned} \phi(1,y,t) &= 1 + y^2, \quad y \in [1,2],\ t \geq 0, \\ \phi(x,2,t) &= 1 + 4x^2, \quad x \in [1,2],\ t \geq 0. \end{aligned} \tag{7.2}$$

The initial condition is

$$\phi(x,y,0) = 0,\ x \in (1,2],\ y \in [1,2). \tag{7.3}$$

The steady state solution of this problem is

$$\phi(x,y) = 1 + (xy)^2. \tag{7.4}$$

The semi-Lagrangian algorithm is terminated when

$$\frac{\|\phi^{n+1} - \phi^n\|_\infty}{\Delta t} \leq 10^{-5}. \tag{7.5}$$

If $\tilde{\phi}_h$ denotes the converged numerical approximation to the steady state solution of the problem given by (7.4) on a grid with mesh size h, the accuracy of the discrete approximation is measured using

$$E(h) = \frac{\|\phi - \tilde{\phi}_h\|_\infty}{\|\tilde{\phi}_h\|_\infty}. \tag{7.6}$$

TABLE 2
Dependence of the error and number of iterations on mesh size with $\Delta t = 0.01$. Estimates of the order of spatial convergence are also given.

h	$\|e\|_\infty$	Iterations	p
0.25	1.013×10^{-1}	246	-
0.125	4.633×10^{-2}	182	1.13
0.0625	1.846×10^{-2}	140	1.33
0.03125	4.625×10^{-3}	111	1.36

TABLE 3
Dependence of the error on mesh size with $\Delta t = 0.001$.

h	$\|e\|_\infty$
0.25	1.088×10^{-1}
0.125	5.473×10^{-2}
0.0625	2.715×10^{-2}
0.03125	1.324×10^{-2}

In Table 2 we show how the error decays as a function of mesh size when $\Delta t = 10^{-2}$ for the scheme described in this paper. If it is assumed that the error behaves like $O(h^p)$, then the error information on two successive meshes given in Table 2 can be used to estimate the order, p, using

$$p = \frac{\ln(E(h)/E(h/2))}{\ln 2}.$$

These values are also provided in Table 2. We see that in the limit of small h the scheme is more than first-order accurate. The number of iterations required to attain the tolerance (7.5) is also given. An interesting feature is that for a given method the semi-Lagrangian algorithm converges in a fewer number of iterations as the mesh is refined. In Table 3 we show how the errors decay for a smaller time-step $\Delta t = 10^{-3}$. In all cases the errors are higher than the corresponding ones in Table 2 as predicted by the error estimate derived in Theorem 1, and the orders of spatial convergence are slightly lower. A second-order variant of the scheme has been developed for conservation laws [16]. This is based on a second-order Runge–Kutta method for determining the departure points and a second-order area-weighting scheme to ensure that the discrete conservation principle is satisfied identically.

8. Numerical results: Newtonian flow. In this section numerical calculations of laminar flow through a 2:1 contraction are presented for a range of Reynolds numbers. These calculations are compared with those generated by other authors [11], [10], [2] using different techniques. In particular, the behavior of the salient corner vortex is investigated qualitatively and quantitatively.

The work of Dennis and Smith [11] on the 2:1 contraction problem is a benchmark against which to test new and emerging numerical techniques for the Navier–Stokes equations. Their method is based on a finite difference approximation of the stream function-vorticity formulation of the governing equations. An upwind differencing scheme of Dennis and Hudson [13] is used to approximate the vorticity transport equation by adding an extra "viscous-like" term proportional to h^2. As we mentioned earlier, the standard upwind approximation is only first-order accurate. The addition of this term, known as artificial viscosity, is essential in order to obtain converged solutions for reasonably large values of Re. Without it the system of algebraic equations loses its diagonal dominance which presents convergence difficulties when solved by

iterative methods unless the mesh size is sufficiently decreased. The advantage of this upwinding scheme is that the second-order accuracy of the finite difference equations is maintained and that the artificial viscosity is applied sparingly at each grid point only if it is required. The local truncation error of the scheme is $O(h^2)$, and, therefore, it is strictly second order if $hRe << 1$. Therefore, for large values of Re small grid sizes are required for meaningful results.

Hunt [2] also uses a finite difference discretization of the stream function-vorticity formulation but on a nonuniform grid. The vorticity unknowns are eliminated to give a system solely in terms of unknown values of the stream function. There are some puzzling features of the scheme. First, when upwinding is used, the scheme fails to converge for $Re > 500$, even though one would expect the addition of a "viscous-like" term to have a stabilizing effect. Second, for a given value of Re, convergence becomes more difficult with mesh refinement.

Karageorghis and Phillips [10] use a spectral domain decomposition method to discretize the stream function formulation of the Navier–Stokes equations. The flow region is decomposed into a number of rectangular subdomains, on each of which the stream function is approximated by a truncated double Chebyshev expansion. The approximations are C^1 continuous across subregion interfaces. The nonlinear fourth-order partial differential equation for the stream function is linearized using Newton's method.

Mesh refinement is studied with reference to the size of the salient or corner vortex. This flow feature is also used as the basis of comparisons with other methods. The length of the corner vortex, L_1, is defined to be the distance between the point where the separation line meets the bottom of the channel and the salient corner. The width, L_2, of the corner vortex is defined to be the distance between the point where the separation line meets the wall parallel to the y-axis at $x = 0$ and the salient corner.

Numerical computations are performed on a series of meshes in order to ensure that the solutions obtained are independent of the mesh parameters. Both uniform and nonuniform meshes are used. The mesh parameters for the four uniform meshes (A–D) are given in Table 4. The main characteristics of the nonuniform meshes (E–G*) are given in Table 5. Meshes E–G* have mesh spacings which vary geometrically from the reentrant corner. In this way we can ensure a greater density of mesh points in the region where the solution changes most rapidly. Note that although meshes F and G contain approximately the same number of control volumes, they differ in the way the nonuniform mesh spacing varies. Mesh G is more refined around the reentrant corner than mesh F. Mesh G* corresponds to a computational domain in which the exit length is doubled from four to eight units. This extended domain allows for the examination of domain truncation effects on the numerical solution for $Re = 500$ and $Re = 1000$.

Tables 6 and 7 show the dependence of the length L_1 and the width L_2 of the salient corner vortex on the mesh. These flow characteristics are sensitive to the computational mesh. The results demonstrate that convergence with mesh refinement has been obtained over the whole range of values of the Reynolds number. The results on all the meshes were calculated with $\Delta t = 10^{-3}$. Tables 6 and 7 also show that the use of the extended domain has no appreciable effect on the values of L_1 and L_2, thus confirming that the length of the downstream channel is adequate for these computations.

Allowing $x \to \infty$ along $y = -\frac{1}{2}$, we would expect $\xi \to -12$ for a fully developed

TABLE 4
The mesh spacings and degrees of freedom for the uniform meshes A–D.

Mesh	Δx	Δy	degrees of freedom
Mesh A	$\Delta x = 0.05$	$\Delta y = 0.05$	6840
Mesh B	$\Delta x = 0.025$	$\Delta y = 0.05$	13720
Mesh C	$\Delta x = 0.05$	$\Delta y = 0.033$	10420
Mesh D	$\Delta x = 0.05$	$\Delta y = 0.025$	14000

TABLE 5
Mesh characteristics and degrees of freedom for the nonuniform meshes E–G*.

Mesh	Δx_{min}	Δy_{min}	Degrees of freedom
Mesh E	1.131×10^{-2}	2.5×10^{-2}	10800
Mesh F	1.131×10^{-2}	1.852×10^{-2}	14580
Mesh G	4.118×10^{-3}	1.512×10^{-2}	14400
Mesh G*	4.118×10^{-3}	1.512×10^{-2}	16800

flow profile at the exit. We may apply this test to check that the downstream channel is long enough for the Re numbers that we solve for. In Figures 5–7 we have plotted vorticity against downstream channel length for $Re = 1$, $Re = 100$, and $Re = 500$, respectively. These figures suggest that the downstream channel length of 4 is suitable for this range of Reynolds numbers.

The sensitivity of the computation with respect to the choice of time-step is shown in Table 8. In this table the dependence of L_1 with respect to the time-step is presented for $Re = 100$. The choice of time-step does not significantly influence the value of this characteristic of the flow problem.

The asymptotic theory of Smith [12] predicts that a separation will occur asymptotically far ahead of the step at a position $x = -L_1$, given by

$$L_1 = 0.1289 ln\, Re + D \,\text{for} Re \gg 1. \tag{8.1}$$

The constant D is of order unity, and its value depends on the contraction ratio. It is determined here by looking at the asymptotic behavior of $L_1 - 0.1289 ln\, Re$. From this analysis we obtain $D = -0.547$. In Figure 8 we plot L_1 given by (8.1) as a function of Re. On this figure we also include the values of L_1 obtained by our numerical calculations. Excellent agreement is obtained with the theoretical prediction for $Re \geq 300$. Note that the theoretical prediction is only valid for large values of the Reynolds number, although in Figure 8 it is plotted on the whole domain.

In Figures 9–14 the streamline contours are presented for Re=1, 10, 50, 100, 200, and 500, respectively, in the domain $-2 \leq x \leq 2$, $-1 \leq y \leq 0$. The salient corner vortex diminishes in size as Re increases from $Re = 1$ to $Re = 50$ and then starts to grow slowly for $Re > 50$.

In Table 9 we compare the values of L_1 obtained for different values of Re on mesh G (G* for $Re = 500,\ 1000$) with other results in the literature (Dennis and Smith [11], Hunt [2], and Karageorghis and Phillips [10]). The results of Dennis and Smith [11] have been obtained using two successive h^2-extrapolations on grids with mesh spacings of $h = \frac{1}{10}, \frac{1}{20}$ and $h = \frac{1}{40}$. The results of Hunt [2] are obtained using a transformed grid with 48×128 points and are calculated with and without the artificial viscosity term. The results of Karageorghis and Phillips are given for the most refined grid that they use with 1537 degrees of freedom.

The results in columns (a), (b), and (c) of Table 9 differ by at most 5% from each other for $1 \leq Re \leq 150$. As Re increases from 150, the values for L_1 in the first three

TABLE 6
The length, L_1, of the salient corner vortex on meshes A–G *as a function of Re.*

Re	A	B	C	D	E	F	G	G*
0				0.255	0.285	0.287	0.285	
1	0.246	0.247	0.252	0.255	0.255	0.255	0.255	
10	0.145	0.142	0.152	0.153	0.151	0.152	0.151	
50	0.118	0.123	0.122	0.122	0.122	0.123	0.122	
100	0.148	0.155	0.146	0.144	0.143	0.144	0.144	
150					0.164	0.163	0.163	
200	0.240	0.219	0.194	0.188	0.186	0.185	0.185	
300					0.223	0.222	0.223	
400					0.249	0.248	0.249	
500					0.267	0.267	0.267	0.268
1000					0.334	0.334	0.335	0.338

TABLE 7
The width, L_2, of the salient corner vortex on meshes A–G *as a function of Re.*

Re	A	B	C	D	E	F	G	G*
0					0.346	0.345	0.345	
1	0.301	0.316	0.295	0.293	0.295	0.294	0.294	
10	0.154	0.162	0.148	0.144	0.147	0.147	0.146	
50	0.114	0.127	0.109	0.106	0.109	0.110	0.110	
100	0.127	0.151	0.120	0.115	0.118	0.118	0.119	
200	0.155	0.174	0.140	0.133	0.133	0.135	0.134	
500					0.157	0.158	0.157	0.158
1000					0.174	0.173	0.173	0.175

columns of the table become closer to each other and agree to within 2%. This shows that the semi-Lagrangian scheme performs particularly well for convection dominated flows. Hunt's finite difference scheme with artificial viscosity gives results that are closer to those in the first three columns than the scheme without artificial viscosity.

In Table 10 the width, L_2, is presented for various values of Re and is compared with the values published by Dennis and Smith [11] and Hunt [2]. The values in columns (a) and (b) are within 11% of each other, although as Re increases from $Re = 50$ this percentage difference falls and for $Re = 500$ the results in columns (a) and (b) are around 1% of each other. As for the L_1 results, this would appear to show that the semi-Lagrangian scheme behaves well for high values of Re. For the schemes of Hunt [2] the values of L_2 generated are smaller than those in the first two columns for the scheme with artificial viscosity and, equivalently, larger for the scheme without artificial viscosity.

An interesting feature of the results in Tables 9 and 10 is that as Re grows from 1 to 50, both values drop so that for $Re = 50$ the value of L_1 is approximately half of its equivalent value for $Re = 1$, and the value of L_2 is approximately a third of its value for $Re = 1$. As Re increases from 50 to 1000, both the width and the length of the vortex grow but at different rates; L_1 grows more quickly than L_2. At $Re = 500$ the length L_1 is slightly larger than it was for $Re = 1$, while the value of L_2 at $Re = 500$ is still only 50% of its value at $Re = 1$. This shows that the vortex grows in size along the upstream channel more quickly than up the wall at $x = 0$ as Re increases.

The maximum values of the stream function ψ, ψ_{max}, are presented in Table 11. As we would predict from the results in Tables 9 and 10, the strength of the vortex diminishes in size between $Re = 1$ and $Re = 50$ and then grows for $50 < Re \leq 1000$.

There is no downstream recirculation region pictured in the streamline plots

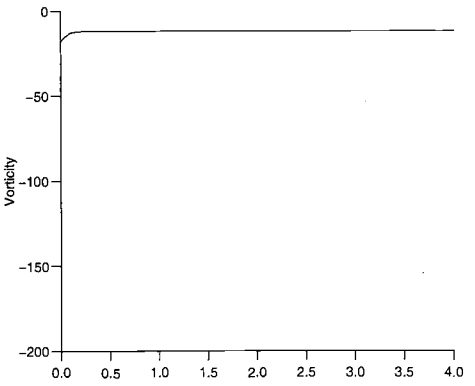


FIG. 5. *Wall vorticity* $\xi(x, -\frac{1}{2})$ *for* $x > 0$, *for* $Re = 1$.

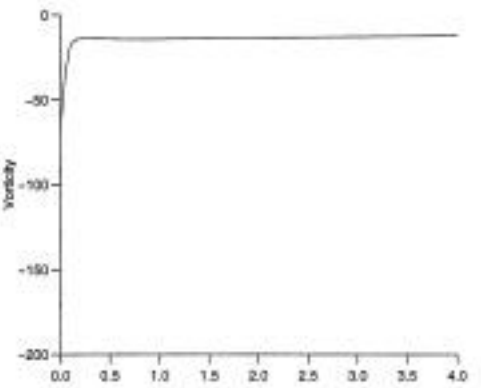


FIG. 6. *Wall vorticity* $\xi(x, -\frac{1}{2})$ *for* $x > 0$, *for* $Re = 100$.

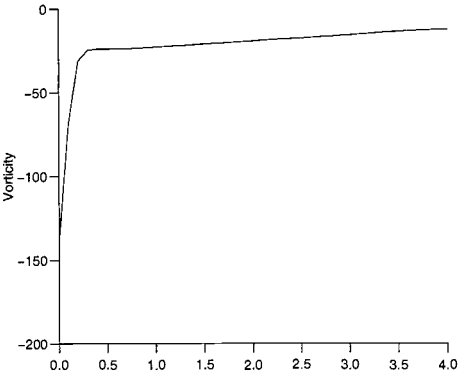


FIG. 7. *Wall vorticity* $\xi(x, -\frac{1}{2})$ *for* $x > 0$, *for* $Re = 500$.

TABLE 8
Dependence of L_1 on Δt for $Re = 100$.

Δt	L_1	L_2
10^{-3}	0.144	0.119
10^{-4}	0.145	0.121
10^{-5}	0.145	0.122

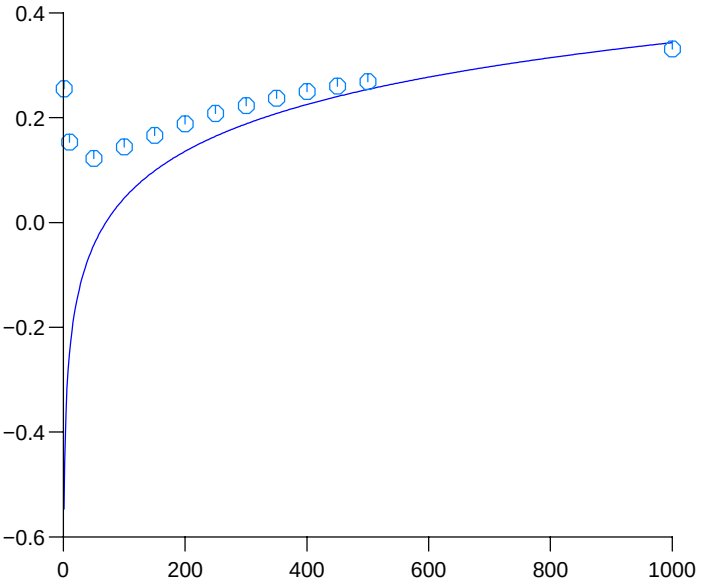


FIG. 8. *Comparison of the asymptotic predition of the value of L_1 by Smith* [12] *with the numerical results obtained in column* (a) *of Table* 9 *as a function of the Reynolds number.*

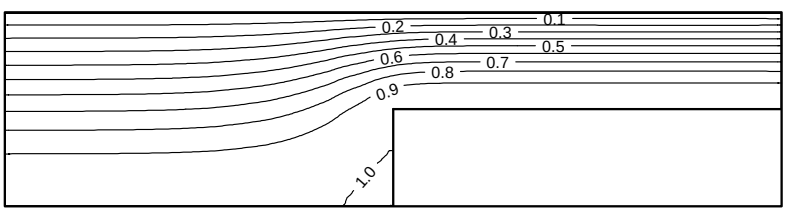


FIG. 9. *Streamlines for $Re = 1$.*

shown in Figures 9–14. However, a close examination of the values of u and v in the region of the reentrant corner indicates that for $Re \geq 100$ some recirculation may exist. For this part of the flow to be accurately resolved more work needs to be done on refining the mesh.

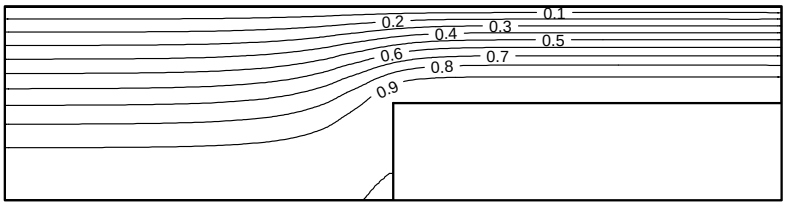


FIG. 10. *Streamlines for* $Re = 10$.

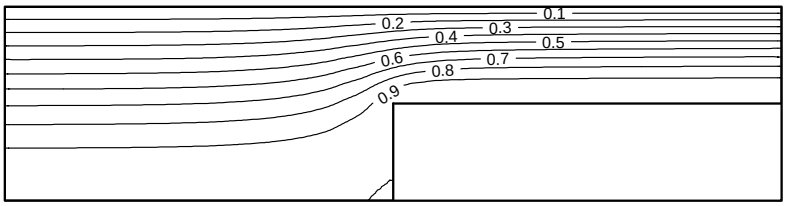


FIG. 11. *Streamlines for* $Re = 50$.

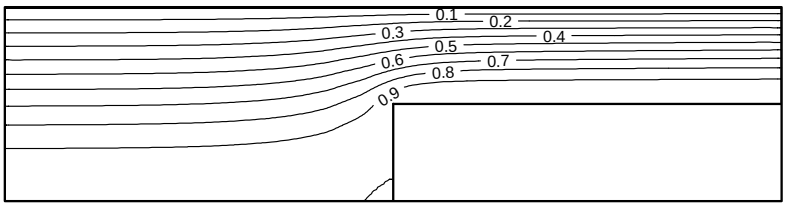


FIG. 12. *Streamlines for* $Re = 100$.

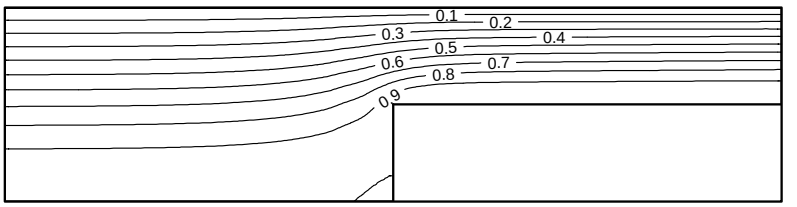


FIG. 13. *Streamlines for* $Re = 200$.

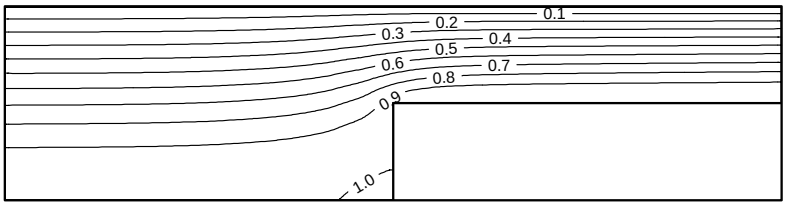


FIG. 14. *Streamlines for* $Re = 500$.

TABLE 9

The length, L_1*, of the salient corner vortex for* (a) *the semi-Lagrangian scheme,* (b) *the spectral collocation method of Karageorghis and Phillips* [10]*,* (c) *the finite-difference scheme of Dennis and Smith* [11]*,* (d) *the finite-difference scheme of Hunt* [2] *without artificial viscosity, and* (e) *the finite-difference scheme of Hunt* [2] *with artificial viscosity.*

Re	(a)	(b)	(c)	(d)	(e)
1	0.255	-	0.255	-	-
10	0.151	0.148	0.155	-	-
50	0.122	0.123	0.129	-	-
100	0.144	0.140	0.144	-	-
150	0.163	0.155	-	-	-
200	0.185	0.183	-	-	-
250	0.208	0.205	-	0.227	0.209
300	0.223	0.223	-	-	-
350	0.237	0.233	-	-	-
400	0.249	0.244	-	-	-
450	0.260	0.255	-	-	-
500	0.268	0.265	0.266	0.308	0.260
1000	0.338	-	0.341	0.394	-

TABLE 10

The width, L_2*, of the salient corner vortex for* (a) *the semi-Lagrangian scheme,* (b) *the finite-difference scheme of Dennis and Smith* [11]*,* (c) *the finite-difference scheme of Hunt* [2] *without artificial viscosity, and* (d) *the finite-difference scheme of Hunt* [2] *with artificial viscosity.*

Re	(a)	(b)	(c)	(d)
1	0.294	0.303	-	-
10	0.146	0.160	-	-
50	0.110	0.122	-	-
100	0.119	0.125	-	-
500	0.158	0.159	0.164	0.149
1000	0.175	0.177	0.188	-

9. Concluding remarks. A semi-Lagrangian finite volume method for solving the time-dependent incompressible Navier–Stokes equations is presented. A time-splitting scheme is used to decouple the treatment of convection from the solution of a generalized Stokes problem. The convection problem is solved using a semi-Lagrangian approach in which the vertices of a control volume at the new time level are traced back in time over a time-step using a particle following transformation. The values of the velocity components in the transformed control volumes at the previous time level are determined using an area-weighting technique which ensures

TABLE 11
The maximum value of the streamfunction, ψ, of the salient corner vortex.

Re	ψ_{max}
1	1.00047
10	1.00009
50	1.00007
100	1.00010
500	1.00054
1000	1.00109

that the conservation property enjoyed by the pure convection problem is satisfied indentically by construction. This approach circumvents problems associated with upwind biased schemes. The generalized Stokes problem is solved using the standard SIMPLER method. The scheme is applied to the flow through a 2:1 contraction and is demonstrated to be robust and accurate. Comparisons are made with other published work on this problem and excellent agreement is found.

Future work will concentrate on developing higher-order methods in time for integrating along the particle paths. The extension of this technique to problems in computational rheology is currently in progress [14, 15].

Acknowledgment. The second author would like to thank the United Kingdom Engineering and Physical Science Research Council for financial support in the form of a research studentship.

REFERENCES

[1] C. HIRSCH, *Numerical Computation of Internal and External Flows. Vol.* 2: *Computational Methods for Inviscid and Viscous Flows*, John Wiley, Chichester, UK, 1990.

[2] R. HUNT, *The numerical solution of the laminar flow in a constricted channel at moderately high Reynolds number using Newton iteration*, Internat. J. Numer. Methods Fluids, 11 (1990), pp. 247–259.

[3] K. W. MORTON, A. PRIESTLEY, AND E. SÜLI, *Stability analysis of the Lagrange–Galerkin method with nonexact integration*, RAIRO Modél. Math. Anal. Numér., 22 (1988), pp. 625–653.

[4] J. S. SCROGGS AND F. H. M. SEMAZZI, *A conservative semi-Lagrangian method for multidimensional fluid dynamics applications*, Numer. Methods Partial Differential Equations, 11 (1995), pp. 445–452.

[5] S. V. PATANKAR AND D. B. SPALDING, *A calculation procedure for heat, mass and momentum transfer in three dimensional parabolic flows*, Int. J. Heat Mass Transfer, 15 (1972), pp. 1787–1806.

[6] B. P. LEONARD, *A stable and accurate convection modelling procedure based on quadratic upstream interpolation*, Comput. Methods Appl. Mech. Engrg., 15 (1979), pp. 59–98.

[7] E. SÜLI AND A. WARE, *A spectral method of characteristics for hyperbolic problems*, SIAM J. Numer. Anal., 28 (1991), pp. 423–445.

[8] S. V. PATANKAR, *A calculation procedure for two-dimensional elliptic situations*, Numer. Heat Transfer, 4 (1981), pp. 409–425.

[9] P. H. GASKELL AND A. K. C. LAU, *Curvature-compensated convective transport: SMART, a new boundedness-preserving transport algorithm*, Internat. J. Numer. Methods Fluids, 8 (1988), pp. 617–641.

[10] A. KARAGEORGHIS AND T. N. PHILLIPS, *Conforming Chebyshev spectral methods for the solution of laminar flow in a constricted channel*, IMA J. Numer. Anal, 11 (1991), pp. 33–54.

[11] S. C. R. DENNIS AND F. T. SMITH, *Steady flow through a channel with a symmetrical constriction in the form of a step*, Proc. Roy. Soc. London A, 372 (1980), pp. 393–414.

[12] F. T. SMITH, *The separating flow through a severely constricted symmetric tube*, J. Fluid Mech., 90 (1979), pp. 725–754.

[13] S. C. R. Dennis and J. D. Hudson, *A difference method for solving the Navier–Stokes equations*, in Proceedings of the First Conference on Numerical Methods in Laminar and Turbulent Flow, Pentech Press, London, 1978.

[14] T. N. Phillips and A. J. Williams, *Semi-Lagrangian finite volume method for viscoelastic flow problems*, in The Mathematics of Finite Elements and Applications X, J. R. Whitman, ed., Elsevier, Amsterdam, 2000, pp. 335–344.

[15] T. N. Phillips and A. J. Williams, *A semi-Lagrangian finite volume method for solving viscoelastic flow problems*, in Proceedings of the Fifth European Rheology Conference, I. Emri and R. Cvelbar, eds., Steinkopff, Portorož, Slovenia, 1998, pp. 299–300.

[16] T. N. Phillips and A. J. Williams, *Conservative semi-Lagrangian finite volume schemes*, Numer. Methods Partial Differential Equations, to appear, 2001.

[17] A. J. Williams, *A Semi-Lagrangian Finite Volume Method for Incompressible Fluid Flow*, Ph.D. thesis, University of Wales, Aberystwyth, UK, 1998.

[18] R. J. LeVeque, *Numerical Methods for Conservation Laws*, 2nd ed., Lectures in Mathematics ETH, Zürich, Birkhauser Verlag, Basel, 1992.

[19] S. V. Patankar, *Numerical Heat Transfer and Fluid Flow*, Hemisphere, New York, 1981.

[20] R. W. Hockney and J. W. Eastwood, *Computer simulation using particles*, Math. Comput., McGraw-Hill, New York, 1981.

[21] F. H. Harlow, *The Particle in Cell Computing Method for Fluid Dynamics*, Methods Comput. Phys. 3, B. Adler, S. Fernbach, and M. Rotenberg, eds., Academic Press, New York, 1964.

[22] O. Pironneau, *On the transport-diffusion algorithm and its applications to the Navier–Stokes equations*, Numer. Math., 38 (1982), pp. 309–332.

[23] J. P. Van Doormal and G. D. Raithby, *Enhancements of the SIMPLE method for predicting incompressible fluid flows*, Numer. Heat Transfer, 7 (1984), pp. 147–163.

[24] J. R. Manson and S. G. Wallis, *Accuracy characteristics of traditional finite volume discretizations for unsteady computational fluid dynamics*, J. Comput. Phys., 132 (1997), pp. 149–153.

[25] J. R. Manson and S. G. Wallis, *Accurate numerical simulation of advection using large time steps*, Internat. J. Numer. Methods Fluids, 24 (1997), pp. 127–139.

SIAM J. SCI. COMPUT.
Vol. 22, No. 6, pp. 2178–2193
© 2001 Society for Industrial and Applied Mathematics

AN $\mathcal{O}(N)$ LEVEL SET METHOD FOR EIKONAL EQUATIONS*

SEONGJAI KIM†

Abstract. A propagating interface can develop corners and discontinuities as it advances. Level set algorithms have been extensively applied for the problems in which the solution has advancing fronts. One of the most popular level set algorithms is the so-called fast marching method (FMM), which requires total $\mathcal{O}(N \log_2 N)$ operations, where N is the number of grid points. The article is concerned with the development of an $\mathcal{O}(N)$ level set algorithm called the *group marching method* (GMM). The new method is based on the narrow band approach as in the FMM. However, it is incorporating a correction-by-iteration strategy to advance a group of grid points at a time, rather than sorting the solution in the narrow band to march forward a single grid point. After selecting a group of grid points appropriately, the GMM advances the group in two iterations for the cost of slightly larger than one iteration. Numerical results are presented to show the efficiency of the method, applied to the eikonal equation in two and three dimensions.

Key words. level set method, narrow band approach, eikonal equation, first-arrival traveltime

AMS subject classifications. 65M06, 86A15

PII. S1064827500367130

1. Introduction.

A propagating interface can develop corners and discontinuities as it advances. It is the case for the solution of, e.g., the eikonal equation, a Hamilton–Jacobi differential equation, in heterogeneous media. Since we do not know which values to assign to the gradient components at the corners and even do not know which values to assign to the function at discontinuities, the solution does not satisfy the equation in the normal (classical) sense. An immediate consideration is to introduce a nonclassical weak solution: a continuous function having possibly discontinuous gradients that solves the equation in average.

Unfortunately, the class of weak solutions is too big; the initial data for these problems does not determine weak solutions uniquely. One way to enforce the uniqueness is to add viscosity described by a dissipative term, e.g., a multiple of the negative Laplace operator, to the equation. The solution for a given viscosity is smooth and therefore uniquely determined. Then one may take the limit of these smooth solutions by letting the coefficient of the Laplace operator approach zero. If the limit exists, its solution is uniquely determined for the appropriate initial data; it is called a *viscosity solution*.

Properties of viscosity solutions were first studied by Lax [14] in the context of hyperbolic conservation laws; good references to work on them are [3, 4, 15].

It has been conjectured that *the first-arrival traveltime (FATT) field is a viscosity solution of the eikonal equation* [30]. The conjecture is supported by some theoretical results and by a great deal of numerical evidence. Techniques for computing viscosity solutions were developed first for hyperbolic conservation laws. Numerical methods for these conservation laws have utilized *upwind* finite differences (FDs) and have adapted to produce viscosity solutions of the eikonal equation [6, 12, 19, 25, 30]. Upwind FD techniques and their applications to Hamilton–Jacobi equations can be found in [16, 17, 18].

*Received by the editors February 7, 2000; accepted for publication (in revised form) December 12, 2000; published electronically April 26, 2001.
http://www.siam.org/journals/sisc/22-6/36713.html
†Department of Mathematics, University of Kentucky, Lexington, KY 40506-0027 (skim@ms.uky.edu).

The level set method is a numerical technique to compute advancing fronts and has been applied to a wide range of important physical problems; see, e.g., [26, 27] and references therein. Even though its solution shows first-order accuracy, the level set method has been widely used mostly due to its built-in stability. It has been adapted for the computation of the FATT and implemented with the narrow band technique, called the *fast marching method* (FMM), in which the narrow band points form a neighborhood of advancing wavefronts [16, 19, 25, 26, 27, 29]. The FMM was first developed by Tsitsiklis [29] and later rederived by Sethian [25]. The FMM requires sorting the solution at each step of the narrow band. For the binary tree sorting, the total cost of the method becomes $\mathcal{O}(N \log_2 N)$. The main object of the article is to develop an $\mathcal{O}(N)$ level set method for the FATT, called the *group marching method* (GMM).

An outline of the paper is as follows. In section 2, the eikonal equation is introduced as the model equation, along with its numerical techniques to solve. The first-order upwind FD scheme is presented, and the FMM is briefly reviewed. In section 3, we suggest the GMM; a pseudocode is presented. Section 4 is devoted to the introduction of the *average normal slowness* which can improve accuracy of the solution. An approximate average slowness which can be easily implemented is discussed in the same section. In section 5, an alternative update formula is considered to improve flexibility in incorporating the average slowness. In section 6, the GMM is tested for various velocity models, update formulae, and average slowness incorporation, in two and three dimensions, and is compared with the FMM. It is numerically verified that the new method performs with the computation cost $\mathcal{O}(N)$. Section 7 discusses some important aspects on accuracy, efficiency, and applicability for the level set methods. The last section includes conclusions.

2. Preliminaries.

2.1. The eikonal equation. The eikonal equation in an isotropic medium is given by

$$|\nabla \tau(\mathbf{x}_s, \mathbf{x})|^2 = \frac{1}{v^2(\mathbf{x})}, \tag{2.1}$$

where $\tau(\mathbf{x}_s, \mathbf{x})$ is the traveltime of the acoustic wave from the source $\mathbf{x}_s$ to the location $\mathbf{x}$ and $v(\mathbf{x})$ denotes the velocity of the propagating wavefront at $\mathbf{x}$. For down-going wavefronts (advancing in the $(z+)$-direction), for example, the equation can be rewritten in the evolutionary form

$$\tau_z = \sqrt{s^2 - \tau_x^2 - \tau_y^2}, \tag{2.2}$$

where $s(= 1/v)$ is called the slowness, the reciprocal of the velocity.

As mentioned earlier, the FATT is a (continuous) viscosity solution of the eikonal equation. There have been various numerical techniques for FATTs: ray-tracing methods, FD methods, and algorithms based on Fermat's principle [24].

The use of ray tracing followed by interpolation of traveltimes is a popular and robust method for computing diffraction trajectories for small or moderate velocity contrasts. For regions with high velocity contrasts, ray tracing methods can produce quite large shadow zones where the computed traveltime field should be interpolated. Such ray-tracing/interpolation processes are cumbersome and computationally expensive, especially in three dimensions. For a comprehensive treatment of ray theory, see [1, 2, 7].

An alternative to the method is to compute traveltimes by solving directly the eikonal equation on a regular grid by FD schemes; see [5, 6, 12, 30, 31, 32] for standard expanding-box methods and [19, 21] for the level set method. A drawback of the standard FD eikonal solvers is that it is not easy to control the propagation angle of rays as it is with ray tracing methods. The level set method incorporating the narrow band method overcomes the drawback of the standard FD schemes; however, it has first-order accuracy and is hardly extendable for a higher-order scheme in realistic media. Nonetheless the FD eikonal solvers have their advantages: they are more efficient computationally than ray tracers and rarely produce shadow zones. The author recently suggested a stable FD eikonal solver incorporating a postsweeping iteration [12], which is second-order and readily extendable to higher-order schemes.

The current level set methods require sorting the traveltimes in the narrow band, a neighborhood of the wavefront, at each stage of the computation. The FMM [25, 28], a narrow band level set method, has adopted the binary tree sorting algorithm; the method turns out to require a total of $\mathcal{O}(N \log_2 N)$ operations, where N is the number of grid points. Even though the level set methods have first-order accuracy, they are still attractive due to their built-in stability and a wide range of applications. It is worth developing an optimal cooperating technique with which the level set method costs $\mathcal{O}(N)$ operations in total.

2.2. The FD scheme. For a numerical scheme for (2.1), consider a cubic domain

$$(x_{\min}, x_{\max}) \times (y_{\min}, y_{\max}) \times (z_{\min}, z_{\max});$$

partition it into $N_x \times N_y \times N_z$ cells of the uniform size $\Delta x \times \Delta y \times \Delta z$ with their vertices

$$\mathbf{x}_{i,j}^k = (x_i, y_j, z_k) = (x_{\min} + i\Delta x, y_{\min} + j\Delta y, z_{\min} + k\Delta z).$$

Let $\tau_{i,j}^k = \tau(\mathbf{x}_s, \mathbf{x}_{i,j}^k)$, and define the forward $(+)$ and backward $(-)$ difference operators for τ_x at the point $\mathbf{x}_{i,j}^k$:

$$D_x^{\pm}\tau_{i,j}^k = \pm\frac{\tau_{i\pm1,j}^k - \tau_{i,j}^k}{\Delta x}. \tag{2.3}$$

Define the *upwind* FD scheme for τ_x incorporating the first-arrivals:

$$\widehat{D}_x\tau_{i,j}^k = \text{mod\_max}\Big(\max\big(D_x^-\tau_{i,j}^k, 0\big), \min\big(D_x^+\tau_{i,j}^k, 0\big)\Big), \tag{2.4}$$

where mod_max returns the larger value in modulus. Note that

$$|\widehat{D}_x\tau_{i,j}^k| = \max\big(D_x^-\tau_{i,j}^k, -D_x^+\tau_{i,j}^k, 0\big),$$

which can be utilized in implementation.

After introducing the analogues for τ_y and τ_z, i.e., $\widehat{D}_y\tau_{i,j}^k$ and $\widehat{D}_z\tau_{i,j}^k$, we can formulate the upwind FD scheme for (2.1):

$$\Big(\widehat{D}_x\tau_{i,j}^k\Big)^2 + \Big(\widehat{D}_y\tau_{i,j}^k\Big)^2 + \Big(\widehat{D}_z\tau_{i,j}^k\Big)^2 = \Big(s_{i,j}^k\Big)^2. \tag{2.5}$$

For down-going wavefronts as in (2.2), the scheme can be explicitly rewritten as

$$\tau_{i,j}^{k+1} = \tau_{i,j}^k + \Delta z \cdot H[\tau]_{i,j}^k, \tag{2.6}$$

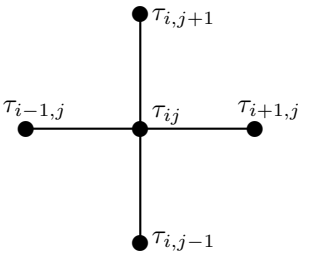


FIG. 1. *The update procedure of the FMM for* τ_{ij}.

where

$$H[\tau]_{i,j}^k = \sqrt{\left(s_{i,j}^k\right)^2 - \left(\widehat{D}_x \tau_{i,j}^k\right)^2 - \left(\widehat{D}_y \tau_{i,j}^k\right)^2}.$$

The quantity inside the radical can be negative in practical simulation. Then it can be either ignored or adjusted by a nonnegative value, depending on the algorithm adopted. In either case, it should be carefully designed not to violate causality or stability and not to deteriorate accuracy.

2.3. Review of the FMM. In this section we review the FMM, a narrow band level set algorithm, presented in [25, 27]. The FMM begins from the source and updates the traveltimes at neighboring grid points. The six neighboring points (four in two dimensions) form the first stage of the narrow band. Then, the FMM expands the narrow band by updating traveltimes at neighboring downwind points of a *Trial* point in the narrow band, and accepting the trial point as an *Alive* (completed) point. The trial point is selected such that its traveltime is smallest among all values available from the narrow band, for which the FD scheme easily satisfies causality.

We summarize the FMM as follows [27]. First, tag the source point as *Alive*, where the traveltime is zero. Then compute traveltimes at all points one grid point away from the source, and tag them as *Close*. Finally, tag *Far* all other grid points, and set the traveltimes for the *Far* points large. Then the FMM loop is carried out as follows.

(a) Let *Trial* be the point in *Close* having the smallest traveltime.
(b) Tag as *Close* all neighbors of *Trial* that are not *Alive*. (If the neighbor is in *Far*, remove it from the list and add it to the set *Close*.)
(c) Recompute the traveltimes at all *Close* neighbors of *Trial* using (2.5).
(d) Remove the point *Trial* from *Close* and add it to *Alive*.
(e) If *Close* is not empty, go to top of loop.

Note that in step (a) of the FMM loop, the point *Trial* is chosen to have the smallest traveltime among all values in *Close*. This implies that the traveltimes in *Close* should be sorted from the smallest to the largest, at each stage of the narrow band. When a binary tree sorting algorithm is applied, the cost is $\mathcal{O}(\log_2 N_B)$, where N_B is the number of grid points in *Close*. Due to the sorting algorithm, the total cost for the FMM becomes $\mathcal{O}(N \log_2 N)$, where N is the total number of grid points.

Now, we present the update procedure (c) of the FMM loop [27]. For a simple presentation, we illustrate it in two dimensions. Let $\mathbf{x}_{ij}$ be a *Close* neighbor of *Trial* to be updated from nearby values: $\tau_{i-1,j}$, $\tau_{i+1,j}$, $\tau_{i,j-1}$, and $\tau_{i,j+1}$. (See Figure 1.) The four nearby points connected by grid line segments form four quadrants; the FMM attempts to solve the quadratic equation given by each quadrant according to

$$\left(\widehat{D}_x \tau_{ij}\right)^2 + \left(\widehat{D}_y \tau_{ij}\right)^2 = \left(s_{ij}\right)^2. \tag{2.7}$$

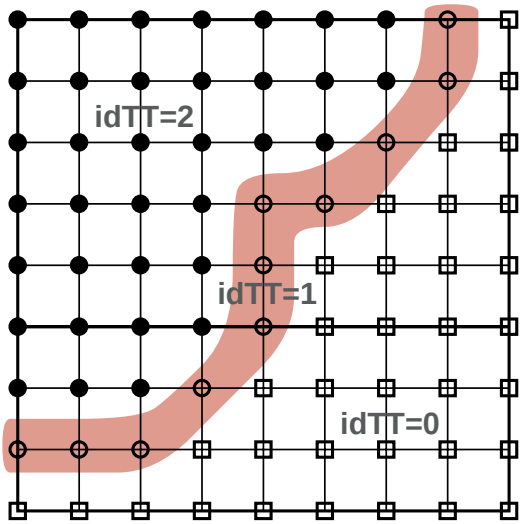


FIG. 2. *The GMM illustrated in two dimensions. The closed circles indicate the points already computed, the open circles form a neighborhood of a wavefront, and the open boxes correspond to the downwind points to be computed at the current stage or later.*

For example, we refer to possible contributors $\tau_{i+1,j}$ and $\tau_{i,j+1}$ (the first quadrant). Without loss of generality, there are two cases.

1. If only $\tau_{i+1,j}$ is known, then we find the larger solution τ of

$$\left(\frac{\tau - \tau_{i+1,j}}{\Delta x}\right)^2 = (s_{ij})^2. \tag{2.8}$$

2. If both $\tau_{i+1,j}$ and $\tau_{i,j+1}$ are known, then we find the larger real solution of

$$\left(\frac{\tau - \tau_{i+1,j}}{\Delta x}\right)^2 + \left(\frac{\tau - \tau_{i,j+1}}{\Delta y}\right)^2 = (s_{ij})^2. \tag{2.9}$$

For each of four quadrants, we construct all possible real solutions τ; we choose the smallest value for the traveltime τ_{ij}.

3. The GMM. A costly component of the narrow band methods is the sorting of the solution in each step of the narrow band of wavefronts, which costs $\mathcal{O}(\log_2 N)$ per grid point. Here we introduce a new narrow band level set algorithm, called the GMM, whose total computation cost is $\mathcal{O}(N)$.

Consider a neighborhood Γ of a wavefront; see the shaded area in Figure 2. (Γ corresponds to *Close* in the FMM.) In the current stage of the GMM, we will select a group of points G out of Γ, recompute the traveltimes at neighboring points of G that are not completed, register the neighboring points as members of Γ if they are not already registered, and finally tag "completed" for the points in G. The group of points should be carefully chosen in such a way that the computed solution does not violate causality, since in our algorithm the traveltimes are not sorted from the smallest to the largest. The main objective in this section is to develop a way to select such a group of points from the narrow band.

In the FMM, the traveltimes on the narrow band are sorted at each stage of the narrow band to find the smallest value, and the traveltime at a point is updated from

the real solutions of eight "quadratic equations" (in three dimensions), each of which associates with a west-east, south-north, and up-down combination of neighboring grid points. The main reason for the sorting and abundant computation is that we do not know where the wavefront is coming from and advancing to. If one wishes to advance the narrow band by a group of points, it is necessary to explicitly incorporate some aspects of wavefront directions into the computation algorithm. Also, it may require updating traveltimes more than once at neighboring downwind points of the group.

We begin with the two-dimensional (2D) case with $h = \Delta x = \Delta y$ for a simple presentation. Recall that in the FMM, the traveltime at a point $\mathbf{x}_{ij}$ is updated by choosing the minimum of the real solutions of four quadratic equations, each of which corresponds to one of four quadrants. So, given neighboring traveltimes, the minimum of the solutions of the quadratic equations must be associated with the quadrant where the raypath is passing before reaching at $\mathbf{x}_{ij}$. Let the first quadrant be in the upwind direction for $\mathbf{x}_{ij}$. (See Figure 1.) Then the updated traveltime at $\mathbf{x}_{ij}$, τ_{ij}, should satisfy

$$\left(\frac{\tau_{ij} - \tau_{i+1,j}}{h}\right)^2 + \left(\frac{\tau_{ij} - \tau_{i,j+1}}{h}\right)^2 = (s_{ij})^2. \tag{3.1}$$

Since τ_{ij} is the larger solution of the above equation,

$$\tau_{ij} \geq \frac{1}{2}(\tau_{i+1,j} + \tau_{i,j+1}). \tag{3.2}$$

Thus it follows from (3.1) and (3.2) that

$$\tau_{ij} \geq \min(\tau_{i+1,j}, \tau_{i,j+1}) + \delta\tau_{ij}, \quad \delta\tau_{ij} = \frac{1}{\sqrt{2}} \cdot h \cdot s_{ij}. \tag{3.3}$$

Defining $s_{\Gamma,\min} = \min\{s_{ij} : \mathbf{x}_{ij} \in \Gamma\}$, we can interpret (3.3) as follows. *Given two points* $\mathbf{x}_{i_1,j_1}$ *and* $\mathbf{x}_{i_2,j_2}$, *if their traveltime difference is less than*

$$\delta\tau \equiv \frac{1}{\sqrt{2}} \cdot h \cdot s_{\Gamma,\min},$$

the angle between the line segment $\overline{\mathbf{x}_{i_1,j_1}\mathbf{x}_{i_2,j_2}}$ *and the wavefront normal is larger than* 45 *degrees, i.e., rather perpendicular than parallel.*

Remark. In three dimensions, $\sqrt{2}$ in (3.3) and $\delta\tau$ should be replaced by $\sqrt{3}$. It can be shown with a physical insight as follows. We begin with the 2D problem. The equality in (3.3) holds only if $\tau_{i+1,j} = \tau_{i,j+1}$, i.e., $\mathbf{x}_{i+1,j}$ and $\mathbf{x}_{i,j+1}$ are on the same wavefront; the distance between the wavefront and $\mathbf{x}_{ij}$ is $h/\sqrt{2}$. When $\tau_{i+1,j} \neq \tau_{i,j+1}$, the traveltime increment at $\mathbf{x}_{ij}$, from $\min(\tau_{i+1,j}, \tau_{i,j+1})$, is clearly larger than $\delta\tau_{ij}$. Therefore, we can claim that $|\tau_{ij} - \min(\tau_{i+1,j}, \tau_{i,j+1})|$ is minimized when the wavefront including $\mathbf{x}_{i+1,j}$ and $\mathbf{x}_{i,j+1}$ (at the same time) reaches the target point $\mathbf{x}_{ij}$, which is the slowness multiplied by the distance between the wavefront and $\mathbf{x}_{ij}$. For the three-dimensional (3D) problem, there is no physical restriction to claim the same; here the distance between the target point and the wavefront including the adjacent three points becomes $h/\sqrt{3}$.

Now we are ready to choose the group G to be *completed* at a time. Define $\tau_{\Gamma,\min} = \min\{\tau_{ij} : \mathbf{x}_{ij} \in \Gamma\}$ and select G as follows:

$$G = \Big\{\mathbf{x}_{ij} \in \Gamma \,:\, \tau(\mathbf{x}_{ij}) \leq \tau_{\Gamma,\min} + \delta\tau\Big\}. \tag{3.4}$$

Let $\mathbf{x}_{i_1,j_1}$ and $\mathbf{x}_{i_2,j_2}$ be two points in G. When they are not adjacent, it is clear that their traveltimes do not affect each other in the update procedure. If they are adjacent, one can barely affect the other, since the wavefront normal is nearer to perpendicular, rather than parallel, to the line segment formed by the points. However, whether the two points are adjacent or not, their neighboring downwind points can be affected by both points. It can be the case, in particular, for intersecting wavefronts, i.e., at or near shocks. The neighboring points may have different traveltimes for a different order of updates, which implies that the group update may not be stable.

To fix instability, we update all the neighboring points of the group G twice—one in an order and the other in the opposite order. The double computation fixes instability. To see it, imagine that we try to update the neighboring points of G one more time. We can readily see that none of the neighboring points changes its traveltime during the extra updates. Of course, it also holds for the 3D problem with $\delta\tau = \frac{1}{\sqrt{3}} \cdot h \cdot s_{\Gamma,\min}$.

Now we summarize the above arguments as in the following algorithm (in three dimensions), called the GMM.

- *Initialization.*
  - (I1) Assign a large number to the traveltime array `TT`, e.g., $\texttt{TT}(\cdot,\cdot,\cdot) \equiv 1.0e5$;
  - (I2) Set zero for the traveltime index `idTT`, i.e., $\texttt{idTT}(\cdot,\cdot,\cdot) \equiv 0$;
  - (I3) Set $\texttt{delTAU} = \frac{1}{\sqrt{3}} \cdot \min(\Delta x, \Delta y, \Delta z) \cdot \min_{i,j,k} s^k_{i,j}$;
  - (I4) On the box of $(2 \times 2 \times 2)$ cells having the source at its center,
    - assign the analytic value of `TT` on the box;
    - set $\texttt{idTT}(\cdot,\cdot,\cdot) = 2$, at the source;
    - set $\texttt{idTT}(\cdot,\cdot,\cdot) \equiv 1$, on the surface of the box; save those point indices to the interface indicator array $\texttt{GAMMA}(\cdot,\cdot,\cdot)$;
    - set `TM` to be the minimum of `TT` on the surface of the box;
- *Marching Forward.*
  - (M1) Set $\texttt{TM} = \texttt{TM} + \texttt{delTAU}$;
  - (M2) For each (i,j,k) in `GAMMA`, in the reverse order, if $(\,\texttt{TT}(i,j,k) \le \texttt{TM}\,)$, recompute traveltimes of neighboring points (ℓ,m,n) where $\texttt{idTT} \le 1$;
  - (M3) For each (i,j,k) in `GAMMA`, in the forward order, if $(\,\texttt{TT}(i,j,k) \le \texttt{TM}\,)$,
    - (a) recompute traveltimes of neighboring points (ℓ,m,n) where $\texttt{idTT} \le 1$;
    - (b) if $\texttt{idTT} = 0$ at a neighboring point (ℓ,m,n), set $\texttt{idTT}(\ell,m,n) = 1$ and save (ℓ,m,n) into `GAMMA`;
    - (c) remove the index (i,j,k) out of `GAMMA`; set $\texttt{idTT}(i,j,k) = 2$;
  - (M4) If $\texttt{GAMMA} \neq \emptyset$, go to (M1);

Remark. Apparently, the reverse computation (M2) is cheaper to carry out than (M3); the double-computation (M2)–(M3) does not increase the computation cost twice. It can be made cheaper as follows. A step of the GMM advances a group of points including not only the global minimum (of the narrow band) but also all local minima that are less than or equal to `TM`. It is not difficult to see if a point in the group is a local minimum of the narrow band. Since it is not necessary to compute twice at local minima, one can modify the algorithm to skip the double-computation there. Let the point (i,j,k) in (M2) be recognized as a local minimum during the recomputation of neighboring downwind points. Then one can update `GAMMA` and `idTT` as in (b) and (c) of (M3); the point (i,j,k) would be skipped by (M3), since it is already out of `GAMMA`. More than half the points in a group seems a local minimum, in practice. The double-computation increases the computation cost, not twice, but

slightly.

Remark. The GMM is in fact an iterative update procedure, converging in two iterations. One may want to select G with a larger $\delta\tau$. In this case, the number of iterations must become larger. Rouy and Tourin [23] has chosen all the grid points as one group and carried out iterations up to convergence. The GMM can be viewed as an intermediate algorithm between the FMM [25, 27] ($\delta\tau = 0$) and the purely iterative algorithm of Rouy and Tourin [23] ($\delta\tau = \infty$).

Remark. When the slowness is constant, two group marchings advance the wavefront to a completely different outer surface. Such a feature can make the GMM more efficient than pointwise methods such as the FMM [27].

4. Average normal slowness. It is well known that the *exact* traveltime from the source $\mathbf{x}_s$ to a location $\mathbf{x}$ can be computed along the raypath

$$\tau(\mathbf{x}_s, \mathbf{x}) = \int_{\mathbf{x}_s}^{\mathbf{x}} \frac{1}{v(\mathbf{x}')} d\sigma = \int_{\mathbf{x}_s}^{\mathbf{x}} s(\mathbf{x}') d\sigma, \tag{4.1}$$

where σ is the length element along the raypath. This implies that accuracy of algorithm (2.5) can be improved by replacing $s_{i,j}^k$ by the *average normal slowness* $\widehat{s_{i,j}^k}$, which is obtained by integrating the slowness along the wavefront normal direction. The wavefront normal, by definition, is $v(\mathbf{x})\nabla\tau$ and is tangent to the raypath for isotropic media.

Velocities are often provided at grid points in practice; the velocity needs to be interpolated over the whole computation domain. For simplicity, we adopt the trilinear spline interpolation in three dimensions; i.e., the velocity is treated as a trilinear function in each cell.

As an example for the computation of the average normal slowness, consider a point $\mathbf{x}_{i,j}^k$ at which the wavefront normal is pointing the $(z+)$-direction. Let us compute $\tau_{i,j}^{k+1}$ out of $\tau_{i,j}^k$ and its adjacent traveltimes $\tau_{i\pm1,j}^k$ and $\tau_{i,j\pm1}^k$. Since $\widehat{D}_x\tau_{i,j}^k$ and $\widehat{D}_y\tau_{i,j}^k$ are likely zero at the point $\mathbf{x}_{i,j}^k$, we can see from (2.6), with $s_{i,j}^k$ replaced by $\widehat{s_{i,j}^k}$, that

$$\tau_{i,j}^{k+1} = \tau_{i,j}^k + \Delta z \cdot \widehat{s_{i,j}^k}, \tag{4.2}$$

where

$$\widehat{s_{i,j}^k} = \frac{1}{\Delta z} \cdot \int_{\mathbf{x}_{i,j}^k}^{\mathbf{x}_{i,j}^{k+1}} \frac{1}{v(\mathbf{x}')} d\sigma = \begin{cases} \dfrac{1}{v_{i,j}^k} = s_{i,j}^k & \text{if } v_{i,j}^{k+1} = v_{i,j}^k, \\ \dfrac{\log(v_{i,j}^{k+1}) - \log(v_{i,j}^k)}{v_{i,j}^{k+1} - v_{i,j}^k} & \text{else.} \end{cases}$$

In realistic media, it is difficult to compute the average normal slowness without raypath information. A reasonable approximation can be simply obtained as follows. For example, for the update from the first quadrant as in (2.8)–(2.9), we first check which one is smaller between $\tau_{i+1,j}$ and $\tau_{i,j+1}$. If $\tau_{i+1,j}$ is smaller, the wavefront normal would have a smaller angle to the line segment $\overline{\mathbf{x}_{i+1,j}\mathbf{x}_{i,j}}$ than to $\overline{\mathbf{x}_{i,j+1}\mathbf{x}_{i,j}}$. So it is reasonable to utilize

$$\widehat{s}_{i,j} \approx \begin{cases} s_{i,j} & \text{if } s_{i+1,j} = s_{i,j}, \\ \dfrac{\log(v_{i+1,j}) - \log(v_{i,j})}{v_{i+1,j} - v_{i,j}} & \text{else.} \end{cases} \tag{4.3}$$

When $\tau_{i,j+1}$ is smaller, the index $(i+1,j)$ in (4.3) should be replaced by $(i,j+1)$.

The practical accuracy of numerical algorithms for traveltimes seems strongly dependent on the ability of the numerical code to compute an *average slowness* close to the average normal slowness. No matter what the order of the numerical scheme is, the solution would turn out to have a first-order accuracy in general media if the point slowness is incorporated. (In principle, the energy does not propagate jumping over discrete points but advances their fronts through all the points in the medium.) Another degraded accuracy problem can appear at singularities such as the source points and caustics; it can be effectively treated by employing numerical techniques such as the locally uniform mesh refinement [12] and adaptive gridding approach [20].

5. An alternative update procedure in two dimensions. In this section, we consider an alternative update formula for the 2D problem which is slightly different from (2.8)–(2.9). We first define it as follows. To update $\tau_{i,j}$, choose the minimum from the four values

$$\begin{aligned} &\tau_{i\pm1,j} + \Delta x \cdot \max\Big(\alpha\widehat{s}_{ij},\ \sqrt{(\widehat{s}_{ij})^2 - (\widehat{D}_z\tau_{i\pm1,j})^2}\Big), \\ &\tau_{i,j\pm1} + \Delta z \cdot \max\Big(\beta\widehat{s}_{ij},\ \sqrt{(\widehat{s}_{ij})^2 - (\widehat{D}_x\tau_{i,j\pm1})^2}\Big), \end{aligned} \tag{5.1}$$

where

$$\alpha = \frac{\Delta x}{\sqrt{(\Delta x)^2+(\Delta z)^2}}, \quad \beta = \frac{\Delta z}{\sqrt{(\Delta x)^2+(\Delta z)^2}}.$$

Here the maximization with $\alpha\widehat{s}_{ij}$ or $\beta\widehat{s}_{ij}$ is incorporated so as not to violate stability.

The above update procedure utilizes delayed traveltime derivatives (purely forward), and therefore it is more efficient in implementation and performance than (2.8)–(2.9). However, it may be *unstable* without the maximization. For example, ignoring the factor $\alpha\widehat{s}_{ij}$, the updating value obtainable from the west (the direction of $\tau_{i-1,j}$) becomes

$$\tau_{i-1,j} + \Delta x \cdot \sqrt{(\widehat{s}_{ij})^2 - (\widehat{D}_z\tau_{i-1,j})^2}.$$

Consider the point $(i,j)=(1,2)$ in Figure 3. Since, apparently, $\widehat{D}_x\tau_{i-1,j-1}=0$ for a constant slowness, we see $(\widehat{s}_{ij})^2 = (\widehat{D}_z\tau_{i-1,j})^2$. So the updated final value of τ_{ij} would become at most $\tau_{i-1,j}$, which is wrong (an underestimation)!

The maximization is equivalent to imposing the *maximum angle condition* which takes care of only the rays coming to the point $\mathbf{x}_{ij}$ through the *numerical domain of dependence*. The maximum angle condition was first introduced in [8] and was adopted by Symes and his colleagues [13, 20]; see also [12]. When τ_{ij} is to be updated from the west, the numerical domain of dependence is the line segment $\overline{\mathbf{x}_{i-1,j-1}\mathbf{x}_{i-1,j+1}}$, and every ray approaching from the line segment to $\mathbf{x}_{ij}$ increases the traveltime τ_{ij} (from $\tau_{i-1,j}$) by $\alpha\widehat{s}_{ij}$ at minimum.

The purely forward update procedure (5.1) is as accurate as (2.8)–(2.9) in practical computation. We will see it here for a constant slowness and in section 6 for general media by numerical simulation. Let $h=\Delta x=\Delta z$ and $s(\mathbf{x})\equiv s_0$ in Figure 3. Let the traveltimes on the narrow band have been computed accurately. Let us try to compute $\tau_{1,2}$, whose analytic value is $\sqrt{5}hs_0 \approx 2.236\cdot hs_0$. First, we utilize the formula (2.8)–(2.9):

$$(\tau_{1,2}-\tau_{0,2})^2 + (\tau_{1,2}-\tau_{1,1})^2 = h^2 s_0^2. \tag{5.2}$$

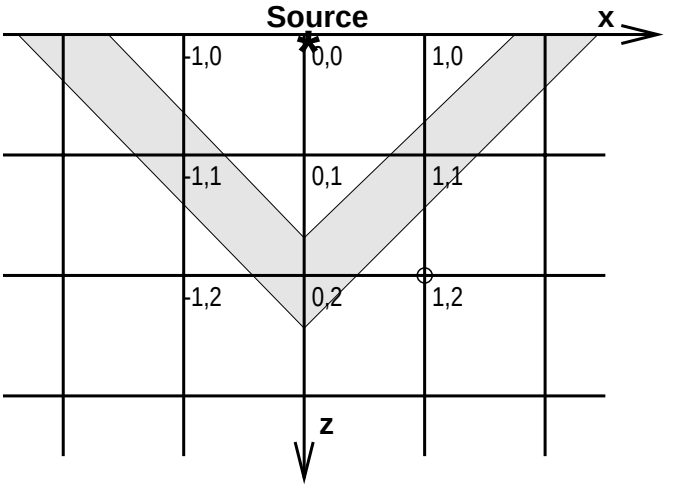


FIG. 3. *The narrow band method in two dimensions. The slowness is assumed to be constant. The shaded area denotes the narrow band at a certain moment. A point source is located at the origin marked by an asterisk.*

Replacing $\tau_{0,2}$ and $\tau_{1,1}$ by $2hs_0$ and $\sqrt{2}hs_0$, respectively, and solving the quadratic equation, we have

$$\tau_{1,2} \approx 2.351 \cdot hs_0,$$

which is the best value for $\tau_{1,2}$ from all former and other updates of (2.8)–(2.9). Now we apply (5.1):

$$\begin{aligned} \tau_{1,2} &= \tau_{1,1} + h \cdot \max\left(\tfrac{1}{\sqrt{2}}s_0,\ \sqrt{s_0^2 - (\widehat{D}_x\tau_{1,1})^2}\right), \\ \tau_{1,2} &= \tau_{0,2} + h \cdot \max\left(\tfrac{1}{\sqrt{2}}s_0,\ \sqrt{s_0^2 - (\widehat{D}_z\tau_{0,2})^2}\right). \end{aligned} \tag{5.3}$$

Again, substituting the analytic values, we have the minimum

$$\tau_{1,2} \approx 2.324 \cdot hs_0,$$

which slightly improves accuracy over (2.8)–(2.9). The above argument can be applied to every point.

The main reason for the introduction of (5.1) is not for such a slight improvement in accuracy, but for more efficient incorporation of the average normal slowness. Note that the formula (5.1) incorporates ray directions more explicitly than (2.8)–(2.9). Since chasing raypaths is expensive, an approximation can be utilized as follows. For example, for the update from the west, an approximate average slowness can be found as

$$\widehat{s}_{i,j} \approx \begin{cases} s_{ij} & \text{if } s_{i-1,j} = s_{i,j}, \\ \dfrac{\log(v_{i-1,j}) - \log(v_{i,j})}{v_{i-1,j} - v_{i,j}} & \text{else.} \end{cases} \tag{5.4}$$

The above is compared with (4.3) in the same line. In heterogeneous media, the above average slowness produces more accurate traveltimes with (5.1) than with (2.8)–(2.9); see numerical results in section 6.

TABLE 1
Accuracy and efficiency of the GMM in two dimensions. A point source is located at $(x, z) = (3000, 0)$.

		Point slowness				Average slowness			
		(2.8)–(2.9)		(5.1)		(2.8)–(2.9)		(5.1)	
v	M^2	CPU	$E(\tau^h)$	CPU	$E(\tau^h)$	CPU	$E(\tau^h)$	CPU	$E(\tau^h)$
v_1	100^2	0.13	4.6e-2	0.09	3.5e-2	0.15	3.4e-2	0.11	2.1e-2
	200^2	0.52	2.5e-2	0.37	1.8e-2	0.62	1.9e-2	0.44	8.2e-3
	400^2	2.14	1.3e-2	1.44	9.3e-3	2.43	1.0e-2	1.78	3.6e-3
v_2	100^2	0.14	3.6e-2	0.09	2.3e-2	0.17	3.5e-2	0.12	1.5e-2
	200^2	0.52	2.0e-2	0.35	1.1e-2	0.69	1.9e-2	0.51	7.2e-3
	400^2	2.10	1.1e-2	1.43	5.9e-3	2.71	1.0e-2	2.04	3.5e-3

Remark. In three dimensions, the numerical domain of dependence forms a solid quadrilateral whose edges have the same length. Since six of these surfaces cannot surround the point $\mathbf{x}_{ij}^k$ to be updated (so instability can happen), some auxiliary difference formulas should be introduced including the points diagonal from $\mathbf{x}_{ij}^k$ such that the union of numerical domains of dependence is all around in every direction to the target point. For the case $\Delta x = \Delta y = \Delta z$, it is not difficult to introduce such FDs. However, it would be very complicated for general cases. Alternatively, one may combine (2.8)–(2.9), the cell-oriented version, and (5.1), the line-oriented version.

6. Numerical experiments. The GMM in section 3 is implemented in two and three dimensions for the FATTs of (2.1). Set the domain $\Omega = (0, 6000\,\mathrm{m})^d$, $d = 2, 3$. We consider four different velocity models: for $\mathbf{x} \in \Omega$,

$$
\begin{aligned}
v_1(\mathbf{x}) &= 1000 + z \text{ m/s}, \\
v_2(\mathbf{x}) &= 1000 + 0.2x + 0.5z \text{ m/s}, \\
v_3(\mathbf{x}) &= 1000 + 0.3x + 0.2y + 0.4z \text{ m/s}, \\
v_4(\mathbf{x}) &= \begin{cases} 4500 \text{ m/s} & \text{if } \mathbf{x} \in [1500, 4500]^d, \\ 2000 \text{ m/s} & \text{else.} \end{cases}
\end{aligned} \tag{6.1}
$$

(For 2D cases, the y-components are dropped.) The domain is partitioned with $h = \Delta x = \Delta y = \Delta z = 6000/M$ for $M > 0$. Point sources are imposed inside or on the surface of the domain. The traveltimes in linear velocities can be computed analytically; the numerical error is measured as

$$E(\tau^h) = \|\tau^h - \tau_{\text{analytic}}\|_\infty,$$

where τ^h denotes the computed traveltime with a grid size h. The average normal slowness is approximated by (4.3) and (5.4).

The main/driver routines are written in C++, and the core computation routines are in F77. The computation is carried out on a Gateway Solo, a 266 MHz laptop having 128M memory and a Linux operating system. The elapsed time CPU is the user time measured in seconds.

Table 1 compares the accuracy and efficiency of the GMM in two dimensions between the point slowness and the average slowness, and between the quadratic formula (2.8)–(2.9) and the purely forward formula (5.1). Two different velocities (v_1 and v_2) are selected here, and a point source is located at $(x, z) = (3000, 0)$. As one can see from the table, incorporating the average slowness increases about 20–25% computation cost, while the formula (5.1) decreases about 30% the cost of (2.8)–(2.9). The velocity v_1 has a larger variance (in particular, in the vertical direction)

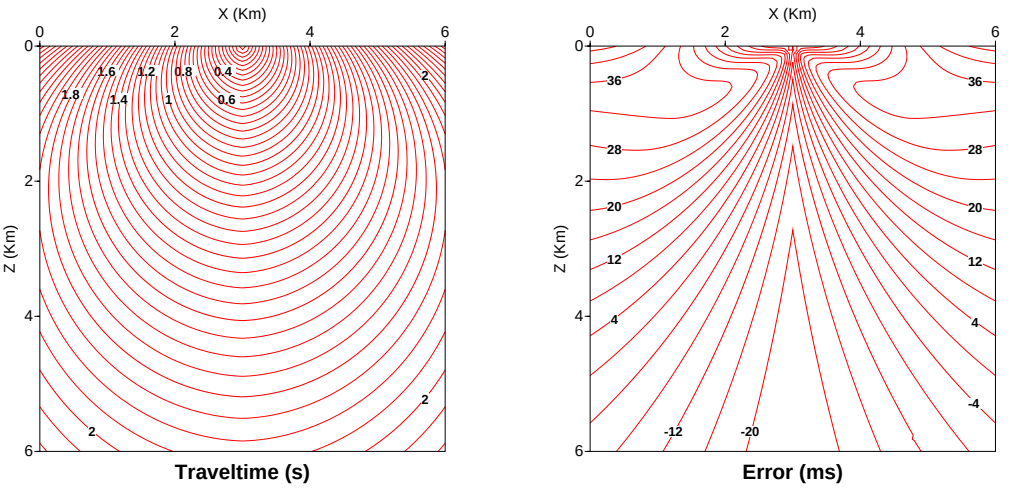


FIG. 4. *The computed traveltime (in seconds) and the error* $\tau^h - \tau_{\text{analytic}}$ *(in milliseconds). Set* $v = v_1$*, and the point source is located at* $(x, z) = (3000, 0)$*. The point slowness is incorporated with the updated formula* (2.8)–(2.9).

than v_2; the formula (5.1) incorporating the average slowness has improved accuracy more dramatically for v_1 than for v_2. We can see from the last panel of the table that its accuracy is slightly better than linear, even measured in the maximum norm. Such *superlinear convergence* has been observed for most linear velocity models. The following should also be noticed.

- The computation cost of the GMM is $\mathcal{O}(N)$ for any choice of slowness and update formula.
- The formula (5.1) is more efficient and accurate than (2.8)–(2.9), whether the slowness is averaged or not.
- The average slowness better improves accuracy when incorporated with the purely forward formula (5.1).

Here we have employed a roughly averaged slowness; however, the numerical error has decreased by a factor of two to three. When the velocity is highly oscillatory or when a higher-order FD scheme is employed, the average slowness is essential to incorporate. The point slowness will impose a first-order accuracy for heterogeneous media, no matter how accurate the FD scheme is.

Figure 4 depicts the computed traveltime (in seconds) and the error $\tau^h - \tau_{\text{analytic}}$ (in milliseconds) for the example discussed in Table 1 ($v = v_1$). The point slowness is incorporated with the update formula (2.8)–(2.9). The error (right side) indicates that the method is both over- and underestimating. Overestimation is not a danger but just an accuracy problem; it can be improved through the minimizing process using more available updating values. On the other hand, underestimation has no way to be fixed.

In Figure 5, we present the error $\tau^h - \tau_{\text{analytic}}$ (in milliseconds) for the point slowness (left) and the average slowness (right), both incorporated with the update formula (5.1). As in Figure 4, $v = v_1$, and the point source is located at $(x, z) = (3000, 0)$. No underestimation has been observed for the formula (5.1), as one can see from these pictures. The error for the average slowness is smaller and smoother than that of the

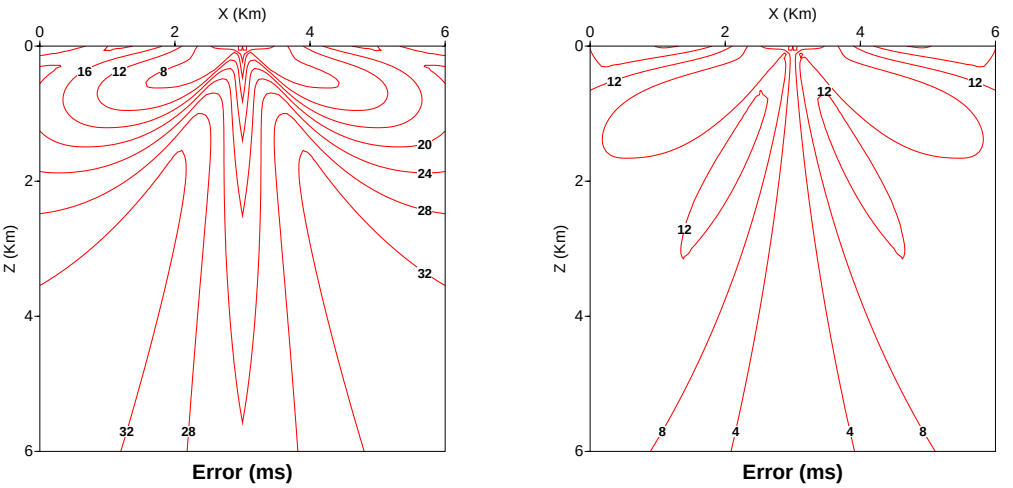


FIG. 5. *The error* $\tau^h - \tau_{\text{analytic}}$ *(in milliseconds) for the point slowness (left) and the average slowness (right). For the update formula,* (5.1) *is utilized. Set* $v = v_1$*, and the point source is located at* $(x, z) = (3000, 0)$.

TABLE 2

Accuracy and efficiency of the FMM and GMM in three dimensions. Set $v = v_3$*, and locate a point source at* $(x, y, z) = (3000, 3000, 1000)$*. The updated formula* (2.8)–(2.9) *is applied.*

	FMM				GMM			
	Point slowness		Average slowness		Point slowness		Average slowness	
M^3	CPU	$E(\tau^h)$	CPU	$E(\tau^h)$	CPU	$E(\tau^h)$	CPU	$E(\tau^h)$
30^3	1.37	1.2e-1	2.04	1.0e-1	0.98	1.2e-1	1.14	8.2e-2
60^3	19.81	6.2e-2	27.62	5.5e-2	7.88	6.2e-2	9.13	4.5e-2
120^3	395.8	3.2e-2	460.5	2.8e-2	64.06	3.2e-2	73.73	2.3e-2

point slowness. Note that the ray direction is vertical on the line segment $\{x = 3000\}$, where the solution incorporating the average slowness is computed without error. (It also can be figured out mathematically.)

In Table 2, we show numerical results for the FMM and GMM in three dimensions. The velocity is chosen as $v = v_3$, and a point source is located at $(x, y, z) = (3000, 3000, 1000)$. The update formula (2.8)–(2.9) is applied. We observe in 3D simulation that the computation cost of the FMM increases more rapidly than the problem size, while GMM costs $\mathcal{O}(N)$. One can easily expect that the GMM is three to five times faster than the FMM for problems of reasonable sizes. It should be noticed that the average slowness improves accuracy better for the GMM.

In Figure 6, we depict the computed traveltime of the GMM in three dimensions superimposed on the cross section $\{y = 3000\}$ of the velocity model $v = v_4$. The number of grid points is 100^3 and the source is located at $(x, y, z) = (1000, 3000, 1000)$. The update formula (2.8)–(2.9) is utilized incorporating the average slowness; the GMM takes 39.89 seconds. There one can see the shocks developed during the advancement of the wavefronts, due to the headwave out from the high velocity region.

7. Discussion. The GMM is yet to be tested for realistic models, in particular, in three dimensions. Since the performance of the GMM is weakly dependent on the

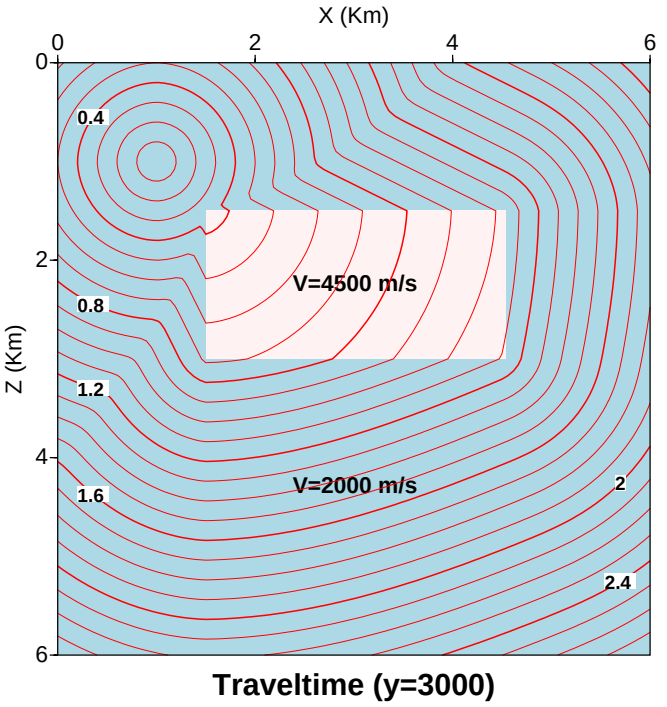


FIG. 6. *The computed traveltime of the GMM in three dimensions superimposed on the cross section* $\{y = 3000\}$ *of the velocity* $v = v_4$. *The number of grid points is* 100^3, *and the source is located at* $(x, y, z) = (1000, 3000, 1000)$.

velocity model, as one can see from Table 1, we can conclude that a 3D problem of a million unknowns can be solved in 40 seconds on a 266 MHz laptop. Nonetheless, I would like to discuss some important aspects on the accuracy, efficiency, and applicability of the level set methods, both the FMM [25, 27] and the GMM.

Higher-order extension. The level set methods are hard to incorporate high-order FD schemes. Sethian [27] suggested one-sided high-order schemes to be considered whenever the traveltimes at the corresponding points are available. Here one should notice that one-sided high-order FD schemes can produce a less accurate solution than the first-order scheme, in particular, in heterogeneous media. Even if the medium is constant, the one-sided second-order scheme easily produces underestimated solutions, as one can see from [27]. An interested reader can also check it for the computation of $\tau_{1,2}$ in Figure 3 using the second-order version of (5.2) presented in [27]. (Your result would be approximately $2.205 \cdot hs_0$; the exact value is $2.236 \cdot hs_0$.)

Computation cost. Now the level set method can be carried out for the computation cost of $\mathcal{O}(N)$ instead of $\mathcal{O}(N \log N)$. However, both the FMM and the GMM access the data (such as the solution, the narrow band points, and the binary tree (FMM)) in an almost random manner, rather than a systematic way along the loop indices. As a consequence, the algorithms can be more expensive than expected, depending on the computers adopted.

Applicability to the traveltime computation in seismology. Seismic image processing methods require an accurate computation of traveltimes of acoustic or elastic waves. When a set of seismic survey data is acquired utilizing the main frequency of 40Hz, for example, the period for the propagation of one wavelength is 25 milliseconds. For a reasonable image processing, the error should be much less than 25 milliseconds over the whole domain whose edge lengths are often 4,000 m to 9,000 m. For the example in Table 2, one has to choose approximately 300 points ($h = 20$ m) in each direction to keep the error below 10 milliseconds; the expected computation time is 16 to 19 minutes in the same machine (ignoring the memory problem), which is extremely expensive. Note that for a section of seismic image, the traveltimes are often computed for more than 10,000 shot-gathers. (If a machine of the same speed tested here is used, it will take at least four months for the traveltime computation.) See [8, 9, 10, 12, 13, 21, 22, 31] for numerical methods and related issues in the traveltime computation and seismic image processing.

Overall, the level set methods should be far more improved (particularly, in accuracy) for a wider range of applications. The update formula discussed at the end of section 5 for the 3D problem, a second-order FD scheme applicable to heterogeneous media, and applications to anisotropic media will be the subjects of a forthcoming paper [11].

8. Conclusions. An optimal level set method called the GMM has been suggested, based on a physical investigation for the solution on the narrow band of wavefronts. It has been applied to the computation of the FATTs of the eikonal equation in two and three dimensions. The GMM incorporates both the narrow band approach [25, 27] and the iterative scheme [23]; it converges in two iterations for a group of points in the narrow band. Strategies have been discussed to reduce the cost of the double-computation. We have introduced the average normal slowness to improve accuracy; its approximation is discussed for an efficient implementation. A new alternative update formula is suggested for the 2D problem, and it is numerically verified to be superior to that of the conventional level set methods in both efficiency and accuracy. For various velocities, grid sizes, and update formulae, the GMM has carried out its job for the cost $\mathcal{O}(N)$, where N is the total number of grid points. The GMM performs three to five times faster than the FMM for reasonable-size problems in three dimensions.

Acknowledgments. The author expresses his sincere thanks to Professor S. Osher, UCLA, for his interests and valuable comments. Constructive comments by two anonymous reviewers improved the paper and are much appreciated.

REFERENCES

[1] V. Cerveny, *Ray synthetic seismograms for complex two- and three-dimensional structures*, J. Geophys., 58 (1985), pp. 2–26.

[2] V. Cerveny, I. A. Molotkov, and I. Psencik, *Ray Methods in Seismology*, University of Karlova Press, Prague, 1977.

[3] M. Crandall and P. Lions, *Viscosity of solutions of Hamilton-Jacobi equations*, Trans. Amer. Math. Soc., 277 (1983), pp. 1–42.

[4] M. Crandall and P. Souganidis, *Developments in the theory of nonlinear first-order partial differential equations*, in Differential Equations, W. Knowles and R. Lewis, eds., North-Holland Math. Stud. 92, North-Holland, Amsterdam, 1984, pp. 131–143.

[5] J. Dellinger, *Anisotropic finite-difference traveltimes*, in Expanded Abstracts, Society of Exploration Geophysicists, Tulsa, OK, 1991, pp. 1530–1533.

[6] J. Dellinger and W. W. Symes, *Anisotropic finite-difference traveltimes using a Hamilton-Jacobi solver*, in Expanded Abstracts, Society of Exploration Geophysicists, Tulsa, OK, 1997, pp. 1786–1789.
[7] F. Friedlander, *Sound Pulses*, Cambridge University Press, Cambridge, UK, 1958.
[8] S. Gray and W. May, *Kirchhoff migration using eikonal equation traveltimes*, Geophysics, 59 (1994), pp. 810–817.
[9] S. Kim, *ENO-DNO-PS: A stable, second-order accuracy eikonal solver*, in Expanded Abstracts, Society of Exploration Geophysicists, Tulsa, OK, 1999, pp. 1747–1750.
[10] S. Kim, *On eikonal solvers for anisotropic traveltimes*, in Expanded Abstracts, Society of Exploration Geophysicists, Tulsa, OK, 1999, pp. 1875–1878.
[11] S. Kim, *An $\mathcal{O}(N)$ Second-Order Level Set Method*, in preparation, 2000.
[12] S. Kim and R. Cook, *3D traveltime computation using second-order ENO scheme*, Geophysics, 64 (1999), pp. 1867–1876.
[13] S. Kim, W. Symes, and M. A. El-Mageed, *Superconvergent difference formulas for traveltimes and amplitudes*, in Mathematical and Numerical Aspects of Wave Propagation, J. A. DeSanto, ed., SIAM, Philadelphia, 1998, pp. 591–593.
[14] P. Lax, *Hyperbolic Systems of Conservation Laws and the Mathematical Theory of Shock Waves*, SIAM, Philadelphia, 1973.
[15] P. L. Lions, *Generalized Solutions of Hamilton-Jacobi Equations*, Res. Notes Math. 69, Pitman, New York, 1982.
[16] R. Malladi and J. A. Sethian, *An $\mathcal{O}(N \log N)$ algorithm for shape modeling*, Proc. Natl. Acad. Sci. USA, 93 (1996), pp. 9389–9392.
[17] S. Osher and J. A. Sethian, *Fronts propagating with curvature dependent speed: Algorithms based on Hamilton-Jacobi formulations*, J. Comput. Phys., 79 (1988), pp. 12–49.
[18] S. Osher and C.-W. Shu, *High-order essentially nonoscillatory schemes for Hamilton–Jacobi equations*, SIAM J. Numer. Anal., 28 (1991), pp. 907–922.
[19] A. Popovici and J. A. Sethian, *Three dimensional traveltime computation using the fast marching method*, in Expanded Abstracts from the 67th Annual International Meeting of Society of Exploration Geophysicists, Dallas, TX, 1997, Society of Exploration Geophysicists, Tulsa, OK, 1997, pp. 1778–1781.
[20] J. Qian, C. Belfi, and W. Symes, *Adaptive finite-difference method for traveltime and amplitude*, in Expanded Abstracts, Society of Exploration Geophysicists, Tulsa, OK, 1999, pp. 1763–1766.
[21] F. Qin, Y. Luo, K. B. Olsen, W. Cai, and G. T. Schuster, *Finite-difference solution of the eikonal equation along expanding wavefronts*, Geophysics, 57 (1992), pp. 478–487.
[22] M. Reshef and D. Kosloff, *Migration of common shot gathers*, Geophysics, 51 (1986), pp. 324–331.
[23] E. Rouy and A. Tourin, *A viscosity solutions approach to shape-from-shading*, SIAM J. Numer. Anal., 29 (1992), pp. 867–884.
[24] W. A. J. Schneider, K. A. Ranzinger, A. H. Balch, and C. Kruse, *A dynamic programming approach to first arrival traveltime computation in media with arbitrarily distributed velocities*, Geophysics, 57 (1992), pp. 39–50.
[25] J. A. Sethian, *A fast marching level set method for monotonically advancing fronts*, Proc. Natl. Acad. Sci. USA, 93 (1996), pp. 1591–1595.
[26] J. A. Sethian, *Theory, algorithms, and applications of level set methods for propagating interfaces*, Acta Numerica, 1996 Acta Numer. 5, Cambridge University Press, Cambridge, UK, 1996, pp. 309–395.
[27] J. A. Sethian, *Fast marching methods*, SIAM Rev., 41 (1999), pp. 199–235.
[28] J. A. Sethian and A. Popovici, *3D traveltime computation using the fast marching method*, Geophysics, 64 (1999), pp. 516–523.
[29] J. Tsitsiklis, *Efficient algorithms for globally optimal trajectories*, IEEE Trans. Automat. Control, 40 (1995), pp. 1528–1538.
[30] J. van Trier and W. W. Symes, *Upwind finite-difference calculation of travel-times*, Geophysics, 56 (1991), pp. 812–821.
[31] J. E. Vidale, *Finite-difference calculation of travel times*, Bull. Seismol. Soc. Amer., 78 (1988), pp. 2062–2076.
[32] J. E. Vidale, *Finite difference calculation of traveltimes in three dimensions*, Geophysics, 55 (1990), pp. 521–526.

SIAM J. SCI. COMPUT.
Vol. 22, No. 6, pp. 2194–2215
© 2001 Society for Industrial and Applied Mathematics

A SCALABLE PARALLEL ALGORITHM FOR INCOMPLETE FACTOR PRECONDITIONING*

DAVID HYSOM† AND ALEX POTHEN†

Abstract. We describe a parallel algorithm for computing incomplete factor (ILU) preconditioners. The algorithm attains a high degree of parallelism through graph partitioning and a two-level ordering strategy. Both the subdomains and the nodes within each subdomain are ordered to preserve concurrency. We show through an algorithmic analysis and through computational results that this algorithm is scalable. Experimental results include timings on three parallel platforms for problems with up to 20 million unknowns running on up to 216 processors. The resulting preconditioned Krylov solvers have the desirable property that the number of iterations required for convergence is insensitive to the number of processors.

Key words. incomplete factorization, ILU, preconditioning, parallel preconditioning

AMS subject classifications. 65F50, 65F10, 68W10, 68R10

PII. S1064827500376193

1. Introduction.

Incomplete factorization (ILU) preconditioning is currently among the most robust techniques employed to improve the convergence of Krylov space solvers for linear systems of equations. (ILU stands for incomplete LU factorization, where L and U are the lower and upper triangular (incomplete) factors of the coefficient matrix.) However, scalable parallel algorithms for computing ILU preconditioners have not been available despite the fact that they have been used for more than twenty years [12]. We report the design, analysis, implementation, and computational evaluation of a parallel algorithm for computing ILU preconditioners.

Our parallel algorithm assumes that three requirements are satisfied.

- The adjacency graph of the coefficient matrix (or the underlying finite element or finite difference mesh) must have good edge separators, i.e., it must be possible to remove a small set of edges to divide the problem into a collection of subproblems that have roughly equal computational work requirements.
- The size of the problem must be sufficiently large relative to the number of processors so that the work required by the subgraph on each processor is suitably large to dominate the work and communications needed for the boundary nodes.
- The subdomain intersection graph (to be defined later) should have a small chromatic number. This requirement will ensure that the dependencies in factoring the boundary rows do not result in undue losses in concurrency.

An outline of the paper is as follows. In section 2, we describe the steps in the parallel algorithm for computing the ILU preconditioner in detail and provide theoretical justification. The algorithm is based on an incomplete fill path theorem; the proof and discussion of the theorem are deferred to an appendix. We also discuss

*Received by the editors August 4, 2000; accepted for publication (in revised form) December 17, 2000; published electronically April 26, 2001. This work was supported by U. S. National Science Foundation grants DMS-9807172 and ECS-9527169, by the U. S. Department of Energy under subcontract B347882 from the Lawrence Livermore Laboratory, by a GAANN fellowship from the Department of Education, and by NASA under contract NAS1-19480 while the authors were in residence at ICASE.

http://www.siam.org/journals/sisc/22-6/37619.html

†Old Dominion University, Norfolk, VA 23529-0162 and ICASE, NASA Langley Research Center, Hampton VA 23681-2199 (hysom@cs.odu.edu, pothen@cs.odu.edu).

the role that a subdomain graph constraint plays in the design of the algorithm, show that the preconditioners exist for special classes of matrices, and relate our work to earlier work on this problem. Section 3 contains an analysis that shows that the parallel algorithm is scalable for two-dimensional (2D-) and three-dimensional (3D-)model problems, when they are suitably ordered and partitioned. Section 4 contains computational results on Poisson and convection-diffusion problems. The first subsection shows that the parallel ILU algorithm is scalable on three parallel platforms; the second subsection reports convergence studies. We tabulate how the number of Krylov solver iterations and the number of entries in the preconditioner vary as a function of the preconditioner level for three variations of the algorithm. The results show that fill levels higher than one are effective in reducing the number of iterations; the number of iterations is insensitive to the number of subdomains; and the subdomain graph constraint does not affect the number of iterations while it makes possible the design of a simpler parallel algorithm.

The background needed for ILU preconditioning may be found in several books; see, e.g., [1, 15, 17, 33]. A preliminary version of this paper was presented at Supercomputing '99 and was published in the conference proceedings [18]. The algorithm has been revised, additional details have been included, and the proof of the theorem on which it is based has been added. The experimental results in section 4 are new, and most of them have been included in the technical reports [19, 20].

2. Algorithms. In this section we discuss the Parallel ILU (PILU) algorithm and its underlying theoretical foundations.

2.1. The PILU algorithm. Figure 2.1 describes the steps of the PILU algorithm at a high level; the algorithm is suited for implementation on both message-passing and shared-address space programming models.

The PILU algorithm consists of four major steps. In the first step, we create parallelism by dividing the problem into subproblems by means of graph partitioning. In the second step, we preserve the parallelism in the interior of the subproblems by locally scheduling the computations in each subgraph. In the third step, we preserve parallelism in the boundaries of the subproblems by globally ordering the subproblems through coloring a suitably defined graph. In the final step, we compute the preconditioner in parallel. Now we will describe the four steps in greater detail.

Step 1: Graph partitioning. In the first step of PILU, we partition the adjacency graph $G(A)$ of the coefficient matrix A into p subgraphs by removing a small set of edges that connects the subgraphs to each other. Each subgraph will be mapped to a distinct processor that will be responsible for the computations associated with the subgraph.

An example of a model five-point grid partitioned into four subgraphs is shown in Figure 2.2. For clarity, the edges corresponding to the coefficient matrix elements (within each subgraph or between subgraphs) are not shown. The edges drawn correspond to fill elements (elements that are zero in the coefficient matrix but are nonzero in the incomplete factors) that join the different subgraphs.

To state the objective function of the graph partitioning problem, we need to introduce some terminology. An edge is a *separator edge* if its endpoints belong to different subgraphs. A vertex in a subgraph is an *interior vertex* if all of its neighbors belong to that subgraph; it is a *boundary vertex* if it is adjacent to one or more vertices belonging to another subgraph. By definition, an interior vertex in a subgraph is not adjacent to a vertex (boundary or interior) in another subgraph. In Figure 2.2, the first 25 vertices are interior vertices of the subgraph S_0, and vertices numbered 26 through 36 are its

Input: A coefficient matrix, its adjacency graph, and the number of processors p.
Output: The incomplete factors of the coefficient matrix.

1. Partition the adjacency graph of the matrix into p subgraphs (subdomains), and map each subgraph to a processor. The objectives of the partitioning are that the subgraphs should have roughly equal work, and there should be few edges that join the different subgraphs.
2. On each subgraph, locally order interior nodes first, and then order boundary nodes.
3. Form the subdomain intersection graph corresponding to the partition, and compute an approximate minimum vertex coloring for it. Order subdomains according to color classes.
4. Compute the incomplete factors in parallel.
   a. Factor interior rows of each subdomain.
   b. Receive sparsity patterns and numerical values of the nonzeros of the boundary rows of lower-numbered subdomains adjacent to a subdomain (if any).
   c. Factor boundary rows in each subdomain and send the sparsity patterns and numerical values to higher-numbered neighboring subdomains (if any).

FIG. 2.1. *High level description of the PILU algorithm.*

boundary vertices. The goal of the partitioning is to keep the amount of work associated with the incomplete factorization of each subgraph roughly equal, while keeping the communication costs needed to factor the boundary rows as small as possible.

There is a difficulty with modeling the communication costs associated with the boundary rows. In order to describe this difficulty, we need to relate this cost more precisely to the separators in the graph. Define the *higher degree* of a vertex v as the number of vertices numbered higher than v in a given ordering. We assume that upward-looking, row-oriented factorization is used. At each boundary between two subgraphs, elements need to be communicated from the lower numbered subgraph to the higher numbered subgraph. The number of these elements is proportional to the sum of the higher degrees (in the filled graph $G(F)$) of the boundary vertices in the lower numbered subgraph. But unfortunately, we do not know the fill edges at this point since we have neither computed an ordering of $G(A)$ nor computed a symbolic factorization. We could approximate by considering higher degrees of the boundary vertices in the graph $G(A)$ instead of the filled graph $G(F)$, but even this requires us to order the subgraphs in the partition.

The union of the boundary vertices on all the subgraphs forms a *wide vertex separator*. This means that the shortest path from an interior vertex in any subgraph to an interior vertex in another subgraph consists of at least three edges; such a path has *length* at least three. The communication cost in the (forward and backward) triangular solution steps is proportional to the sum of the sizes of the wide vertex separators. None of the publicly available graph partitioning software has the minimization of wide separators as its objective function, but it is possible to modify existing software to optimize this objective.

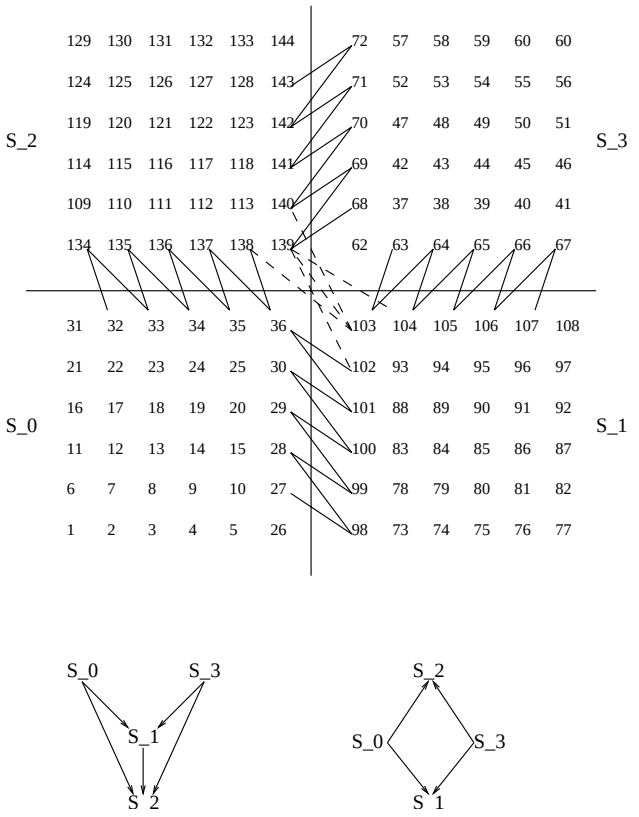


FIG. 2.2. *An example that shows the partitioning, mapping, and vertex ordering used in the PILU algorithm. The graph on the top is a regular* 12×12 *grid with a five-point stencil partitioned into four subdomains and then mapped on four processors. The subdomains are ordered by a coloring algorithm to reduce dependency path lengths. Only the level one and two fill edges that join the different subdomains are shown; all other edges are omitted for clarity. The figure on the bottom right shows the subdomain intersection graph when the subdomain graph constraint is enforced. (This prohibits fill between the boundary nodes of the subdomains* S_1 *and* S_2*, indicated by the broken edges in the top graph.) The graph on the bottom left shows the subdomain intersection graph when the subdomain graph constraint is not enforced.*

The goal of the partitioning step is to keep the amount of work associated with each subgraph roughly equal (for load balance) while making the communication costs due to the boundaries as small as possible. As the previous two paragraphs show, modeling the communication costs accurately in terms of edge and vertex separators in the initial graph $G(A)$ is difficult, but we could adopt the minimization of the wide separator sizes as a reasonable goal. This problem is NP-complete, but there exist efficient heuristic algorithms for partitioning the classes of graphs that occur in practical situations. (Among these graph classes are 2D-finite element meshes and 3D-meshes with good aspect ratios.)

Step 2: Local reordering. In the second step, in each subgraph we order the interior vertices before the boundary vertices. This ordering ensures that during the incomplete factorization, an interior vertex in one subgraph cannot be joined by a fill edge to a vertex in another subgraph, as will be shown later. Fill edges between two subgraphs can join only their boundary vertices together. Thus interior vertices corresponding to the initial graph $G(A)$ remain interior vertices in the graph of the

factor $G(F)$. The consequences of this are that the rows corresponding to the interior vertices in each subdomain of the initial problem $G(A)$ can be factored concurrently, and that communication is required only for factoring rows corresponding to the boundary rows. The reader can verify that in each subgraph in Figure 2.2 the interior nodes have been ordered before the boundary nodes.

The observation concerning fill edges in the preceding paragraph results from an application of the following *incomplete fill path theorem*. Given the adjacency graph $G(A)$ of a coefficient matrix A, the theorem provides a static characterization of where fill entries arise during an incomplete factorization $A = \hat{L}\hat{U} + E$, where $\hat{L}$ is the lower triangular incomplete factor, $\hat{U}$ is the upper triangular incomplete factor, and E is the remainder matrix. The characterization is static in that fill is completely described by the structure of the graph $G(A)$; no information from the factor is required.

We need a definition before we can state the theorem. A *fill path* is a path joining two vertices i and j, all of whose interior vertices are numbered lower than the end vertices i and j.[1]

Recall also the definition of the levels assigned to nonzeros in an incomplete factorization. To discuss the sparsity pattern of the incomplete factors, we consider the filled matrix $F = \hat{L} + \hat{U} - I$. The sparsity pattern of F is initialized to that of A. All nonzero entries in F corresponding to nonzeros in A have level zero, and zero entries have level infinity. New entries that arise during factorization are assigned a level based on the levels of the causative entries, according to the rule

$$\text{level}(f_{ij}) = \min_{1 \le h < \min\{i,j\}} \{\text{level}(f_{ih}) + \text{level}(f_{hj}) + 1\}.$$

The incomplete fill path theorem describes an intimate relationship between fill entries in ILU(k) factors and path lengths in graphs.

THEOREM 2.1. *Let $F = \hat{L} + \hat{U} - I$ be the filled matrix corresponding to an incomplete factorization of A, and let f_{ij} be a nonzero entry in F. Then f_{ij} is a level k entry if and only if there exists a shortest fill path of length $k+1$ that joins i and j in $G(A)$.*

A proof and a discussion of this theorem are included in the appendix.

Now consider the adjacency graph $G(A)$ and a partition $\Pi = \{S_0, \ldots, S_{p-1}\}$ of it into subgraphs (subdomains). Any path joining two interior nodes in distinct subdomains must include at least two boundary nodes, one from each of the subgraphs; since each boundary node is numbered higher than (at least one of) the path's end vertices (since these are interior nodes in the subgraph), this path cannot be a fill path. If two interior nodes belonging to separate subgraphs were connected by a fill path and the corresponding fill entry were permitted in F, the interior nodes would be transformed into boundary nodes in $G(F)$. This is undesirable for parallelism, since then there would be fewer interior nodes to be eliminated concurrently.

The local ordering step preserves interior and boundary nodes during the factorization and ensures that a subdomain's interior rows can be factored independently of row updates from any other subdomain. Therefore, when subdomains have relatively large interior/boundary node ratios, and contain approximately equal amounts of computational work, we expect PILU to exhibit a high degree of parallelism.

[1]The reader has doubtless noted that *interior* is used in a different sense here than previously. We trust it will be obvious from the context where *interior* is used to refer to nodes in paths and where it is used to refer to nodes in subgraphs.

Step 3: Global ordering. The global ordering phase is intended to preserve parallelism while factoring the rows corresponding to the boundary vertices. In order to explain the loss of concurrency that could occur during this phase of the algorithm, we need the concept of a subdomain intersection graph, which we shall call a subdomain graph for brevity.

The subdomain graph $S(G, \Pi) = (V_s, E_s)$ is computed from a graph G and its partition $\Pi = \{S_0, \ldots, S_{p-1}\}$ into subgraphs. The vertex set V_s contains a vertex corresponding to every subgraph in the partition; the edge set E_s contains edge $\{S_i, S_j\}$ if there is an edge in G with one endpoint in S_i and the other in S_j. We can compute a subdomain graph $S(A)$ corresponding to the initial graph $G(A)$ and its partition. (This graph should be denoted $S(G(A), \Pi)$, but we shall write $S(A)$ for simplicity.) We could also compute a subdomain graph $S(F)$ corresponding to the graph of the factor $G(F)$. The subdomain graph $S(A)$ corresponding to the partition of the initial graph $G(A)$ (the top graph) in Figure 2.2 is shown in the graph at the bottom right in that figure.

We impose a constraint on the fill, *the subdomain graph constraint.* The subdomain graph corresponding to $G(F)$ is restricted to be identical to the subdomain graph corresponding to $G(A)$. This prohibits some fill in the filled graph $G(F)$: if two subdomains are not joined by an edge in the original graph $G(A)$, any fill edge that joins those subdomains is not permitted in the graph of the incomplete factor $G(F)$. The description of the PILU algorithm in Figure 2.1 assumes that the subdomain graph constraint is satisfied. This constraint makes it possible to obtain scalability in the parallel ILU algorithm. Later, we discuss how the algorithm should be modified if this constraint is relaxed.

Each subdomain's nodes (in $G(A)$) are ordered contiguously. Consequently, saying "subdomain r is ordered before subdomain s" is equivalent to saying "all nodes in subdomain r are ordered, and then all nodes in subdomain s are ordered." This permits $S(A)$ to be considered as a directed graph, with edges oriented from lower to higher numbered vertices.

Edges in $S(F)$ indicate data dependencies in factoring the boundary rows of the subdomains. If an edge in $S(F)$ joins r and s and subdomain r is ordered before subdomain s, then updates from the boundary rows of r have to be applied to the boundary rows of s before the factorization of the latter rows can be completed. It follows that ordering $S(F)$ so as to reduce directed path lengths reduces serial bottlenecks in factoring the boundary rows. If we impose the subdomain graph constraint, these observations apply to the subdomain graph $S(A)$ as well since then $S(A)$ is identical with $S(F)$.

We reduce directed path lengths in $S(A)$ by coloring the vertices of the subdomain graph with few colors using a heuristic algorithm for graph coloring, and then by numbering the subdomains by color classes. The boundary rows of all subdomains corresponding to the first color can be factored concurrently without updates from any other subdomains. These subdomains update the boundary rows of higher numbered subdomains adjacent to them. After the updates, the subdomains that correspond to the second color can factor their boundary rows. This process continues by color classes until all subdomains have factored their boundary rows. The number of steps it takes to factor the boundary rows is equal to the number of colors it takes to color the subdomain graph.

In Figure 2.2, let p_i denote the processor that computes the subgraph S_i. Then p_0 computes the boundary rows of S_0 and sends them to processors p_1 and p_2. Similarly,

p_3 computes the boundary rows of subgraph S_3 and sends them to p_1 and p_2. The latter processors first apply these updates and then compute their boundary rows.

How much parallelism can be gained through subdomain graph reordering? We can gain some intuition through analysis of simplified model problems, although we cannot answer this question a priori for general problems and all possible partitions. Consider a matrix arising from a second order PDE that has been discretized on a regularly structured 2D grid using a standard five-point stencil. Assume that the grid is naturally ordered and that it has been partitioned into square subgrids and mapped into a square grid of p processors. In the worst case, the associated subdomain graph, which itself has the appearance of a regular 2D grid, can have a dependency path of length $2(\sqrt{p}-1)$. Similarly, a regularly structured 3D grid discretized with a seven-point stencil that is naturally ordered and then mapped on a cube containing p processors can have a dependency path length of $3(\sqrt[3]{p}-1)$. However, regular 2D grids with the five-point stencil and regular 3D grids with the seven-point stencil are bipartite graphs and can be colored with two colors. If all subdomains of the first color class are numbered first, and then all subdomains of the second color class are numbered, the longest dependency path in S will be reduced to one. This discussion shows that coloring the subdomain graph is an important step in obtaining a scalable parallel algorithm.

Step 4: Preconditioner computation. Now that the subdomains and the nodes in each subdomain have been ordered, the preconditioner can be computed. We employ an upward-looking, row oriented factorization algorithm. The interior of each subdomain can be computed concurrently by the processors, and the boundary nodes can be computed in increasing order of the color classes. Either a level-based ILU(k) or a numerical threshold based ILUT(τ, p) algorithm may be employed on each subdomain. Different incomplete factorization algorithms could be employed in different subdomains when appropriate, as in multiphysics problems. Different fill levels could be employed for the interior nodes in a subdomain and for the boundary nodes to reduce communication and synchronization costs.

2.2. Relaxing the subdomain graph constraint. Now we consider how the subdomain graph constraint might be relaxed. Given a graph $G(A)$ and a partition of it into subgraphs, we color the subdomain graph $S(A)$ and order its subdomains as before. Then we compute the graph $G(F)$ of an incomplete factor and its subdomain graph $S(F)$. To do this, we need to discover the dependencies in $S(F)$, but initially we have only the dependencies in $S(A)$ available. This has to be done in several rounds, because fill edges could create additional dependencies between the boundary rows of subdomains, which in turn might lead to further dependences. The number of rounds needed is the length of a longest dependency path in the subdomain graph $G(F)$, and this could be $\Omega(p)$. This discussion applies when an ILU(k) algorithm is employed, with symbolic factorization preceding numerical factorization. If ILUT were to be employed, then symbolic factorization and numerical factorization must be interleaved, as would be done in a sequential algorithm.

We can then color the vertices of $S(F)$ to compute a schedule for factoring the boundary rows of the subdomains. For achieving concurrency in this step the subdomain graph $S(F)$ should have a small chromatic number (independent of the number of vertices in $G(A)$). Note that the description of the PILU algorithm in Figure 2.1 needs to be modified to reflect this discussion when the subdomain graph constraint is relaxed.

The graph $G(F)$ in Figure 2.2 indicates the fill edges that join S_1 to S_2 as broken

lines. The corresponding subdomain intersection graph $S(F)$ is shown on the lower left. The edge between S_1 and S_2 necessitates three colors to color $S(F)$: the subdomains S_0 and S_3 form one color class; S_1 by itself constitutes the second color class; and S_2 by itself makes up the third color class. Thus three steps are needed for the computation of the boundary rows of the preconditioner when the subdomain graph constraint is relaxed. Note that the processor responsible for the subdomain S_2 can begin computing its boundary rows when it receives an update from either S_0 or S_3, but that it cannot complete its computation until it has received the update from the subdomain S_1.

Theorem 2.1 has an intuitively simple geometric interpretation. Given an initial node i in $G(A)$, construct a topological "sphere" containing all nodes that are at a distance less than or equal to $k+1$ edges. Then a fill entry f_{ij} is admissible in an $ILU(k)$ factor only if j is within the sphere. Note that all such nodes j do not cause fill edges since there needs to be a fill path joining i and j. By applying Theorem 2.1, we can gain an intuitive understanding of the fill entries that may be discarded on account of the subdomain graph constraint. Referring again to Figure 2.2, we see that prohibited edges arise when two nonadjacent subdomains in $G(A)$ have nodes that are joined by a fill path of length less than $k+1$. No level zero edge is discarded by the constraint.

2.3. Existence of PILU preconditioners. The existence of preconditioners computed from the PILU algorithm can be proven for some classes of problems.

Meijerink and van der Vorst [28] proved that if A is an M-matrix, then ILU factors exist for any predetermined sparsity pattern, and Manteuffel [27] extended this result to H-matrices with positive diagonal elements. These results immediately show that PILU preconditioners with sparsity patterns based on level values exist for these classes of matrices. This is true even when different level values are used for the various subdomains and boundaries.

Incomplete Cholesky (IC) preconditioners for symmetric problems could be computed with our parallel algorithmic framework using preconditioners proposed by Jones and Plassmann [21] and by Lin and Moré [23] on each subdomain and on the boundaries. The sparsity patterns of these preconditioners are determined by the numerical values in the matrix and by memory constraints. Lin and Moré have proved that these preconditioners exist for M- and H-matrices. Parallel IC preconditioners also can be shown to exist for M- and H-matrices. If the subdomain graph constraint is not enforced, then the preconditioner computed in parallel corresponds to a preconditioner computed by the serial algorithm from a reordered matrix. If the constraint is enforced, some specified fill elements are dropped from the Schur complement; it can be shown that the resulting Schur complement matrix is componentwise larger than the former and hence still an M-matrix.

2.4. Relation to earlier work. We now briefly discuss earlier parallel ILU algorithms that are related to the PILU algorithm proposed here. Earlier attempts at parallel algorithms for preconditioning (including approaches other than incomplete factorization) are surveyed in [6, 12, 34]; orderings suitable for parallel incomplete factorizations have been studied inter alios in [4, 11, 13]. The surveys also describe the alternate approximate inverse approach to preconditioning.

Saad [33, section 12.6.1] discusses a distributed ILU(0) algorithm that has the features of graph partitioning, elimination of interior nodes in a subdomain before boundary nodes, and coloring the subdomains to process the boundary nodes in parallel. Only level 0 preconditioners are discussed there, so that fill between subdomains, or

within each subdomain, do not need to be considered. No implementations or results were reported, although Saad has informed us recently of a technical report [24] that includes an implementation and results. Our work, done independently, shows how fill levels higher than zero can be accommodated within this algorithmic framework. We also analyze our algorithm for scalability and provide computational results on the performance of PILU preconditioners. Our results show that fill levels higher than zero are indeed necessary to obtain parallel codes with scalability and good performance.

Karypis and Kumar [22] have described a parallel ILUT implementation based on graph partitioning. Their algorithm does not include a symbolic factorization, and they discover the sparsity patterns and the values of the boundary rows after the numerical computation of the interior rows in each subdomain. The factorization of the boundary rows is done iteratively, as in the discussion given above, where we show how the subdomain graph constraint might be relaxed. The partially filled graph of the boundary rows after the interior rows are eliminated is formed, and this graph is colored to compute a schedule for computing the boundary rows. Since fill edges in the boundary rows are discovered as these rows are being factored, this approach could lead to long dependency paths that are $\Theta(p)$. The number of boundary rows is $\Omega(N^{1/2})$ for 2D meshes, and $\Omega(N^{2/3})$ for 3D meshes with good aspect ratios. If the cost of factoring and communicating a boundary row is proportional to the number of rows, then this phase of their algorithm could cost $\Omega(p\sqrt{N})$, severely limiting the scalability of the algorithm (cf. the discussion in section 3).

Recently Magolu monga Made and van der Vorst [25, 26] have reported variations of a parallel algorithm for computing ILU preconditioners. They partition the mesh, linearly order the subdomains, and then permit fill in the interior and the boundaries of the subdomains. The boundary nodes are classified with respect to the number of subdomains they are adjacent to, and are eliminated in increasing order of this number. Since the subdomains are linearly ordered, a "burn from both ends" ordering is employed to eliminate the subdomains. Our approaches are similar, except that we additionally order the subdomains by means of a coloring to reduce dependency path lengths to obtain a scalable algorithm. They have provided an analysis of the condition number of the preconditioned matrices for a class of 2D second order elliptic boundary value problems. They permit high levels of fill (four or greater) as we do, and show that the increased fill permitted across the boundaries enables the condition number of the preconditioned matrix to be insensitive to the number of subdomains (except when the latter gets too great). We have worked independently of each other.

A different approach, based on partitioning the mesh into rectangular strips and then computing the preconditioner in parallel steps in which a "wavefront" of the mesh is computed at each step by the processors, was proposed by Bastian and Horton [3] and was implemented for shared memory multiprocessors recently by Vuik, van Nooyen, and Wesseling [36]. This approach has less parallelism than the one considered here.

3. Performance analysis. In this section we present simplified theoretical analyses of algorithmic behavior for matrices arising from PDEs discretized on 2D grids with five-point stencils and 3D grids with seven-point stencils. Since our arguments are structural in nature, we assume ILU(k) is the factorization method used. After a word about nomenclature, we begin with the 2D case.

The word *grid* refers to the grid (mesh) of unknowns for regular 2D and 3D grids with five- and seven-point stencils, respectively; this is identical to the adjacency graph $G(A)$ of the coefficient matrix of these problems. We use the terms *eliminating*

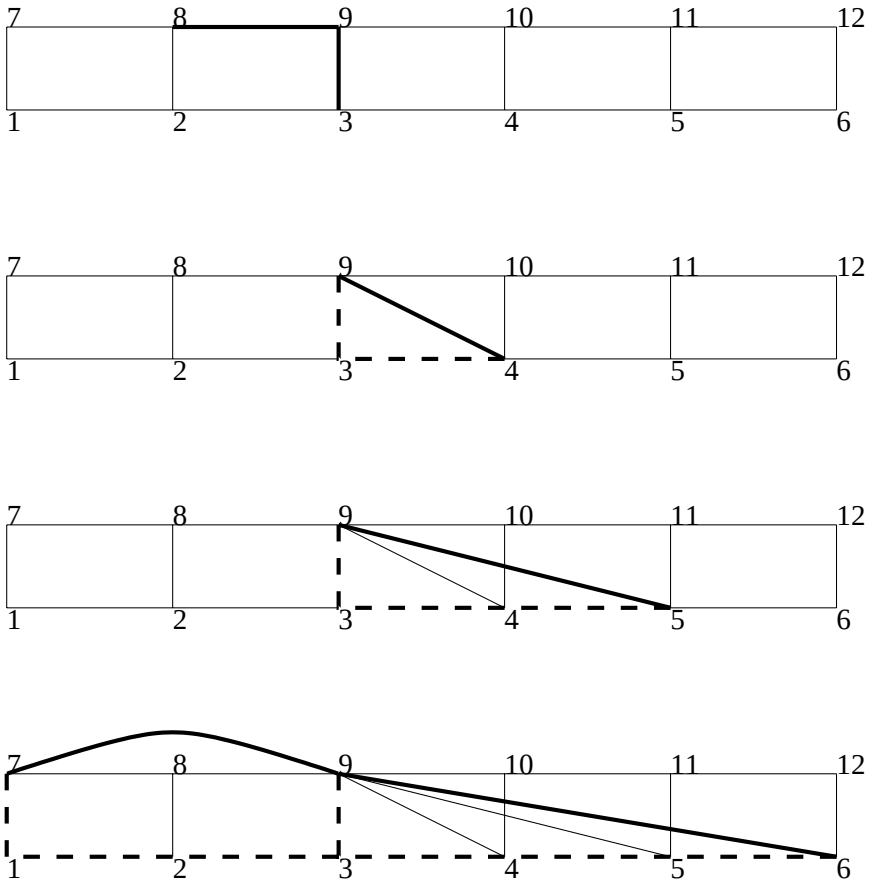


FIG. 3.1. *Counting lower triangular fill edges in a naturally ordered grid. We count the number of edges incident on vertex* 9. *Considering the graphs from top to bottom, we find that there are two level* 0 *edges; there is one level* 1 *edge, due to fill path* 9, 3, 4; *there is one level* 2 *edge due to fill path* 9, 3, 4, 5; *there are two level* 3 *edges, due to fill paths* 9, 3, 4, 5, 6 *and* 9, 3, 2, 1, 7. *We can generalize that two additional fill edges are created for every level greater than three, except near the boundaries. We conclude that asymptotically there are* $2k$ *lower triangular edges incident on a vertex in a level* k *factorization. Since the mesh corresponds to a structurally symmetric problem, there are* $2k$ *upper triangular edges incident on a vertex as well.*

a node and *factoring a row* synonymously.

We assume the grid has been block-partitioned, with each subdomain consisting of a square subgrid of dimension $c \times c$. We also assume the subdomain grid has dimensions $\sqrt{p} \times \sqrt{p}$, so there are p processors in total. There are thus $N = c^2 p$ nodes in the grid, and subdomains have at most $4c = 4\sqrt{\frac{N}{p}}$ boundary nodes.

If subdomain interior nodes are locally numbered in natural order and $k \ll c$, each row in the factor F asymptotically has $2k$ (strict) upper triangular and $2k$ (strict) lower triangular nonzero entries. The justification for this statement arises from a consideration of the incomplete fill path theorem; the intuition is illustrated in Figure 3.1.

Assuming that the classical ILU(k) algorithm is used for symbolic factorization, both symbolic and numeric factorization of row j entails $4k^2$ arithmetic operations. This is because for each lower triangular entry f_{ji} in matrix row j, factorization requires an arithmetic operation with each upper triangular entry in row i.

A red-black ordering of the subdomain graph gives an optimal bipartite division.

If red subdomains are numbered before black subdomains, our algorithm simplifies to the following three stages.

1. Red processors eliminate all nodes; black processors eliminate interior nodes.
2. Red processors send boundary-row structure and values to black processors.
3. Black processors eliminate boundary nodes.

If these stages are nonoverlapping, the cost of the first stage is bounded by the cost of eliminating all nodes in a subdomain. This cost is $4k^2c^2 = \frac{4k^2N}{p}$.

The cost for the second stage is the cost of sending structural and numerical values from the upper-triangular portions of the boundary rows to neighboring processors. If $k \ll c$, the incomplete fill path theorem can be used to show that, asymptotically, a processor only needs to forward values from c rows to each neighbor. We assume a standard, noncontentious communication model wherein α and β represent message startup and per-word-transfer times, respectively. We measure these times in nondimensional units of flops by dividing them by the time it takes to execute one flop. The time for an arithmetic operation is thus normalized to unity. Then the cost for the second step is $4(\alpha + 2k\beta c) = 4(\alpha + 2k\beta\sqrt{\frac{N}{p}})$.

Since the cost of factoring a boundary row can be shown to be asymptotically identical to that for factoring an interior row, the cost for eliminating the $4c$ boundary nodes is $(4k^2)(4c) = 16k^2\sqrt{\frac{N}{p}}$. Speedup can then be expressed as

$$\text{speedup} = \frac{4k^2N}{\frac{4k^2N}{p} + 4(\alpha + 2k\beta\sqrt{\frac{N}{p}}) + 16k^2\sqrt{\frac{N}{p}}}.$$

The numerator represents the cost for sequential execution, and the three terms in the denominator represent the costs for the three stages (arithmetic for interior nodes, communication costs, and arithmetic for the boundary nodes) of the parallel algorithm.

Three implications from this equation are in order. First, for a fixed problem size and number of processors, the parallel computational cost (the first and third terms in the denominator) is proportional to k^2, while the communication cost (the second term in the denominator) is proportional to k. This explains the increase in efficiency with level that we have observed. Second, if the ratio N/p is large enough, the first term in the denominator will become preeminent, and efficiency will approach 100%. Third, if we wish to increase the number of processors p by some factor while maintaining a constant efficiency, we need only increase the size of the problem N by the same factor. This shows that our algorithm is scalable. This observation is not true for a direct factorization of the coefficient matrix, where the dependencies created by the additional fill cause loss in concurrency.

For the 3D case we assume partitioning into cubic subgrids of dimension $c \times c \times c$ and a subdomain grid of dimension $p^{1/3} \times p^{1/3} \times p^{1/3}$, which gives $N = c^3p$. Subdomains have at most $6c^2$ boundary nodes. A development similar to that above shows that, asymptotically, matrix rows in the factor F have $2k^2$ (strict) upper and lower triangular entries, so the cost for factoring a row is $4k^4$. Speedup for this case can then be expressed as

$$\begin{aligned}\text{speedup} &= \frac{4k^4N}{\frac{4k^4N}{p} + 6(\alpha + 2k^2\beta(\frac{N}{p})^{1/3}) + 24k^4(\frac{N}{p})^{1/3}} \\ &= \frac{2k^4N}{\frac{2k^4N}{p} + 3(\alpha + 2k^2\beta(\frac{N}{p})^{1/3}) + 12k^4(\frac{N}{p})^{1/3}}.\end{aligned}$$

4. Results. Results in this section are based on the following model problems.
Problem 1. Poisson's equation in two or three dimensions:

$$\Delta u = g.$$

Problem 2. Convection-diffusion equation with convection in the xy plane:

$$-\varepsilon \triangle u + \frac{\partial}{\partial x} e^{xy} u + \frac{\partial}{\partial y} e^{-xy} u = g.$$

Homogeneous boundary conditions were used for both problems. Derivative terms were discretized on the unit square or cube, using 3-point central differencing on regularly spaced $n_x \times n_y \times n_z$ grids ($n_z = 1$ for 2D). The values for ε in Problem 2 were set to 1/500 and 1/1000. The problem becomes increasingly unsymmetric, and more difficult to solve accurately as ε decreases. The right-hand sides of the resulting systems, $Ax = b$, were artificially generated as $b = A\hat{e}$, where $\hat{e}$ is the all-ones vector.

ILU(k) preconditioning is amenable to performance analysis since the nonzero structures of ILU(k) preconditioners are identical for *any* PDE that has been discretized on a 2D or 3D grid with a given stencil. The structure depends on the grid and the stencil only and is not affected by numerical values if pivoting is not needed for numerical stability. Identical structures imply identical symbolic factorization costs, as well as identical flop counts during the numerical factorization and solve phases. In parallel contexts, communication patterns and costs are also identical. While preconditioner effectiveness—the number of iterations until the stopping criteria is reached—differs with the numerics of the particular problem being modeled, the parallelism available in the preconditioner does not.

The structure of ILUT preconditioners, on the other hand, is a function of the grid, the stencil, and the numerics. Changing the problem, particularly for non-diagonally dominant cases, can alter the preconditioner structure, even when the grid and stencil remain the same.

We report our performance evaluation for ILU(k) preconditioners, although the parallel algorithmic framework proposed here could just as easily work with ILUT(τ, p). We have compared the performance of ILU(k) with ILUT in an earlier report [18]. We report there that for Problem 2 with $\varepsilon = 1/500$, ILUT(0.001, 10) incurred more fill than ILU(5) on a 2D domain for grid sizes up to 400×400; for 3D domains and grid sizes up to $64 \times 64 \times 64$, the same ILUT preconditioner incurred fill between ILU(2) and ILU(3).

In addition to demonstrating that our algorithm can provide high degrees of parallelism, we address several other issues. We study the influence of the subdomain graph constraint on the fill permitted in the preconditioner and on the convergence of preconditioned Krylov space solvers. We also report convergence results as a function of the number of nonzeros in the preconditioner.

4.1. Parallel performance. We now report timing and scalability results for preconditioner factorization and application on three parallel platforms:

- an SGI Origin2000 at NASA Ames Research Center (AMES);
- the Coral PC Beowulf cluster at ICASE, NASA Langley Research Center;
- a Sun HPC 10000 Starfire server at Old Dominion University (ODU).

TABLE 4.1

Time (sec.) required for incomplete (symbolic and numeric) factorization for a 3D *scaled problem;* 91,125 *unknowns per processor, seven-point stencil, ILU*(2) *factorization on interior nodes, and ILU*(1) *factorization on boundary nodes. Dashes (-) for Beowulf and HPC* 10000 *indicate that the machines have insufficient cpus to perform the runs.*

Procs	Origin2000 AMES	Beowulf (ICASE)	HPC 10000 (ODU)
1	2.04	2.27	2.13
8	2.44	3.11	2.43
27	2.96	4.06	2.97
64	3.11	4.64	-
125	3.18	-	-
216	3.32	-	-

Both problems were solved using Krylov subspace methods as implemented in the PETSc [2] software library. Problem 1 was solved using the conjugate gradient method, and Problem 2 was solved using Bi-CGSTAB [35]. PETSc's default convergence criterion was used, which is five orders of magnitude (10^5) reduction in the residual of the preconditioned system. We used our own codes for problem generation, partitioning, ordering, and symbolic factorization.

Table 4.1 shows incomplete factorization timings for a 3D memory-scaled problem with approximately 91,125 unknowns per processor. As the number of processors increases, so does the size of the problem. The coefficient matrix of the problem factored on 216 processors has about 19.7 million rows. ILU(2) was employed for the interior nodes, and ILU(1) was employed for the boundary nodes. Reading down any of the columns shows that performance is highly scalable, e.g., for the SGI Origin2000, factorization for 216 processors and 19.7 million unknowns required only 62% longer than the serial case. Scanning horizontally indicates that performance was similar across all platforms, e.g., execution time differed by less than a factor of two between the fastest (Origin2000) and slowest (Beowulf) platforms.

Table 4.2 shows similar data and trends for the triangular solves for the scaled problem. Scalability for the solves was not quite as good as for factorization; e.g., the solve with 216 processors took about 2.5 times longer than the serial case. This is expected due to the lower computation cost relative to communication and synchronization costs in triangular solution.

We observed that the timings for identical repeated runs on the HPC 10000 and SGI typically varied by 50% or more, while repeated runs on the Beowulf were remarkably consistent.

Table 4.3 shows speedup for a constant-sized problem of 1.7 million unknowns. There is a clear correlation between performance and subdomain interior/boundary node ratios; this ratio needs to be reasonably large for good performance.

The performances reported in these tables are applicable to any PDE that has been discretized with a seven-point central difference stencil since the sparsity pattern of the symbolic factor depends on the grid and the stencil only.

4.2. Convergence studies. Our approach for designing parallel ILU algorithms reorders the coefficient matrices whose incomplete factorization is being computed. This reordering could have a significant influence on the effectiveness of the ILU preconditioners. Accordingly, in this section we report the number of iterations of a preconditioned Krylov space solver needed to reduce the residual by a factor of 10^5.

We compare three different algorithms.

TABLE 4.2
Time (sec.) to compute triangular solves for 3D *scaled problem;* $91,125$ *unknowns per processor, seven-point stencil,* $ILU(2)$ *factorization on interior nodes,* $ILU(1)$ *factorization on boundary nodes. Dashes (-) for Beowulf and HPC* 10000 *indicate that the machines have insufficient cpus to perform the runs.*

Procs	Origin2000 (AMES)	Beowulf (ICASE)	HPC 10000 (ODU)
1	.182	.187	.289
8	.431	.359	.515
27	.405	.508	.629
64	.472	.556	-
125	.610	-	-
216	.646	-	-

TABLE 4.3
Speedup for 3D *constant-size problem; the grid was* $120 \times 120 \times 120$ *for a total of approximately* 1.7 *million unknowns; data is for* $ILU(0)$ *factorization performed on the SGI Origin*2000*; "I/B ratio" is the ratio of interior to boundary nodes in each subdomain.*

Procs	Unknowns/ Processor	I/B ratio	Time (sec.)	Efficiency (%)
8	216,000	9.3	2.000	100
27	64,000	6.0	0.846	70
64	27,000	4.3	.408	62
125	13,824	3.4	.307	42

- Constrained PILU(k) is the parallel ILU(k) algorithm with the subdomain graph constraint enforced.
- In unconstrained PILU(k), the subdomain graph constraint is dropped, and all fill edges up to level k between the boundary nodes of different subdomains are permitted, even when such edges join two nonadjacent subdomains of the initial subdomain graph $S(A)$.
- In block Jacobi ILU(k) (BJILU(k)), all fill edges joining two different subdomains are excluded.

Intuitively, one expects, especially for diagonally dominant matrices, that larger amounts of fill in preconditioners will reduce the number of iterations required for convergence.

4.2.1. Fill count comparisons. For a given problem, the number of permitted fill edges is a function of three components: the factorization level, k; the subdomain size(s); and the discretization stencil. While the numerical values of the coefficients of a particular PDE influence convergence, they do not affect fill counts. Therefore, our first set of results consists of fill count comparisons for problems discretized on a $64 \times 64 \times 64$ grid using a standard, seven-point stencil.

Table 4.4 shows fill count comparisons between unconstrained PILU(k), constrained PILU(k), and block Jacobi ILU(k) for various partitionings and factorization levels. The data shows that more fill is discarded as the factorization level increases, and as subdomain size (the number of nodes in each subdomain) decreases. These two effects hold for both constrained PILU(k) and block Jacobi ILU(k) but are much more pronounced for the latter. For example, less than 5% of fill is discarded from unconstrained PILU(k) factors when subdomains contain at least 512 nodes (so that the

TABLE 4.4

Fill comparisons for the $64 \times 64 \times 64$ grid. U denotes unconstrained, C denotes constrained, and B denotes block Jacobi ILU(k) preconditioners. The columns headed "nzF/nzA" show the ratio of the number of nonzeros in the preconditioner to the number of nonzeros in the original problem and are indicative of storage requirements. The columns headed "constraint effects" present another view of the same data: here, the percentage of nonzeros in the constrained PILU(k) and block Jacobi ILU(k) factors are shown relative to that for the unconstrained PILU(k). These columns show the amount of fill dropped due to the subdomain graph constraint.

Nodes per subdom.	Subdom. count	Level	**nzF/nzA** U	C	B	**Constraint effects (%)** C	B
262,144	1	0	1.00	1.00	1.00	100.00	100.00
		1	1.84	1.84	1.84	100.00	100.00
		2	3.22	3.22	3.22	100.00	100.00
		3	5.96	5.96	5.96	100.00	100.00
		4	9.73	9.73	9.73	100.00	100.00
32,768	8	0	1.00	1.00	0.99	100.00	98.64
		1	1.87	1.87	1.80	99.99	96.53
		2	3.36	3.35	3.12	99.96	92.91
		3	6.32	6.32	5.70	99.92	90.13
		4	10.50	10.49	9.19	99.89	87.56
4,096	64	0	1.00	1.00	0.96	100.00	95.93
		1	1.89	1.89	1.72	99.90	91.24
		2	3.45	3.44	2.91	99.62	84.36
		3	6.51	6.47	5.19	99.34	79.72
		4	10.81	10.70	8.17	99.06	75.61
512	512	0	1.00	1.00	0.90	100.00	90.50
		1	1.92	1.91	1.57	99.46	81.62
		2	3.59	3.52	2.53	98.05	70.35
		3	6.72	6.50	4.27	96.62	63.47
		4	10.96	10.43	6.32	95.20	57.69
64	4,096	0	1.00	1.00	0.80	100.00	79.64
		1	1.97	1.92	1.29	97.58	65.15
		2	3.73	3.42	1.86	91.67	49.79
		3	6.60	5.64	2.71	85.37	41.04
		4	10.01	7.76	3.35	77.56	33.45
8	32,768	0	1.00	1.00	0.58	100.00	57.92
		1	2.05	1.85	0.80	90.07	38.81
		2	3.98	2.55	0.87	64.14	21.84
		3	6.15	2.89	0.90	46.95	14.72
		4	7.40	2.90	0.90	39.26	12.23

subgraphs on each processor are not too small), but up to 42% is discarded from block Jacobi factors. Thus, one might tentatively speculate that, for a given subdomain size and level, PILU(k) will provide more effective preconditioning than BJILU(k). We have observed similar behavior for 2D problems also. For both 2D and 3D problems, when there is a single subdomain the factors returned by the three algorithms are identical. For the single subdomain case, the ordering we have used corresponds to the natural ordering for these model problems.

An important observation to make in Table 4.4 is how the sizes (number of nonzeros) of the preconditioners depend on levels of fill. For the 3D problems considered here (cube with 64 points on each side, seven-point stencil), a level one preconditioner typically requires twice as much storage as the coefficient matrix A; when the level is two, this ratio is about three; when the level is three, it is about six; and when the level is four, it is about ten. For 2D problems (square grid with 256 points on a side,

TABLE 4.5

Iteration comparisons for the $64\times64\times64$ *grid. U denotes unconstrained, C denotes constrained, and B denotes block Jacobi ILU(k) preconditioners. The starred entries (*) indicate that, since there is a single subdomain, the factor is structurally and numerically identical to the unconstrained PILU(k). Dashed entries (-) indicate the solutions either diverged or failed to converge after* 200 *iterations. For Problem* 2*, when* $\varepsilon = 1/500$ *the level zero preconditioners did not reduce the relative error in the solution by a factor of* 10^5 *at termination; when* $\varepsilon = 1/1000$*, the level one preconditioners did not do so either.*

Nodes per subdom.	Subdom. count	Level	Problem 1			Problem 2					
						$\epsilon = 1/500$			$\epsilon = 1/1000$		
			U	C	B	U	C	B	U	C	B
262,144	1	0	43	*	*	19	*	*	-	*	*
		1	29	*	*	16	*	*	30	*	*
		2	24	*	*	8	*	*	32	*	*
		3	19	*	*	8	*	*	14	*	*
		4	16	*	*	6	*	*	8	*	*
32,768	8	0	45	45	53	32	32	26	-	-	-
		1	32	33	41	14	14	19	38	39	41
		2	27	29	37	11	11	17	38	38	66
		3	22	24	33	8	8	13	16	15	21
		4	19	21	29	7	7	13	10	11	18
4,096	64	0	43	43	55	33	33	49	-	-	-
		1	31	32	45	15	15	21	42	41	46
		2	25	27	41	12	11	22	24	28	78
		3	20	23	39	9	9	16	18	17	28
		4	17	20	36	8	8	19	11	12	27
512	512	0	41	41	56	28	28	67	-	-	-
		1	29	31	48	18	16	29	39	40	111
		2	25	26	46	11	12	36	21	21	106
		3	21	23	44	11	11	31	20	21	110
		4	18	21	43	9	12	34	13	14	70
64	4,096	0	43	43	64	28	28	-	63	63	-
		1	30	33	60	17	18	124	55	56	-
		2	26	30	58	13	15	115	25	28	-
		3	21	28	58	12	17	127	24	36	-
		4	17	28	58	10	17	132	11	27	-
8	32,768	0	46	46	83	43	43	-	83	83	-
		1	32	41	82	24	46	-	152	-	-
		2	25	40	82	11	45	-	13	115	-
		3	19	40	82	5	44	-	7	107	-
		4	16	40	82	4	45	-	6	111	-

five-point stencil), the growth of fill with level is slower; the ratios are about 1.4 for level one, 1.8 for level two, 2.6 for level three, 3.5 for level four, 4.3 for level five, and 5.4 for level six.

In parallel computation fill levels higher than those employed in sequential computing are feasible since modern multiprocessors are either clusters or have virtual shared memory, and these have memory sizes that increase with the number of processors. Another point to note is that the added memory requirement for these level values is not as prohibitive as it is for a complete factorization. Hence it is practical to trade-off increased storage in preconditioners for reducing the number of iterations in the solver.

4.2.2. Convergence of preconditioned iterative solvers. The fill results in the previous subsection are not influenced by the actual numerical values of the

nonzero coefficients; however, the convergence of preconditioned Krylov space solvers is influenced by the numerical values. Accordingly, Table 4.5 shows iterations required for convergence for various partitionings and fill levels for the three variant algorithms that we consider. The data in these tables can be interpreted in various ways; we begin by discussing two ways that we think are primarily significant.

First, by scanning vertically one can see how changing the number of subdomains, and hence, matrix ordering, affects convergence. The basis for comparison is the iteration count when there is a single subdomain. The partitioning and ordering for these cases is identical to, and our data in close agreement with, that reported by Benzi, Joubert, and Mateescu [4] for natural ordering. (They report results for Problem 2 with $\varepsilon = 1/500$ but not for $\varepsilon = 1/1000$.)

A pleasing property of both the constrained and unconstrained PILU algorithms is that the number of iterations increases only mildly when we increase the number of subdomains from one to 512 for these problems. This insensitivity to the number of subdomains when the number of nodes per subdomain is not too small confirms that the PILU algorithms enjoy the property of parallel algorithmic scalability. For example, Poisson's equation (Problem 1) preconditioned with a level two factorization and a single subdomain required 24 iterations. Preconditioning with the same level, constrained PILU(k) on 512 subdomains needed only two more iterations. Similar results are observed for the convection-diffusion problems also. This property is a consequence of the fill between the subdomains that is included in the PILU algorithm. Similar results have been reported in [26, 36], and the first paper includes a condition number analysis supporting this observation.

Increasing the level of fill generally has the beneficial effect of reducing the number of iterations needed; this influence is largest for the worse-conditioned convection-diffusion problem with $\varepsilon = 1/1000$. For this problem, level zero preconditioners do not converge for reasonable subdomain sizes. Also, even though level one preconditioners require fewer iteration numbers than level two preconditioners in some cases, when the PETSc solvers terminate because the residual norms are reduced by 10^5, the relative errors are larger than 10^{-5} for the former preconditioners. The relative errors are also large for the convection-diffusion problem with $\varepsilon = 1/500$ when the level is set to zero.

Second, scanning the data in Table 4.5 horizontally permits evaluation of the subdomain graph constraint's effects. Again, unless subdomains are small and the factorization level is high; constrained and unconstrained PILU(k) show very similar behavior. Consider, for example, Poisson's equation (Problem 1) preconditioned with a level two factorization and 512 subdomains. The solution with unconstrained PILU(k) required 25 iterations while constrained PILU(k) required 26.

We also see that PILU(k) preconditioning is more effective than BJILU(k) for all 3D trials. (Recall that the single apparent exception, Problem 2, $\varepsilon = 1/500$, ILU(0) with $32,768$ nodes per subdomain, has large relative errors at termination.) Again, the extremes of convergence behavior are seen for Problem 2 with $\varepsilon = 1/1000$. Here, with level one preconditioners, BJILU(k) suffers large relative errors at termination while the other two algorithms do not, when the number of subdomains is 64 or fewer.

On 2D domains, while PILU(k) is more effective than BJILU(k) for Poisson's equation, BJILU(k) is sometimes more effective in the convection-diffusion problems.

We also examine iteration counts as a function of preconditioner size graphically. A plot of this data appears in Figure 4.1. In these figures the performance of the

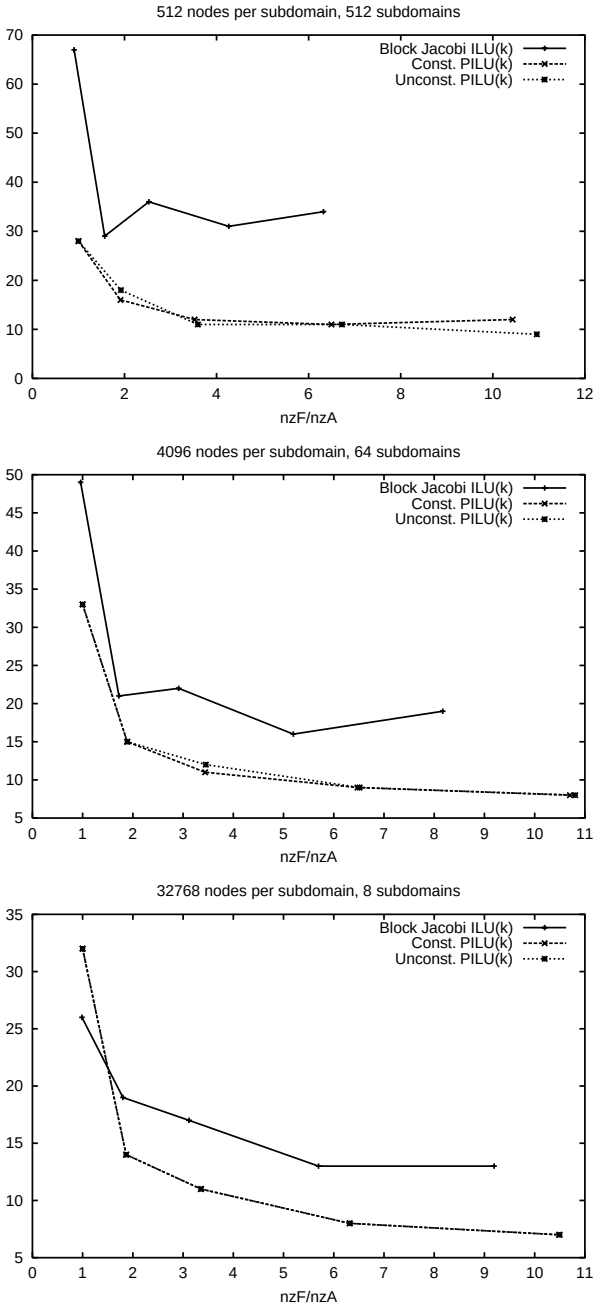


FIG. 4.1. *Convergence comparison as a function of preconditioner size for the convection-diffusion problem,* $\epsilon = 1/500$ *on the* $64 \times 64 \times 64$ *grid. Data points are for levels* 0 *through* 4. *Data points for constrained and unconstrained PILU(k) are indistinguishable in the third graph.*

constrained and unconstrained PILU algorithms is often indistinguishable. We find again that PILU(k) preconditioning is more effective than BJILU(k) for 3D problems for a given preconditioner size; however, this conclusion does not always hold for 2D

problems, especially for lower fill levels. As the number of vertices in the subdomains increases, higher fill levels become more effective in reducing the number of iterations needed for convergence. We find that fill levels as high as four to six can be the most effective when the subdomains are sufficiently large. Fill levels higher than these do not seem to be merited by these problems, even for the difficult convection-diffusion problems with $\varepsilon = 1/1000$, where a level four preconditioner reduces the number of iterations below ten.

5. Conclusions. We have designed and implemented a PILU algorithm, a scalable parallel algorithm for computing ILU preconditioners that creates concurrency by means of graph partitioning. The theoretical basis of the algorithm is the incomplete fill path theorem that statically characterizes fill elements in an incomplete factorization in terms of paths in the adjacency graph of the initial coefficient matrix. To obtain a scalable parallel algorithm, we employ a subdomain graph constraint that excludes fill between subgraphs that are not adjacent in the adjacency graph of the initial matrix. We show that the PILU algorithm is scalable by an analysis for 2D- and 3D-model problems and by computational results from parallel implementations on three parallel computing platforms.

We also study the convergence behavior of preconditioned Krylov solvers with preconditioners computed by the PILU algorithm. The results show that fill levels higher than one are effective in reducing the number of iterations, that the number of iterations is insensitive to the number of subdomains, and that the subdomain graph constraint does not affect the number of iterations needed for convergence while it makes possible the design of a scalable parallel algorithm.

Appendix. Proof of the incomplete fill path theorem.

THEOREM A.1. *Let $F = \hat{L} + \hat{U} - I$ be the filled matrix corresponding to an incomplete factorization of A, and let f_{ij} be a nonzero entry in F. Then f_{ij} is a level k entry if and only if there exists a shortest fill path of length $k+1$ that joins i and j in $G(A)$.*

Proof. If there is a shortest fill path of length $k+1$ joining i and j, we prove that the edge exists by induction on the length of the fill path.

Define a *chord* of a path to be an edge that joins two nonconsecutive vertices on the path. The fill path joining i and j is chordless, since a chord would lead to a shorter fill path.

The base case $k = 0$ is immediate, since a fill path of length one in the graph $G(A)$ is an edge $\{i, j\}$ in $G(A)$ that corresponds to an original nonzero in A.

Now assume that the result is true for all lengths less than $k+1$. Let h denote the highest numbered interior vertex on the fill path joining i and j.

We claim that the (i, h) section of this path is a shortest fill path in $G(A)$ joining i and h. This section is a fill path by the choice of h since all intermediate vertices on this section are numbered lower than h. If there were a fill path joining i and h that is shorter than the (i, h) section, then we would be able to concatenate it with the (h, j) section to form a shorter (i, j) fill path. Hence the (i, h) section is a shortest fill path joining i and h. Similarly, the (h, j) section of this path is the shortest fill path joining h and j.

Each of these sections has fewer than $k+1$ edges, and hence the inductive hypothesis applies. Denote the number of edges in the (i, h) ((h, j)) section of this path by k_1 (k_2), where $k_1 + k_2 = k + 1$. By the inductive hypothesis, the edge $\{i, h\}$ is a fill edge of level $k_1 - 1$, and the edge $\{h, j\}$ is a fill edge of level $k_2 - 1$. Now by the sum rule for updating fill levels, when the vertex h is eliminated, we have a fill edge

$\{i, j\}$ of level

$$(k_1 - 1) + (k_2 - 1) + 1 = (k_1 + k_2) - 1 = (k + 1) - 1 = k.$$

Now we prove the converse. Suppose that $\{i, j\}$ is a fill edge of level k; we show that there is a fill path in $G(A)$ of length $k + 1$ edges by induction on the level k.

The base case $k = 0$ is immediate, since the edge $\{i, j\}$ constitutes a trivial fill path of length one. Assume that the result is true for all fill levels less than k. Let h be a vertex whose elimination creates the fill edge $\{i, j\}$ of level k. Let the edge $\{i, h\}$ have level k_1, and let the edge $\{h, j\}$ have level k_2; by the sum rule for computing levels, we have that $k_1 + k_2 + 1 = k$. By the inductive hypothesis, there is a shortest fill path of length $k_1 + 1$ joining i and h, and such a path of length $k_2 + 1$ joining h and j. Concatenating these paths, we find a fill path joining i and j of length

$$(k_1 + 1) + (k_2 + 1) = k_1 + k_2 + 2 = k + 1.$$

We need to prove that the (i, j) fill path in the previous paragraph is a shortest fill path between i and j. Consider the elimination of any another vertex g that causes the fill edge $\{i, j\}$. By the choice of the vertex h, if the level of the edge $\{i, g\}$ is k'_1 and that of $\{g, j\}$ is k'_2, then $k'_1 + k'_2 + 1 \geq k$. The inductive hypothesis applies to the (i, g) and (g, j) sections, and hence the sum of their lengths is at least $k + 1$.

This completes the proof. □

This result is a generalization of the following theorem that characterizes fill in complete factorizations for direct methods, due to Rose and Tarjan [30].

THEOREM A.2. *Let $F = L + U - I$ be the filled matrix corresponding to the complete factorization of A. Then $f_{ij} \neq 0$ if and only if there exists a fill path joining i and j in the graph $G(A)$.*

Here we associate level values with each fill edge and relate it to the length of shortest fill paths. The incomplete fill path theorem enables new algorithms for incomplete symbolic factorization that are more efficient than the conventional algorithm that simulates numerical factorization. We have described these algorithms in an earlier work [29] and the report is in preparation.

D'Azevedo, Forsyth, and Tang [9] have defined the (sum) level of a fill edge $\{i, j\}$ using the length criterion employed here, and hence they were aware of this result. However, the theorem is neither stated nor proved in their paper. Definitions of level that compute levels of fill nonzeros by rules other than by summing the levels of the causative pairs of nonzeros have been used in the literature. The "maximum" rule defines the level of a fill nonzero to be the minimum over all causative pairs of the maximum value of the levels of the causative entries:

$$\text{level}(f_{ij}) = \min_{1 \leq h \leq \min\{i,j\}} \max\{\text{level}(f_{ih}), \text{level}(f_{hj})\} + 1.$$

A variant of the incomplete fill path theorem can be proved for this case, but it is not as simple or elegant as the one for the "sum" rule. Further discussion of these issues will be deferred to a future report.

Acknowledgments. We thank Dr. Edmond Chow of CASC, Lawrence Livermore National Laboratory, and Professor Michele Benzi of Emory University for helpful discussions.

REFERENCES

[1] O. Axelsson, *Iterative Solution Methods*, Cambridge University Press, Cambridge, UK, 1994.

[2] S. Balay, W. D. Gropp, L. Curfman McInnes, and B. F. Smith, *PETSc home page*, http://www.mcs.anl.gov/petsc, 1999.

[3] P. Bastian and G. Horton, *Parallelization of robust multigrid methods: ILU factorization and frequency decomposition method*, SIAM J. Sci. Statist. Comput., 12 (1991), pp. 1457–1470.

[4] M. Benzi, W. Joubert, and G. Mateescu, *Numerical experiments with parallel orderings for ILU preconditioners*, Electron. Trans. Numer. Anal., 8 (1999), pp. 88–114.

[5] A. M. Bruaset and H. P. Langtangen, *Object-oriented design of preconditioned iterative methods in Diffpack*, ACM Trans. Math. Software, 23 (1997), pp. 50–80.

[6] T. F. Chan and H. A. van der Vorst, *Approximate and incomplete factorizations*, in Parallel Numerical Algorithms, ICASE/LaRC Interdiscip. Ser. Sci. Engrg. 4, D. E. Keyes, A. Sameh, and V. Venkatakrishnan, eds., Kluwer Academic, Dordecht, The Netherlands, 1997, pp. 167–202.

[7] E. Chow and M. A. Heroux, *An object-oriented framework for block preconditioning*, ACM Trans. Math. Software, 24 (1998), pp. 159–183.

[8] E. Chow and Y. Saad, *Experimental study of ILU preconditioners of indefinite matrices*, J. Comput. Appl. Math, 86 (1997), pp. 387–414.

[9] E. F. D'Azevedo, P. A. Forsyth, and W.-P. Tang, *Ordering methods for preconditioned conjugate gradient methods applied to unstructured grid problems*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 944–961.

[10] S. Doi and A. Lichnewsky, *A Graph-Theory Approach for Analyzing the Effects of Ordering on ILU Preconditioning*, Tech. report 1452, Institut National de Recherche in Informatique et en Automatique, Rocqhencourt, BP105-78153, Le Chesnay Cedex, France, 1991.

[11] S. Doi and T. Washio, *Ordering strategies and related techniques to overcome the trade-off between parallelism and convergence in incomplete factorizations*, Parallel Comput., 25 (1995), pp. 1995–2014.

[12] J. J. Dongarra, I. S. Duff, D. C. Sorensen, and H. A. van der Vorst, *Numerical Linear Algebra for High Performance Computers*, SIAM, Philadelphia, 1998.

[13] I. S. Duff and G. A. Meurant, *The effect of ordering on preconditioned conjugate gradients*, BIT, 29 (1983), pp. 635–657.

[14] V. Eijkhout, *Analysis of parallel incomplete point factorizations*, Linear Algebra Appl., 154–156 (1991), pp. 723–740.

[15] A. Greenbaum, *Iterative Methods for Solving Linear Systems*, SIAM, Philadelphia, 1997.

[16] G. Haase, *Parallel incomplete Cholesky preconditioners based on the nonoverlapping data distribution*, Parallel Comput., 24 (1998), pp. 1685–1703.

[17] W. Hackbusch and G. Wittum, eds., *Incomplete Decomposition (ILU): Algorithms, Theory, and Applications*, Notes Numer. Fluid Mech. 41, Vieweg, Braunschweig, Wiesbaden, 1993.

[18] D. Hysom and A. Pothen, *Efficient parallel computation of ILU(k) preconditioners*, in Proceedings of Supercomputing 99, ACM, New York, 1999, published on CDROM.

[19] D. Hysom and A. Pothen, *Efficient Parallel Computation of ILU(k) Preconditioners*, Tech. report 2000-23, ICASE, NASA Langley Research Center, Hampton, VA, 2000.

[20] D. Hysom and A. Pothen, *Parallel ILU Ordering and Convergence Relationships: Numerical Experiments*, Tech. report 2000-24, ICASE, NASA Langley Research Center, Hampton, VA, 2000.

[21] M. T. Jones and P. E. Plassmann, *An improved incomplete Cholesky factorization*, ACM Trans. Math. Software, 21 (1995), pp. 5–17.

[22] G. Karypis and V. Kumar, *Parallel threshold-based ILU factorization*, in Proceedings of the ACM Conference on Supercomputing, San Jose, CA, 1997, CD-ROM, ACM, New York, 1997.

[23] C.-J. Lin and J. J. Moré, *Incomplete Cholesky factorizations with limited memory*, SIAM J. Sci. Comput., 21 (1999), pp. 24–45.

[24] S. Ma and Y. Saad, *Distributed ILU*(0) *and SOR Preconditioners for Unstructured Sparse Linear Systems*, Tech. report 94-27, Army High Performance Computing Research Center, University of Minnesota, Minneapolis, MN, 1994.

[25] M. Magolu monga Made and H. A. van der Vorst, *Parallel incomplete factorizations with pseudo-overlapped subdomains*, Parallel Comput., to appear.

[26] M. Magolu monga Made and H. A. van der Vorst, *Spectral analysis of parallel incomplete factorizations with implicit pseudo-overlap*, Numer. Linear Algebra Appl., to appear.

[27] T. A. Manteuffel, *An incomplete factorization technique for positive definite linear systems*, Math. Comput., 34 (1980), pp. 307–327.
[28] J. A. Meijerink and H. A. van der Vorst, *An iterative solution method for linear equation systems of which the coefficient matrix is a symmetric M-matrix*, Math. Comput., 31 (1977), pp. 148–162.
[29] A. Pothen and D. Hysom, *Fast algorithms for incomplete factorization*, in Proceedings of the Copper Mountain Conference on Iterative Methods, Copper Mountain, CO, 1998, report in preparation.
[30] D. J. Rose and R. E. Tarjan, *Algorithmic aspects of vertex elimination on directed graphs*, SIAM J. Appl. Math., 34 (1978), pp. 176–197.
[31] Y. Saad, *Sparskit, version* 2, http://www-users.cs.umn.edu/ saad/software.html, 1990, 1994.
[32] Y. Saad, *ILUT: A dual-threshold incomplete LU factorization*, Numer. Linear Algebra Appl., 1 (1994), pp. 387–402.
[33] Y. Saad, *Iterative Methods for Sparse Linear Systems*, PWS Publishing, Boston, 1996.
[34] H. A. van der Vorst, *High performance preconditioning*, SIAM J. Sci. Statist. Comput., 10 (1989), pp. 1174–1185.
[35] H. A. van der Vorst, *Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 13 (1992), pp. 631–634.
[36] C. Vuik, R. R. P. van Nooyen, and P. Wesseling, *Parallelism in ILU-preconditioned GMRES*, Parallel Comput., (1998), pp. 1927–1946.

SIAM J. SCI. COMPUT.
Vol. 22, No. 6, pp. 2216–2238
© 2001 Society for Industrial and Applied Mathematics

ACCURATE COMPUTATIONS ON INERTIAL MANIFOLDS*

M. S. JOLLY†, R. ROSA‡, AND R. TEMAM§

Abstract. An algorithm is implemented for the computation of inertial manifolds to arbitrary accuracy. It is applied to the Kuramoto–Sivashinsky equation to recover the high wave number components of elements on the global attractor. The computational results demonstrate that this algorithm is significantly more accurate than previous methods, when compared after just a few iterations. An essential feature of the algorithm is that it does not require that the entire manifold be constructed. Hence the algorithm is particularly suited for computing trajectories on the manifold, and its computational complexity is independent of the dimension.

Key words. computation of inertial manifolds, approximate inertial manifolds, Kuramoto–Sivashinsky equation

AMS subject classifications. Primary, 34G20; Secondary, 34D45, 35Q99, 47H20, 58F39

PII. S1064827599351738

1. Introduction.

This paper demonstrates how to accurately compute inertial manifolds for evolutionary equations. These manifolds are finite-dimensional, exponentially attracting, positively invariant, and smooth. In terms of a Fourier series representation of the solution to a PDE, essentially the manifold provides a relation for the modes with high wave numbers in terms of those with low wave numbers. The existence of such a manifold typically hinges on whether there is a large enough gap in the spectrum of the linear part of the equation compared to the variation of the nonlinear part. A weaker gap condition appears in most constructive proofs of local (un)stable and center manifolds. Moreover, for local manifolds one may take a small enough neighborhood to reduce the strength of the nonlinear term and thereby achieve the gap condition. The accurate computation of such objects, however, even when the manifolds are two-dimensional, poses some special difficulties (see [24], [2], [31], and for related issues regarding invariant tori, see [12]).

We implement here an algorithm developed in [47] for the computation of inertial manifolds. It provides a sequence of approximate inertial manifolds (AIMs) which converges to the actual inertial manifold. This sequence is distinguished from all other sequences of AIMs, except that in [10], in that convergence may be achieved with the dimension of the manifold held fixed. The others typically need to increase the dimension in order to decrease the error from the inertial manifold. The sequence implemented here is distinguished from the earlier sequence in [10] in that the computational complexity of the algorithm is independent of the dimension of the manifold.

*Received by the editors February 11, 1999; accepted for publication (in revised form) October 30, 2000; published electronically April 26, 2001. This work was supported in part by National Science Foundation grants DMS-9706903 and DMS-9705229, CNPq, Brasília, Brazil, and FAPERJ, Rio de Janeiro, Brazil. Computations were carried out on facilities made possible by NSF grant CDA-9601632. Some of the work was completed at the Institute for Mathematics and its Applications, University of Minnesota, and the Center for Nonlinear Studies, Los Alamos National Laboratory.
http://www.siam.org/journals/sisc/22-6/35173.html

†Department of Mathematics, Indiana University, Rawles Hall, Bloomington, IN 47405 (msjolly@indiana.edu).

‡Instituto de Matematica, Universidade Federal do Rio De Janeiro, Rio De Janeiro, RJ 21945-970, Brazil (rrosa@labma.ufrj.br).

§Laboratoire d'Analyse Numérique, Université Paris Sud, Orsay 91405, France, and The Institute for Scientific Computing and Applied Mathematics, Indiana University, Bloomington, IN 47405 (temam@indiana.edu).

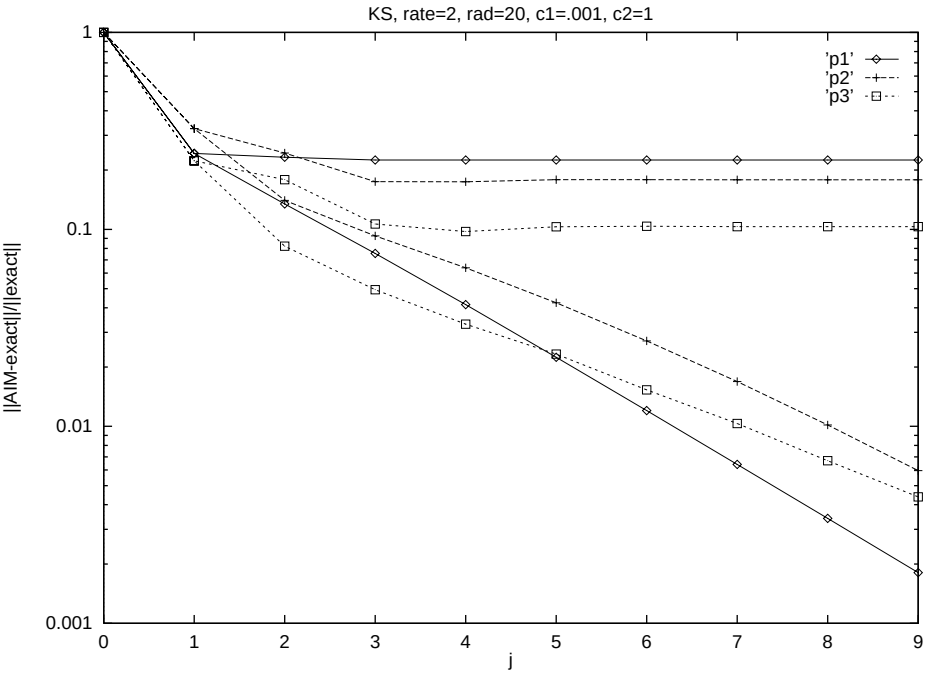


FIG. 1. *Relative error for Φ_j, Ψ_j at the points p_1,p_2, and p_3. The three upper plots are for Ψ_j, and the three lower plots are for Φ_j.*

If for a particular low-mode vector the corresponding high-mode vector is desired, then this algorithm can be applied without computing the high-mode vectors corresponding to other low-mode vectors. Its primary purpose, then, is not the construction of all or even a major portion of such a manifold. Rather, it is best suited for computing solutions *on* such manifolds, particularly, *backward in time*.

In the case of a PDE after spatial discretization, the dimension of a full stable manifold would increase with enhanced spatial accuracy. This is a situation where restricting the system to an inertial manifold can be quite useful. Doing so automatically picks out the relevant stable submanifold, which has low dimension and hence low complexity, so that it can be computed in a straightforward fashion, regardless of whether it consists of one or more connecting orbits. Precisely, this situation arises with the Kuramoto–Sivashinsky equation (KSE). The stability of connecting orbits for the KSE was demonstrated in [8] by means of a reduction to an approximate inertial form (given by Ψ_2 as defined in (3.8)). That work extends the parameter range for stability previously obtained in [1] by perturbative techniques. The stable and unstable manifolds which collide to form these connecting orbits were visualized in [32] and found to be intimately involved in the formation of a completely different set of Silnikov orbits. The computations in [32] were carried out for the same approximate inertial form (up to high order terms) as used in [8].

The point we wish to make is that we can now compute the inertial form for the KSE much more accurately than ever before. This is demonstrated by Figure 1, where the relative error in computing the manifold is measured at three locations on the global attractor, denoted p_1, p_2, and p_3 in Figure 2. The three upper plots are for the sequence of AIMs used in [8] and [32], and earlier in [33], [49]. As indicated by the leveling-off, there is rapid convergence in these instances, albeit to a manifold which is

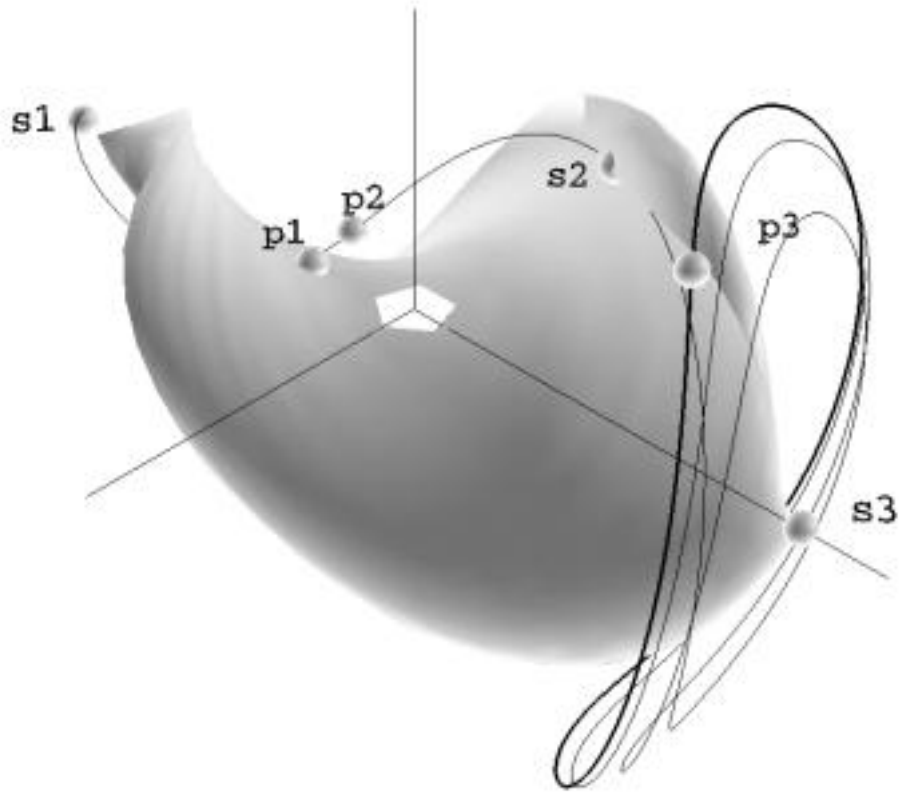


FIG. 2. *Elements of the global attractor for the KSE at* $L = L^*$.

not an inertial manifold, or even invariant. In contrast, the three lower plots indicate that the algorithm of interest here continues to converge, through nine iterations, to the attractor and hence the inertial manifold. These results are discussed in detail in section 4.

The general framework for the method is that of an evolutionary equation of the form

$$\frac{du}{dt} + Au = f(u), \quad u \in E, \quad u(0) = u_0, \tag{1.1}$$

where A is a linear operator and E is a Banach space. It is easiest to describe the approach in the particular case where A has a complete set of eigenvectors associated with a sequence of positive eigenvalues and E is a Hilbert space. An inertial manifold is often sought as the graph of a function $\Phi : PE \to QE$, where $P = P_n$ is the projector onto the span of the first n eigenfunctions of A and $Q = Q_n = I - P$. The

restriction of the flow to an inertial manifold yields the inertial form

$$\frac{dp}{dt} + Ap = Pf(p + \Phi(p)), \quad p(0) = p_0 \in PE, \tag{1.2}$$

which is a finite set of ODEs with the same long-time dynamics as the original evolutionary equation, which may very well be infinite-dimensional. This framework is similar to that for a more general class of invariant manifolds. The algorithm discussed here may be applied under less stringent conditions to local (un)stable manifolds. It has been adapted in [36] to compute inertial manifolds with delay and in [35] to compute center manifolds.

The existence of an inertial manifold is known for a number of physical systems including the KSE, the Cahn–Hilliard equation, and certain reaction diffusion equations in several space dimensions, just to name a few (see [7], [41], [51], and references therein). A great variety of numerical schemes based on AIMs have been developed, analyzed, and tested in the literature (see [13], [49], [20], and references therein). One issue is how much, if any, such methods can save in computing time. This article is not concerned with this question. Instead, we wish to demonstrate that one can accurately compute an inertial manifold so that the dimension of phase space can be kept quite small. Such a reduction can help in the geometric understanding of certain global bifurcations.

In the next section we provide the general assumptions for both the existence of an inertial manifold as well as the valid application of the algorithm. The algorithm itself is recalled in section 3, along with several rigorous estimates on its convergence from [47]. We then apply the method to the KSE in section 4. The computations in this case are compared to those of another sequence of AIMs. To carry out the computations, we are forced to truncate to a finite number of high modes. We provide error estimates for this effect in section 5.

2. General assumptions. The following assumptions guarantee both the existence of an inertial manifold and the convergence of the algorithm used here. While the framework is somewhat technical, we will need to refer to each element described below when we validate the rate of convergence in our implementation. The generality is motivated by the application to PDEs; the assumptions simplify considerably in the case of an ODE.

A1. The nonlinear term f in (1.1) is globally Lipschitz continuous from a Banach space E into another Banach space F,

$$|f(u) - f(v)|_F \le M_1 |u - v|_E \qquad \forall\, u, v \in E, \tag{2.1}$$

with

$$E \subset F \subset \mathcal{E};$$

the injections being continuous, each space is dense in the following one, and $\mathcal{E}$ is also a Banach space. It follows that

$$|f(u)|_F \le M_0 + M_1 |u|_E \qquad \forall\, u \in E,$$

for some $M_0 \ge 0$. (Actually, $M_0 = |f(0)|_F$ is optimal.)

A2. The linear operator $-A$ generates a strongly continuous semigroup $\{e^{-tA}\}_{t \ge 0}$ of bounded operators on $\mathcal{E}$ such that

$$e^{-tA} F \subset E \qquad \forall\, t > 0.$$

A3. There exist two sequences of numbers $\{\lambda_n\}_{n=n_0}^{n_1}$, $\{\Lambda_n\}_{n=n_0}^{n_1}$ for some $n_0 \in \mathbb{N}$, $n_1 \in \mathbb{N} \cup \infty$ with $0 < \lambda_n \leq \Lambda_n$ for $n_0 \leq n \leq n_1$, and a sequence $\{P_n\}_{n=n_0}^{n_1}$ of finite dimensional projectors, such that if $Q_n = I - P_n$, then the following exponential dichotomies hold.

$P_n\mathcal{E}$ is invariant under e^{-tA} for $t \geq 0$, and $\{e^{-tA}|_{P_n\mathcal{E}}\}_{t\geq 0}$ can be extended to a strongly continuous group $\{e^{-tA}P_n\}_{t\in\mathbb{R}}$ of bounded operators on $P_n\mathcal{E}$ with

$$\begin{aligned} &\|e^{-tA}P_n\|_{\mathcal{L}(E)} \leq K_1 e^{-\lambda_n t} \qquad \forall\ t \leq 0, \\ &\|e^{-tA}P_n\|_{\mathcal{L}(F,E)} \leq K_1 \lambda_n^{\alpha} e^{-\lambda_n t} \qquad \forall\ t \leq 0. \end{aligned} \tag{2.2}$$

$Q_n\mathcal{E}$ is positively invariant under e^{-tA} for $t \geq 0$ with

$$\begin{aligned} &\|e^{-tA}Q_n\|_{\mathcal{L}(E)} \leq K_2 e^{-\Lambda_n t} \qquad \forall\ t \geq 0, \\ &\|e^{-tA}Q_n\|_{\mathcal{L}(F,E)} \leq K_2 (t^{-\alpha} + \Lambda_n^{\alpha}) e^{-\Lambda_n t} \qquad \forall\ t > 0, \end{aligned} \tag{2.3}$$

where $K_1, K_2 \geq 1$, and $0 \leq \alpha < 1$. We will at times drop the subscripts on the projectors P and Q for simplicity.

A4. Equation (1.1) has a continuous semiflow $\{S(t)\}_{t\geq 0}$ in E, given by $S(t)u_0 = u(t)$, where $u = u(t)$ is the mild solution of (1.1) defined through the variation of constants formula

$$u(t) = e^{-tA}u_0 + \int_0^t e^{-(t-s)A} f(u(s))ds \qquad \forall\ t \geq 0.$$

A5. There exists $K_3 \geq 0$ independent of n such that

$$\|AP_n\|_{\mathcal{L}(E)} \leq K_3 \lambda_n.$$

A6. For simplicity we assume that A is invertible.

A7. The *spectral gap condition* (SGC)

$$\Lambda_n - \lambda_n > 3M_1K_1K_2[\lambda_n^{\alpha} + (1+\gamma_{\alpha})\Lambda_n^{\alpha}] \quad \text{holds for some } n \in \mathbb{N}, \tag{2.4}$$

where

$$\gamma_{\alpha} = \begin{cases} \int_0^{+\infty} e^{-r} r^{-\alpha} dr & \text{if } 0 < \alpha < 1, \\ 0 & \text{if } \alpha = 0. \end{cases}$$

In many PDE applications, the spectrum of A consists of positive semisimple eigenvalues

$$\mu_1 \leq \mu_2 \leq \cdots \text{ with } \mu_n \to +\infty, \tag{2.5}$$

associated with eigenvectors $w_1, w_2, \ldots$, so that we may set $\lambda_n = \mu_n$, $\Lambda_n = \mu_{n+1}$, and

$$P_n = \text{projector onto span } \{w_1, w_2, \ldots, w_n\}.$$

In most applications the phase space can be taken to be a Hilbert space. For some parabolic equations, however, depending on the relationship between the dimension of the space, the order of the linear term, and the strength of the nonlinear term, the appropriate phase space is merely a Banach space (see [40], for instance). Strictly

speaking, one needs only to have the exponential dichotomy in A3 to hold for the particular choice of n satisfying the SGC in A7, though typically one has many choices for the splitting in A3, and the SGC holds only for *some* large n. In some of the cases we will consider here, one or more of the first eigenvalues are negative, so we simply neglect those and start with Λ_{n_0} as the first positive eigenvalue and choose λ_{n_0} to be some positive number smaller than Λ_{n_0}. If the eigenvalues were semisimple but complex, one would take λ_n and Λ_n to be the real part of the eigenvalues. The assumption that the projector P_n is finite-dimensional excludes the cases in which A has a continuous spectrum, but that is only because we want a finite-dimensional inertial manifold in the end. The whole construction would go through if we allow P_n to be infinite-dimensional and hence A to have bands with continuous spectrum, and we would end up with an infinite-dimensional "inertial manifold."

In most applications the global Lipschitz condition in (2.1) does not hold for the evolutionary equation in its original form. This can be corrected by annihilating the nonlinearity outside a ball in phase space. To do so and to be assured that none of the long-time dynamics are altered, we require in this situation that the evolutionary system be dissipative. This means there is an absorbing ball $B_\rho \subset E$, i.e., there exists $\rho > 0$ such that

$$|S(t)u_0|_E \le \rho \quad \forall\ t \ge t_0(|u_0|_E) \ge 0.$$

The prepared evolutionary equation is to have the nonlinearity (and hence the dynamics) unchanged inside the ball and has the nonlinearity set to zero outside an even larger ball. We give a specific preparation procedure in section 4.

We briefly recall below the construction of the *exact* inertial manifold as done in [48]. The discretization of this approach is what leads to the algorithm tested here. There are a number of earlier alternative approaches to prove the existence of inertial manifolds which are mostly based on either the Lyapunov–Perron method (c.f. [17]) or the graph transform method of Hadamard (c.f. [7]), and which construct the *entire* manifold at once, as the graph of a function $\Phi : P_nE \to Q_nE$. This is in contrast to the approach here, a variation of the Lyapunov–Perron method in which for a given $p_0 \in P_nE$, the corresponding image $\Phi(p_0)$ is found in terms of a single trajectory on the manifold. Thus it is a one-dimensional object (the trajectory) which is discretized, rather than an n-dimensional object (the manifold). Earlier use of this approach can be found in [26], [27], [5], and [43].

The construction is described in two steps. First, we obtain an invariant manifold. Then the exponential attraction property is established, making the manifold in fact an inertial manifold. When (2.4) holds, we can find σ such that

$$\lambda_n + 2M_1K_1\lambda_n^\alpha < \sigma < \Lambda_n - 2M_1K_2(1+\gamma_\alpha)\Lambda_n^\alpha. \tag{2.6}$$

For σ in this range, a single trajectory on an invariant manifold can be found as the fixed point $\varphi = \varphi(p)$ of a mapping $\mathcal{T}(\cdot,p)$ given by

$$\begin{aligned}\mathcal{T}(\varphi,p)(t) = e^{-tA}p &- \int_t^0 e^{-(t-s)A}Pf(\varphi(s))ds \\ &+ \int_{-\infty}^t e^{-(t-s)A}Qf(\varphi(s))ds \quad \forall\ t \le 0, \quad \forall\ p \in PE,\end{aligned} \tag{2.7}$$

in the Banach space

$$\mathcal{F}_\sigma = \left\{ \varphi \in C\left((-\infty,0],E\right); \|\varphi\|_\sigma = \sup_{t\le 0}(e^{\sigma t}|\varphi(t)|_E) < +\infty \right\}.$$

The entire manifold $\mathcal{M}$ is the collection of such trajectories given by $\mathcal{M} = \text{graph}\Phi$, where $\Phi : PE \to QE$ is defined by the fixed point φ of (2.7),

$$\Phi(p) = Q\varphi(p)(0) \qquad \forall\ p \in PE.$$

The mapping $\mathcal{T}$ is a strict contraction in $\mathcal{F}_\sigma$, uniformly in PE, with

$$\|\mathcal{T}(\varphi_1, p) - \mathcal{T}(\varphi_2, p)\|_\sigma \leq \kappa_{n,\sigma}\|\varphi_1 - \varphi_2\|_\sigma \quad \forall\ \varphi_1, \varphi_2 \in \mathcal{F}_\sigma, \quad \forall\ p \in PE, \tag{2.8}$$

where

$$\kappa_{n,\sigma} \equiv \frac{M_1 K_1 \lambda_n^\alpha}{\sigma - \lambda_n} + \frac{M_1 K_2 (1+\gamma_\alpha)\Lambda_n^\alpha}{\Lambda_n - \sigma} < 1. \tag{2.9}$$

In order to show that the invariant manifold constructed as above is in fact an inertial manifold, all that remains is to establish the exponential attraction property. In fact, the SGC in A7 assures a stronger property sometimes referred to as *asymptotic completeness* or *exponential tracking* [18]. This means that corresponding to any initial condition $u_0 \in E$ there exists a particular solution *on* the manifold, to which the trajectory through u_0 is attracted at an exponential rate. The so-called relevance theorem asserts the same property for the center manifold of a fixed point provided the fixed point is stable [3].

The proof of the asymptotic completeness in [48] requires that σ be taken in a narrower range,

$$\lambda_n + 3M_1K_1K_2\lambda_n^\alpha < \sigma < \Lambda_n - 3M_1K_1K_2(1+\gamma_\alpha)\Lambda_n^\alpha. \tag{2.10}$$

In this case one has, in fact, that $\kappa_{n,\sigma} < 2/3$ so that the rate of contraction is guaranteed to be faster. Thus if we replace the gap condition in A7 with the weaker condition

$$\Lambda_n - \lambda_n > 2M_1K_1\lambda_n^\alpha + 2M_1K_2(1+\gamma_\alpha)\Lambda_n^\alpha, \tag{2.11}$$

we can satisfy (2.6) but not necessarily (2.10) and consequently have an invariant manifold $\mathcal{M} = \text{graph}\Phi$, which is not necessarily an inertial manifold. If (2.4) holds, then $\mathcal{M} = \text{graph}\Phi$ is indeed an inertial manifold.

We should mention that the same map in (2.7) is used in [43] in an alternate framework, where the contraction is obtained in a different space, leading to an SGC that is in fact weaker than (2.4) yet still gives both the invariance and exponential attraction. When applicable, the approach in [43] can provide a manifold of smaller dimension than that provided by A1–A7. Our error analysis for the discretization of the map in (2.7) will, however, use the assumptions in A1–A7.

A related object of study is the global attractor $\mathcal{A}$. Its existence is guaranteed provided the system is dissipative and the semiflow $S(t)$ is compact [25], [51]. In this case it can be defined by

$$\mathcal{A} = \bigcap_{t \geq 0} S(t)B_\rho.$$

This is the largest bounded invariant set for the system and contains all steady states, periodic solutions, and invariant tori, as well as the unstable manifolds associated with those objects. The global attractor must be contained in any inertial manifold and thus provides a lower bound on the minimal dimension for all inertial manifolds. Unlike inertial manifolds, the global attractor can be quite complicated geometrically and attract solutions at an algebraic rate.

3. The algorithm. To discretize the mapping $\mathcal{T}$, the function space $\mathcal{F}_\sigma$ is replaced by

$$\tilde{\mathcal{F}} = \{\psi : \ (-\infty, 0] \ \to E; \psi \text{ is piecewise constant with a finite number of discontinuities}\}.$$

Given a time step $\tau > 0$, a number of steps $N \in \mathbb{N}$, and a base point $p_0 \in PE$, we define $\mathcal{T}_\tau^N : \tilde{\mathcal{F}} \times PE \to \tilde{\mathcal{F}}$ by

$$\begin{aligned} \mathcal{T}_\tau^N(\psi, p_0)(t) = e^{k\tau A} P p_0 - \int_{-k\tau}^{0} e^{-(-k\tau - s)A} P f(\psi(s)) ds \\ + \int_{-\infty}^{-k\tau} e^{-(-k\tau - s)A} Q f(\psi(s)) ds, \qquad k = 0, 1, \dots, N, \end{aligned} \tag{3.1}$$

for $t \in (-(k+1)\tau, -k\tau]$ if $k < N$, or $t \in (-\infty, -N\tau]$ if $k = N$.

Note that $\mathcal{T}_\tau^N$ acts as a sort of "projection" of $\mathcal{T}$ on $\tilde{\mathcal{F}}$ in that

$$\mathcal{T}_\tau^N(\psi, p_0)(-k\tau) = \mathcal{T}(\psi, p_0)(-k\tau).$$

We now consider the particular sequence of AIMs in [47] by assigning as the initial guess

$$\varphi^0(p_0)(t) = p_0 \qquad \forall\ t \le 0, \qquad \forall\ p_0 \in PE, \tag{3.2}$$

and by choosing two sequences $\{\tau_j\}_{j\in\mathbb{N}}, \tau_j > 0$ and $\{N_j\}_{j\in\mathbb{N}}, N_j \in \mathbb{N}, N_j \ge 0$. Proceeding by Picard iteration, we have

$$\varphi^j(p_0) = \mathcal{T}_{\tau_j}^{N_j}(\varphi^{j-1}(p_0), p_0) \qquad \forall\ p_0 \in PE, \tag{3.3}$$

for $j = 1, 2, \dots$. Observe that τ_0 and N_0 are not relevant since $\varphi^0(p_0)$ is constant. The convergent sequence of AIMs is then given by

$$\mathcal{M}_j = \mathcal{M}_{j,n} = \text{ graph } \Phi_j = \{p + \Phi_j(p);\ p \in P_n E\},$$

where

$$\Phi_j(p_0) = Q_n \varphi^j(p_0)(0) \qquad \forall\ p_0 \in P_n E.$$

Again, since the approximating trajectories φ^j belong to $\tilde{\mathcal{F}}$, the integrals involved in the iteration process (3.3) can be explicitly calculated. For simplicity we denote

$$\varphi_k^j(p_0) = \varphi^j(p_0)(-k\tau_j), \qquad k = 0, 1, \dots, N_j,$$

for $p_0 \in PE$ and $j = 0, 1, \dots$, so that $\Phi_j = Q\varphi_0^j$. For the case where $k\tau_j < N_{j-1}\tau_{j-1}$, we have

$$\begin{aligned} \varphi_k^j(p_0) = e^{k\tau_j A} P p_0 - \int_{-k\tau_j}^{-\ell_k \tau_{j-1}} e^{-(-k\tau_j - s)A} P f(\varphi_{\ell_k}^{j-1}(p_0))\ ds \\ - \sum_{\ell=0}^{\ell_k - 1} \int_{-(\ell+1)\tau_{j-1}}^{-\ell\tau_{j-1}} e^{-(-k\tau_j - s)A} P f(\varphi_\ell^{j-1}(p_0))\ ds \\ + \int_{-\infty}^{-N_{j-1}\tau_{j-1}} e^{-(-k\tau_j - s)A} Q f(\varphi_{N_{j-1}}^{j-1}(p_0))\ ds \\ + \sum_{\ell=\ell_k+1}^{N_{j-1}-1} \int_{-(\ell+1)\tau_{j-1}}^{-\ell\tau_{j-1}} e^{-(-k\tau_j - s)A} Q f(\varphi_\ell^{j-1}(p_0))\ ds \\ + \int_{-(\ell_k+1)\tau_{j-1}}^{-k\tau_j} e^{-(-k\tau_j - s)A} Q f(\varphi_{\ell_k}^{j-1}(p_0))\ ds, \end{aligned} \tag{3.4}$$

where ℓ_k is a nonnegative integer defined by

$$\begin{cases} -(\ell_k+1)\tau_{j-1} < -k\tau_j \le -\ell_k\tau_{j-1} & \text{if } -(N_{j-1}+1)\tau_{j-1} < -k\tau_j, \\ \ell_k = N_{j-1} & \text{otherwise.} \end{cases} \tag{3.5}$$

The alternative case, where $k\tau_j \ge N_{j-1}\tau_{j-1}$, is actually simpler, as φ^{j-1} is constant over $(-\infty, k\tau_j]$. By a straightforward calculation of the integrals we have the following.

Case 1. $k\tau_j < N_{j-1}\tau_{j-1}$:

$$\begin{aligned} \varphi_k^j(p_0) &= e^{k\tau_j A}P_n p_0 - A^{-1}(e^{(k\tau_j-\ell_k\tau_{j-1})A}P_n - P_n)f(\varphi_{\ell_k}^{j-1}(p_0)) \\ &\quad - \sum_{\ell=0}^{\ell_k-1} A^{-1}(e^{(k\tau_j-\ell\tau_{j-1})A}P_n - e^{(k\tau_j-(\ell+1)\tau_{j-1})A}P_n)f(\varphi_\ell^{j-1}(p_0)) \\ &\quad + A^{-1}e^{-(N_{j-1}\tau_{j-1}-k\tau_j)A}Q_n f(\varphi_{N_{j-1}}^{j-1}(p_0)) \\ &\quad + \sum_{\ell=\ell_k+1}^{N_{j-1}-1} A^{-1}(e^{-(\ell\tau_{j-1}-k\tau_j)A} - e^{-((\ell+1)\tau_{j-1}-k\tau_j)A})Q_n f(\varphi_\ell^{j-1}(p_0)) \\ &\quad + A^{-1}(I - e^{-((\ell_k+1)\tau_{j-1}-k\tau_j)A})Q_n f(\varphi_{\ell_k}^{j-1}(p_0)). \end{aligned} \tag{3.6}$$

Case 2. $k\tau_j \ge N_{j-1}\tau_{j-1}$:

$$\begin{aligned} \varphi_k^j(p_0) &= e^{k\tau_j A}P_n p_0 - A^{-1}(e^{(k\tau_j-N_{j-1}\tau_{j-1})A}P_n - P_n)f(\varphi_{N_{j-1}}^{j-1}(p_0)) \\ &\quad - \sum_{\ell=0}^{N_{j-1}-1} A^{-1}(e^{(k\tau_j-\ell\tau_{j-1})A}P_n - e^{(k\tau_j-(\ell+1)\tau_{j-1})A}P_n)f(\varphi_\ell^{j-1}(p_0)) \\ &\quad + A^{-1}Q_n f(\varphi_{N_{j-1}}^{j-1}(p_0)). \end{aligned} \tag{3.7}$$

3.1. Comparison with other AIMs. Regardless of the choice of the sequences $\{\tau_j\}_{j\in\mathbb{N}}$ and $\{N_j\}_{j\in\mathbb{N}}$, the zeroth AIM is given by $\Phi_0 \equiv 0$ since $Q_n\varphi^0 \equiv 0$ for φ^0 given in (3.2). The first nontrivial approximation is given by $\Phi_1(p) = A^{-1}Qf(p)$. An alternative sequence of AIMs is defined implicitly (for large enough n, see [19], [16]) by the Q-component of the steady state equation

$$Ap = Qf(p+q).$$

This alternative sequence is found recursively by

$$\Psi_{j,n}(p) = \Psi_j(p) = A^{-1}Q_n f(p + \Psi_{j-1}(p)), \quad \Psi_0(p) \equiv 0, \quad p \in P_n\mathbb{H}. \tag{3.8}$$

Note that $\Phi_1 = \Psi_1$, which happens also to coincide with a manifold first introduced in [14] with a completely different derivation. For a number of PDE cases, in particular, for the KSE and two-dimensional Navier–Stokes equations, it has been shown that the $\mathcal{N}_{j,n} = \text{graph } \Psi_{j,n}$ converge to the global attractor $\mathcal{A}$ in the sense that

$$\text{dist}_E(\mathcal{A}, \mathcal{N}_{j,n}) \to 0 \quad \text{as } n \to \infty \tag{3.9}$$

for *any* fixed j, even for $j = 0$ (see [34], [53]). The convergence in (3.9) is valid even in the absence of the SGC, i.e., it is independent of the existence of an actual inertial manifold. Much of the motivation behind nonlinear Galerkin methods based

on such AIMs comes from an algebraic improvement in the estimates for the rate of convergence to the attractor for $j > 0$ compared to that for $j = 0$. In some cases, however, solutions to a PDE can be shown to be in the Gevrey class with respect to the spatial variable so that this improvement is on top of a decay rate in the wave number that is already exponential [23], [39]. A number of other AIMs have been introduced in an attempt to improve the approximation provided by $\mathcal{N}_{j,n}$ (see, for instance, [50], [11], [9]), but in each case one must increase the dimension of the manifold in order to converge to the attractor.

The distinction we wish to emphasize between the sequences $\{\mathcal{M}_{j,n}\}$ and $\{\mathcal{N}_{j,n}\}$ (and many like the latter) is that in the case of $\{\mathcal{M}_{j,n}\}$ we may instead fix n, hence the dimension of the manifolds, and achieve convergence as $j \to \infty$. We restate below, in slightly more generality, the main convergence result proved in [47].

THEOREM 3.1. *Assume that* A1–A6 *hold and that the sequences* $\{\tau_j\}_j$ *and* $\{N_j\}_j$ *are chosen so that*

$$0 < \tau_j \le c_1 \gamma^j \kappa_{n,\sigma}^j \qquad \forall\ j \in \mathbb{N}$$

with $\kappa_{n,\sigma}$ *as defined in* (2.9) *for* n *satisfying* (2.4) *(resp.,* (2.11)*), and*

$$N_j \tau_j \ge c_2 j \qquad \forall\ j \in \mathbb{N},$$

for some constants $c_1, c_2 > 0$*, some* $0 < \gamma < 1$*, and some* σ *satisfying* (2.10) *(resp.,* (2.6)*). Then*

$$d(\Phi, \Phi_j) \le c_3 e^{-c_4 j} \qquad \forall\ j \in \mathbb{N},$$

for suitable constants $c_3, c_4 > 0$*, and* $\mathcal{M} =$ *graph* Φ *is an inertial (resp., invariant) manifold.*

Thus if the SGC holds, then the convergence is to an inertial manifold. If one replaces the gap condition in A7 with the weaker condition (2.11), then the proof as in [47] for the convergence of the sequence $\{\mathcal{M}_{j,n}\}$ as $j \to \infty$ still goes through, but not the asymptotic completeness of, or even the exponential attraction to, the limiting manifold. The limiting manifold in that case is merely guaranteed to be invariant. Finally, in the absence of (2.11), the convergence to the global attractor is still guaranteed, provided $n \to \infty$ as well [47].

In the original method of proof for the existence of inertial manifolds [17], the manifold is the graph of the fixed point of the mapping described by

$$\Theta_{j+1}(p_0) = \int_{-\infty}^{0} e^{sA} Q f(\tilde{p} + \Theta_j(\tilde{p})) ds, \tag{3.10}$$

where $\tilde{p} = \tilde{p}(s; p_0, \Theta_j)$ solves

$$\frac{d\tilde{p}}{dt} + A\tilde{p} = Pf(\tilde{p} + \Theta_j(\tilde{p})), \quad \tilde{p}(0) = p_0. \tag{3.11}$$

In [10], a discretization of this mapping was presented and shown to provide a convergent algorithm for computing inertial manifolds. Such a discretization was actually implemented in [46] to compute the manifold for a reaction diffusion equation over a portion of $P_n\mathcal{E}$ in a case where $n = 2$. Note that to obtain Θ_{j+1} requires global knowledge of Θ_j as one cannot predict where the trajectory in $P\mathcal{E}$ of the solution to (3.11) will go, i.e., where one will need to evaluate Θ_j. Practically speaking, one

must interpolate the discrete representation of Θ_j to solve (3.11) backward in time. Thus the complexity of the computation grows dramatically with the dimension of the manifold n. Yet, if one is interested in computing the complete manifold (or at least a patch of it), the approach in [10] is probably more appropriate than the one used here.

If, however, one is primarily interested in computing particular solutions on the manifold, there is no need to compute the manifold in its entirety. In this case the objective is to solve the initial value problem for the inertial form (1.2) so that at each time step, one needs only to compute the functional relation $q = \Phi(p)$ at a particular $p \in P\mathcal{E}$ (or several such p locations, in a multistep scheme). To emphasize the difference with the approach in (3.10), note that to obtain φ_{j+1} via (2.7), one need only know φ_j, a one-dimensional object. Thus the complexity is essentially independent of the dimension of the manifold. Another application where one would prefer to apply (3.6), (3.7) is in postprocessing via an AIM [21]. Roughly speaking, in that approach one evolves the solution via an adequately accurate Galerkin (or nonlinear Galerkin) approximation to obtain final value $p(t_1)$, and then recovers values for the high modes *only at the final time* from $q(t_1) = \Phi_a(p(t_1))$, where Φ_a is some appropriately chosen AIM. In this instance the enslavement relation is needed only at a single base point.

While Theorem 3.1 gives the main motivation for implementing this algorithm, the validation of a code for this purpose requires more details from the analysis. Specifically, we need information about the convergence in $\tilde{\mathcal{F}}$. This is given in two lemmas below, which are proved in [47]. For both lemmas it is again assumed that A1–A7 hold. The first ingredient is an a priori estimate on the variation in time of the exact solution on the manifold.

LEMMA 3.2. *For $t < 0$ and τ such that $0 \le \tau \le -t$, we have that*

$$|\varphi(p_0)(t+\tau) - \varphi(p_0)(t)|_E \le \tau \beta_{n,\sigma} e^{-\sigma t}(1 + |p_0|_E)$$

for all $p_0 \in P_n E$, where

$$\beta_{n,\sigma} = 2M_0K_1\lambda_n^{\alpha} + \frac{2K_1(K_3\lambda_n + M_1\lambda_n^{\alpha})}{1-\kappa_{n,\sigma}}\left\{\frac{M_0K_1}{\lambda_n^{1-\alpha}} + \frac{(1+\gamma_\alpha)M_0K_2}{\Lambda_n^{1-\alpha}} + K_1\right\}. \tag{3.12}$$

It is not so much that we need to invoke the result of Lemma 3.2 as that we need the quantity $\beta_{n,\sigma}$, which plays a central role in the error estimate for the algorithm in (3.6), (3.7). The reason we restate Lemma 3.2 here is to recall the purpose of $\beta_{n,\sigma}$. While the expression in (3.12) is adequate under the general assumptions A1–A7, it is conceivable that for a particular PDE, one might know more about the variation of solutions, in which case a smaller expression for $\beta_{n,\sigma}$ might be found. The next lemma measures the actual error after applying the mapping $\tilde{\mathcal{T}}$. It states precisely how $\beta_{n,\sigma}$ enters into the estimate. We do not have a contraction in the strict sense. Rather, we have a contraction plus a residual which decreases provided the time step τ decreases and the product $N\tau$ increases.

LEMMA 3.3. *Let N be a nonnegative integer, let $\tau > 0$, and assume that σ and σ_0 satisfy* (2.10) *for a fixed $n \in \mathbb{N}$ with $\sigma_0 < \sigma$. Then for all $p_0 \in P_n E$ and all $\psi \in \tilde{\mathcal{F}}$,*

we have

$$\begin{aligned}\|\varphi(p_0) - \mathcal{T}_\tau^N(\psi, p_0)\|_\sigma &\le \kappa_{n,\sigma}\|\varphi(p_0) - \psi\|_\sigma \\ &+ \left(\tau\beta_{n,\sigma} + \frac{\beta_{n,\sigma_0}}{\sigma - \sigma_0}e^{-(\sigma-\sigma_0)N\tau}\right)(1 + |p_0|_E),\end{aligned} \tag{3.13}$$

where β is as in Lemma 3.2.

The expression for $\beta_{n,\sigma}$ serves as a guide to choosing a number of parameter settings for the algorithm. We will examine in section 5 the effects of particular terms in $\beta_{n,\sigma}$ on the error.

4. Application to the KSE. The KSE we consider is often written as

$$\frac{\partial u}{\partial t} + \frac{\partial^4 u}{\partial x^4} + \frac{\partial^2 u}{\partial x^2} + u\frac{\partial u}{\partial x} = 0, \quad (x,t) \in \mathbb{R} \times \mathbb{R}^+, \quad u(x,0) = u_0(x), \quad x \in \mathbb{R}, \tag{4.1}$$

subject to periodic boundary conditions $u(x,t) = u(x+L,t)$, $L > 0$. A considerable amount of theoretical and computational work on the KSE can be found in the literature (see [1], [4], [28], and [42] for a small sampling). The dissipativity for the KSE was first established only in the invariant subspace of odd functions in [44]. In fact, it was the KSE with periodic *and odd* boundary conditions that served as one of the first applications of the existence theory of inertial manifolds [15]. Dissipativity for the general periodic case has been established only somewhat recently, independently by Collet et al. [6] and Goodman [22] (see also the earlier work of Il'yashenko [29] and some simplifications in Pinto [45]). This paved the way for the existence of a global attractor and an inertial manifold in the general case. Nevertheless, in the computations which follow at the end of this section, *we restrict the flow to the odd subspace*, in part, to cut the dimension of the inertial manifold in half.

The dimension of the inertial manifold for the KSE was first estimated in [15] and later improved in [52] by using the estimate for the absorbing ball from [6]. These estimates, however, are of the form dim $\le cL^b$, where most of the effort is devoted to making the exponent b as small, yet the value of the constant c is not readily available. There is an arithmetic error in [52] which leads to a false estimate of dim $\le cL^{1.64}(\ln L)^{0.2}$ for the inertial manifold in the space L^2, when in fact that analysis should yield dim $\le cL^{2.7}(\ln L)^{0.2}$.

We wish to apply the algorithm at the moderate value of $L = L^* = 4\sqrt{2}\pi$ and thus must actually *evaluate* the dimension. This calculation has been carried out in [37] to arrive at rigorous values for the dimension for the KSE prepared at radii ranging from that of a ball which can be shown to be absorbing, down to a much smaller one which nevertheless seems to contain the global attractor. We sketch briefly here the main steps involved.

In the appendix to [37], A. Cheskidov calculates a rigorous radius of an absorbing ball in L^2 to be $\rho_0 = 972$ at $L = L^*$. (The same radius is valid for the general periodic case, even though the dimension of the inertial manifold will be larger.) Also in [37], we rework the analysis in [52] to obtain an explicit estimate for ρ_1, the radius of an absorbing ball in H^1, in terms of ρ_0.

The equation is prepared in a manner quite different from what is traditionally done, which is to replace f by f_ρ, where $f_\rho(u) = \chi_\rho(|u|_\diamond)f(u)$, with

$$\chi_\rho(r) = \chi\left(\frac{r^2}{\rho^2}\right) \tag{4.2}$$

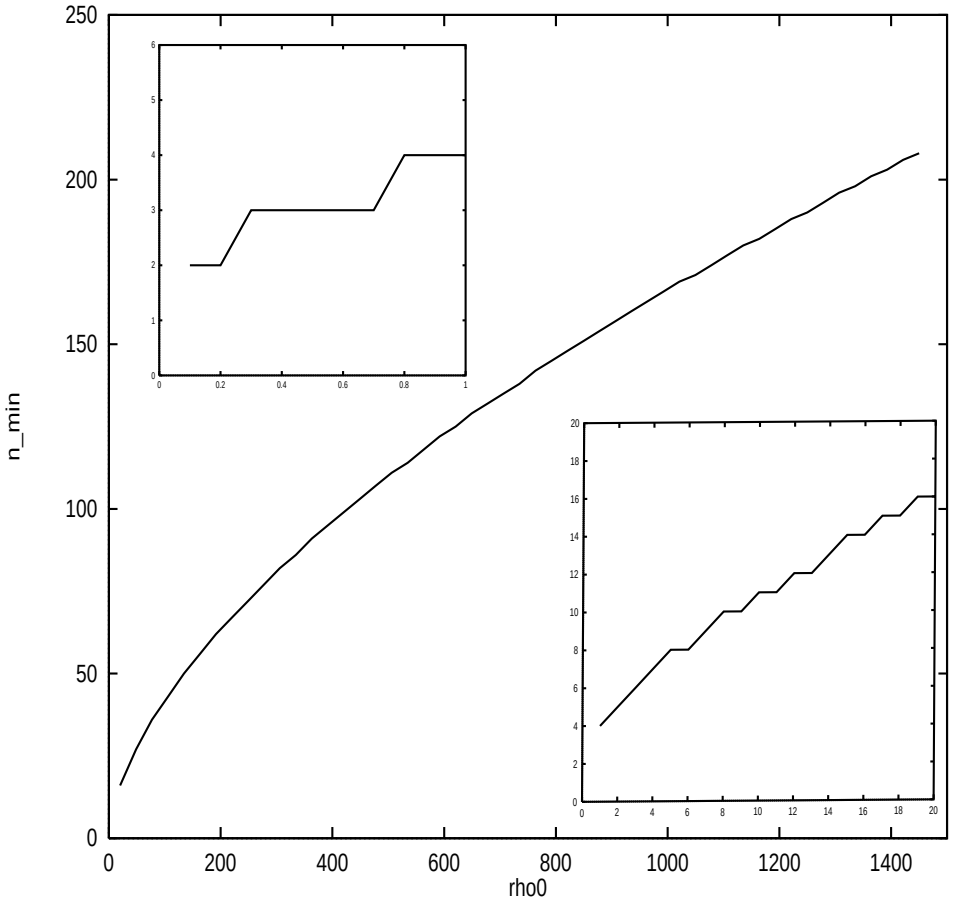


FIG. 3. *Minimal dimension of an inertial manifold for prepared KSE.*

for some norm $|\cdot|_\diamond$ and some function $\chi \in C^1(\mathbb{R}^+)$ satisfying

$$\chi|_{[0,1]} = 1, \quad \chi|_{[2,\infty)} = 0, \quad 0 \le \chi(s) \le 1 \quad \forall\, s \in [1,2], \quad \forall\, s \in \mathbb{R}^+. \tag{4.3}$$

Instead, we let $|\cdot|$, $|\cdot|_s$ denote the L^2 and H^s norms, respectively, and we show (in [37]) that

$$\left| u\frac{\partial u}{\partial x} - v\frac{\partial v}{\partial x} \right|_{-1} \le \sqrt{2}\rho_0^{1/2}\rho_1^{1/2}|u-v| \quad \text{provided } |u|, |v| \le \rho_0 \text{ and } |u|_1, |v|_1 \le \rho_1,$$

and thus the nonlinear term $f(u) = u\partial u/\partial x$ when restricted to the ball in H^1 is Lipschitz from the L^2-topology to the H^{-1}-topology. We then invoke the following theorem of Valentine (see [30]).

THEOREM 4.1. *Let E, F be Hilbert spaces, and let S be a subset of E. For any Lipschitz function $g : S \to F$ there exists a Lipschitz function $\tilde{g} : E \to F$ such that $\tilde{g}|_S = g$ and* $\mathrm{Lip}(\tilde{g}) = \mathrm{Lip}(g)$.

Thus we can extend the restriction of f to the ball in H^1 to a globally Lipschitz function from L^2 to H^{-1}, keeping the Lipschitz constant fixed at $M = \sqrt{2}\rho_0^{1/2}\rho_1^{1/2}$. Numerical evaluation of both sides of the sharp gap condition in [43] leads to a minimal

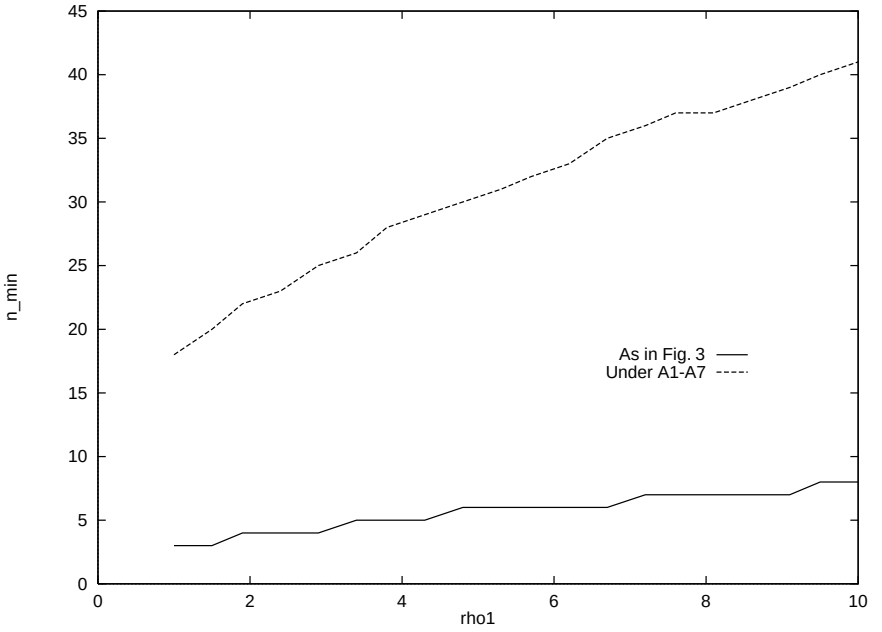


FIG. 4. *Minimal dimension under two different sets of assumptions.*

dimension of 164 (for $L = L^*$, in the odd case). If the estimate for the absorbing ball can be improved, so would be the estimate for the dimension of the manifold. This effect is displayed in Figure 3. One can also very well consider a possibly overly prepared equation, where the nonlinear term is modified outside a ball of *arbitrary* radius ρ_0, regardless of the rigorous estimate for the absorbing ball, much like in the case of a local invariant manifold. Any dynamics *entirely* within the ball of preparation are still preserved. On the other hand, we can expect trajectories of the KSE which are even just partially outside the ball to be altered in an indeterminable way.

The particular steps outlined above gave the best results of several explored in [37]. It is under the more general framework in A1–A7, however, that the error estimates in section 4 were derived. We express the KSE in the form of (1.1) by setting

$$Au = D^4u + D^2u + u, \qquad f(u) = -uDu + u, \tag{4.4}$$

so that the eigenvalues of the linear part are

$$\mu_k = \left(\frac{2\pi}{L}k\right)^4 - \left(\frac{2\pi}{L}k\right)^2 + 1, \quad k = 1, 2, \dots,$$

corresponding to a complete set of orthonormal eigenfunctions in L^2_{odd}

$$w_k(x) = \sqrt{\frac{2}{L}} \sin\left(\frac{2\pi}{L}kx\right).$$

Here we set $\lambda_n = \mu_n$ and $\Lambda_n = \mu_{n+1}$ for n to be determined by A7. (The treatment that produced Figure 3 used $Au = D^4u + D^2u$.)

We also select as spaces

$$E = H^1_{\text{odd}}, \quad F = \mathcal{E} = L^2_{\text{odd}}. \tag{4.5}$$

It is shown in [52] that the conditions A1–A6 hold for the spaces E,F, $\mathcal{E}$, as in (4.5) if $K_1 = K_2 = (\frac{4}{3})^{1/4}$, and $\alpha = 1/4$. Furthermore, it is shown in [37] that

$$|f(u) - f(v)| \le d_1(r)|u - v|_1, \quad \text{provided } |u|_1, |v|_1 < r,$$

where

$$d_1(r) = \sqrt{2}\tilde{L}^{1/2}r + \tilde{L}, \tag{4.6}$$

and $\tilde{L} = L/2\pi$.

We prepare the nonlinear term by applying Theorem 4.1 to obtain an extension of $f|_{B_r}$ to a globally Lipschitz function $f_r : H^1_{\text{odd}} \to L^2_{\text{odd}}$ such that $\text{Lip}(f_r) = d_1(r)$ and $f_r = f$ on B_r, the ball of radius r in H^1_{odd}. Thus we may take $M_0 = 0$ and $M_1 = d_1(r)$ in A1. In Figure 4 we plot the minimal dimension to satisfy the conditions A1–A7, along with the data from Figure 3, for comparison. In the latter case, we plot against the value of ρ_1 which is computed in terms of ρ_0, as in [52].

4.1. Computational results. Since we do not have an exact form for the inertial manifold we instead fix the length L and select several points on the global attractor, and hence on any inertial manifold. We then try to reproduce the Q-component of each point using only the P-component. We consider first the case where u is the solution to the eight-dimensional Galerkin approximation, and we proceed under the ansatz that there is a three-dimensional inertial manifold. Eight modes have been found to be the minimum needed for the Galerkin method to capture the correct dynamics; using more modes does not seem to change the qualitative features described below nor to introduce new elements to the global attractor [34], [38].

The computations are carried out on a renormalized version of the KSE

$$\frac{\partial v}{\partial \tau} + 4\frac{\partial^4 v}{\partial y^4} + \vartheta\left[\frac{\partial^2 v}{\partial y^2} + v\frac{\partial v}{\partial y}\right] = 0, \quad y \in [0, 2\pi], \tag{4.7}$$

where $\vartheta = L^2/\pi^2$ is taken to be the new parameter. The change of variables which relates (4.7) to (4.1) is given by

$$u(t,x) = lv\left(\frac{l^4}{4}t, lx\right) \quad \text{so that } |u|_1 = |u_x|_{L^2(0,L)} = l^{3/2}|v_y|_{L^2(0,2\pi)} = l^{3/2}|v|_1, \tag{4.8}$$

where $l = 2\pi/L$.

Several elements of the global attractor are shown in Figure 2 at $\vartheta = L^*/\pi^2 = 32$. Plotted here are most of the two-dimensional unstable manifold of the origin along with the one-dimensional unstable manifold of the "mixed-mode" steady state, labeled s_2, and a stable limit cycle (in bold). The axes are $\sin(x)$, $\sin(2x)$, and $\sin(3x)$, with the two-dimensional unstable manifold of the origin being tangent to the first two axes. The spheres labeled s_1 and s_3 represent bimodal (as they are along the $\sin(2x)$ axis) node and saddle steady states, respectively. There is a symmetric mixed-mode steady state which is hidden by the two-dimensional unstable manifold of the origin. We do not plot the one-dimensional unstable manifold of this second mixed-mode steady state, nor the two-dimensional unstable manifold of s_3, to avoid complicating

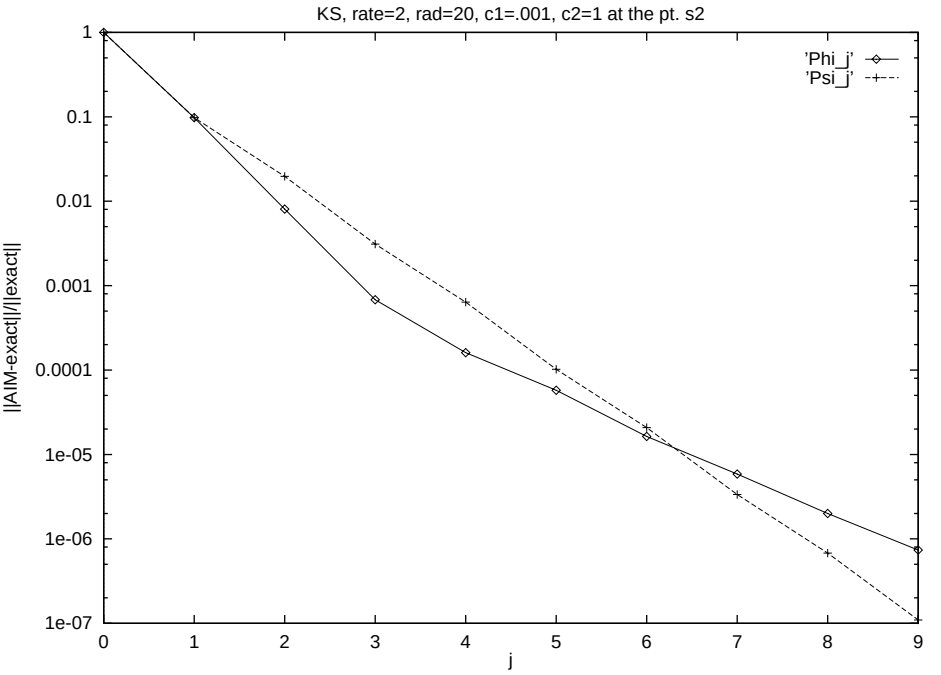


FIG. 5. *Relative error for Φ_j, Ψ_j at the steady state s_2.*

the picture. We can report, however, that the latter seems to have as its boundary s_3 and the limit cycle. Its shape is bowl-like, as indicated by the portion of the one-dimensional manifold of s_2 that approaches the limit cycle. Further extension of the two-dimensional unstable manifold of the origin suggests that it has as its boundary the unstable manifolds of mixed-mode steady states together with all of the steady states.

The convergence results for the sequences of AIMs given by Φ_j, Ψ_j, taken at the points p_1 (which is on the two-dimensional unstable manifold of the origin, not the one-dimensional manifold of s_2), p_2 (which is on the one-dimensional manifold of s_2), and p_3 (which is on the limit cycle) are shown in Figure 1. Note that the plots for Ψ_j all level off starting at small values of j. Indeed, it has been rigorously shown that in some sense the error in Ψ_2 is comparable to that of the limiting function $\Psi_\infty \equiv \lim_{j\to\infty} \Psi_j$. On the other hand the error for the sequence Φ_j continues to improve through $j = 9$ (and should do so beyond, until round-off errors produce diminished returns). At a steady state, however, one expects the two sequences to perform comparably, since graph Ψ_∞ contains all of the stationary solutions. This is confirmed by the results at s_2 shown in Figure 5.

To completely justify the computations we need to satisfy the gap condition and thus consider a manifold of higher dimension. All of the elements of the global attractor described above lie within a ball given by $|v|_1 \le 15$ in the phase space for (4.7), which by (4.8) corresponds to $|u|_1 \le 15 \cdot 2^{3/2} \cdot 32^{-3/4} \approx 3.2$. We see from Figure 4 that for the KSE prepared at $\rho_1 = 3.2$, rigorous analysis provides an inertial manifold of dimension five, and the convergence estimates in section 4 hold for a manifold of dimension 25.

With this in mind, we also consider a 64-mode Galerkin truncation of the KSE.

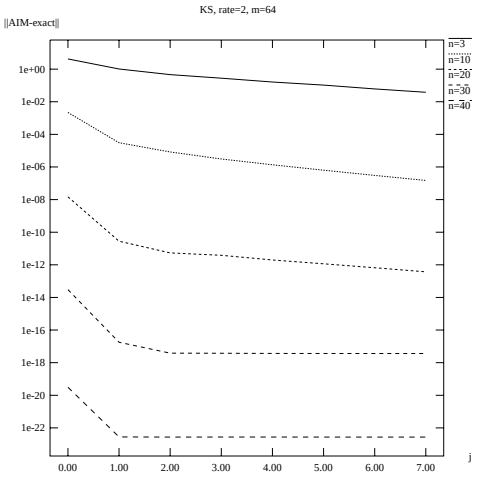


FIG. 6. *Gradient error from "exact" high-mode component of the point p3.*

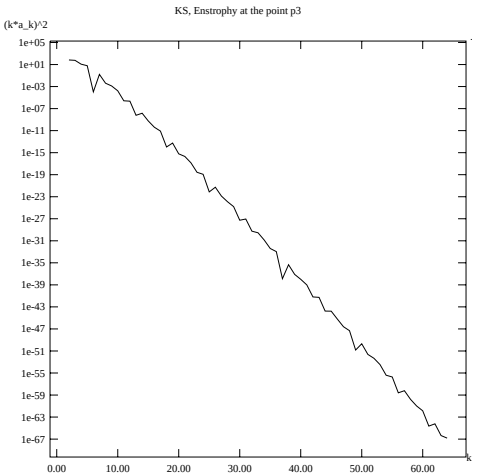


FIG. 7. *Spectrum of the H_1 norm of the point p3.*

We plot in Figure 6 the error of Φ_j from the "exact" Q_n-component at various values of n for the analogue of point p_3 for this finer discretization. The two observations to make are that the errors decrease dramatically as n increases, but that saturation sets in quickly as j increases in the cases where $n = 30$ and $n = 40$. One explanation for this is that the point p_3 is not exactly on the limit cycle. It was computed so that the image of the Poincaré return map (to the plane defined by the first mode being zero) differed from p_3 in the 12th decimal place in the $|\cdot|_1$ norm. The Q_n-component itself decays rapidly in this norm, as expected from the Gevrey regularity of the solution and as indicated in the plot of the spectrum of the H_1 norm in Figure 7. For these reasons we also plot the $|\cdot|_1$ norm of the residual between successive iterations in Figure 8. The fact that the residual continues to decay through seven iterations suggests that the saturation in Figure 6 was indeed due to the inaccuracy of the point p_3 rather than round-off error. The fact that the residual decays faster

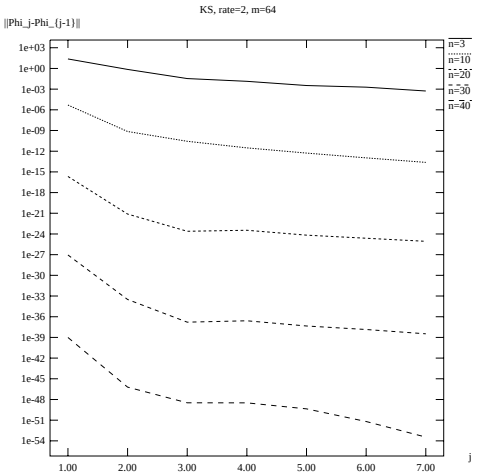


FIG. 8. *Gradient norm of residue.*

with increasing n is consistent with the convergence theory of the algorithm.

These computations have been carried out to compute invariant (perhaps inertial) manifolds for Galerkin approximations of the KSE. In the next section we analyze how these manifolds are connected to those for the PDE case.

5. Effect of truncating the high modes. In this section we examine the convergence of graph Φ_j to the exact inertial manifold as the wave number for truncation of the Q-modes is increased. Such a truncation is necessary for the PDE case considered here as the Q-mode is otherwise infinite-dimensional. It seems reasonable to expect that the convergence to the exact inertial manifold still takes place provided the threshold for this truncation tends to infinity along with either the dimension or the number of iterations. We outline how to show that this is indeed the case and quantify the error obtained with this further truncation even when the threshold for this truncation is kept bounded. The proofs for the estimates below are similar to those in [47] and can be found in [36].

5.1. Error estimates. For the analysis in this section, in addition to the assumptions made in section 2, we also require the following assumption.

A8. For $m > n$ we have $\Lambda_m > \Lambda_n$, $P_m P_n = P_n$, and, as a consequence, $Q_m Q_n = Q_m$.

The difference in the construction of the AIMs is that now we iterate the map $P_m \mathcal{T}_\tau^N$ instead of $\mathcal{T}_\tau^N$. The expression for the map $P_m \mathcal{T}_\tau^N$ is that for $\mathcal{T}_\tau^N$ in (3.1) with Q replaced by $P_m Q_n$, the other terms being unchanged. The same is true for the expressions (3.6) and (3.7) with respect to the "approximating trajectories" $\tilde{\varphi}_j$ obtained in this construction. More precisely, the AIMs are obtained in the following way. We first choose sequences $\{\tau_j\}_{j\in\mathbb{N}}$, $\tau_j > 0$, $\{N_j\}_{j\in\mathbb{N}}$, $N_j \in \mathbb{N}$, $N_j \geq 0$, and also $\{m_j\}_{j\in\mathbb{N}}$, $m_j > n$. Then we set

$$\tilde{\varphi}^0(p_0)(t) = p_0 \qquad \forall t \leq 0, \quad \forall p_0 \in P_n E,$$

and we proceed by Picard iteration:

$$\tilde{\varphi}^j(p_0) = P_{m_j} \mathcal{T}_{\tau_j}^{N_j}(\tilde{\varphi}^{j-1}(p_0), p_0) \qquad \forall p_0 \in P_n E.$$

Finally, we define the AIMs as the graphs

$$\tilde{\mathcal{M}}_j = \text{graph } \tilde{\Phi}_j = \{y + \tilde{\Phi}_j(y);\ y \in P_n E\}$$

of the maps $\tilde{\Phi}_j : P_n E \to P_{m_j} Q_n E$ defined by

$$\tilde{\Phi}_j(p_0) = P_{m_j} Q_n \tilde{\varphi}^j(p_0)(0) \qquad \forall p_0 \in P_n E.$$

Since the SGC (2.4) is satisfied, an inertial manifold $\mathcal{M} = \text{graph } \Phi$ of a function $\Phi : P_n E \to Q_n E$ exists as described in section 2, with $\Phi(p_0) = Q_n \varphi(p_0)$, where $\varphi(p_0)$ denotes the exact backward solution of (1.1) that passes through $p_0 + \Phi(p_0) \in \mathcal{M}$ at time $t = 0$, which is obtained as the fixed point of the map $\mathcal{T}(\cdot, p_0)$. For each $j \in \mathbb{N}$, one can show that the set $\tilde{\mathcal{M}}_j$ is a finite-dimensional topological submanifold of E.

For the convergence of the AIMs, one can proceed similarly to the proof in [47]. The first step concerns only the exact solution; hence we borrow it unmodified from [47]; it is Lemma 3.2 appearing in section 3 above. The second step is to estimate how the map $P_m \mathcal{T}_\tau^N$ brings elements in $\tilde{\mathcal{F}}$ closer to the exact orbits on the inertial manifold.

LEMMA 5.1. *Let N be a nonnegative integer, $m \in \mathbb{N}$ with $m > n$, and $\tau > 0$, and assume σ and σ_0 satisfy* (2.10) *with $\sigma_0 < \sigma$. Then for all $p_0 \in P_n E$ and all $\psi \in \tilde{\mathcal{F}}$, we have*

$$\begin{aligned} &\|\varphi(p_0) - P_m \mathcal{T}_\tau^N(\psi, p_0)\|_\sigma \le \kappa_{n,\sigma} \|\varphi(p_0) - \psi\|_\sigma \\ &+ \left(\tau \beta_{n,\sigma} + \frac{\beta_{n,\sigma_0}}{\sigma - \sigma_0} e^{-(\sigma - \sigma_0) N \tau} + \tilde{\beta}_{n,\sigma} \frac{\Lambda_m^\alpha}{\Lambda_m - \sigma} \right) (1 + |p_0|_E), \end{aligned} \tag{5.1}$$

where $\beta_{n,\sigma}$ and β_{n,σ_0} are as in Lemma 3.2*, and*

$$\tilde{\beta}_{n,\sigma} = K_2 (1 + \gamma_\alpha) \left\{ M_0 + \frac{M_1}{1 - \kappa_{n,\sigma}} \left[\frac{M_0 K_1}{\lambda_n^{1-\alpha}} + (1 + \gamma_\alpha) \frac{M_0 K_2}{\Lambda_n^{1-\alpha}} + K_1 \right] \right\}. \tag{5.2}$$

We may now estimate the distance from each "approximating trajectory" $\tilde{\varphi}^j(p_0)$ to the exact trajectory $\varphi(p_0)$ by iterating the estimate in Lemma 5.1.

LEMMA 5.2. *Let σ and σ_0 satisfy* (2.10) *with $\sigma_0 < \sigma$. Then*

$$\begin{aligned} &\sup_{p_0 \in P_n E} \frac{\|\varphi(p_0) - \tilde{\varphi}^j(p_0)\|_\sigma}{1 + |p_0|_E} \\ &\le \kappa_{n,\sigma}^j \left(K_1 + \frac{1}{1 - \kappa_{n,\sigma}} \left[\frac{M_0 K_1}{\lambda_n^{1-\alpha}} + (1 + \gamma_\alpha) \frac{M_0 K_2}{\Lambda_n^{1-\alpha}} + K_1 \right] \right) \\ &+ \sum_{\ell=0}^{j-1} \kappa_{n,\sigma}^\ell \left(\tau_{j-\ell} \beta_{n,\sigma} + \frac{\beta_{n,\sigma_0}}{\sigma - \sigma_0} e^{-(\sigma - \sigma_0) N_{j-\ell} \tau_{j-\ell}} + \tilde{\beta}_{n,\sigma} \frac{\Lambda_{m_{j-\ell}}^\alpha}{\Lambda_{m_{j-\ell}} - \sigma} \right), \end{aligned} \tag{5.3}$$

where $\beta_{n,\sigma}$ and β_{n,σ_0} are as in Lemma 3.2*, and $\tilde{\beta}_{n,\sigma}$ is as in Lemma* 5.1.

For the exponential convergence of the manifolds the sequences $\{\tau_j\}_j$, $\{N_j\}_j$, and $\{m_j\}_j$ have to be chosen appropriately.

THEOREM 5.3. *Assume* A1–A8 *hold. Assume also that it is possible to choose a sequence $\{m_j\}_j$ with $m_j > n$ and such that*

$$\Lambda_{m_j}^{1-\alpha} \ge \frac{c_0}{\delta^j} \qquad \forall j \in \mathbb{N}, \tag{5.4}$$

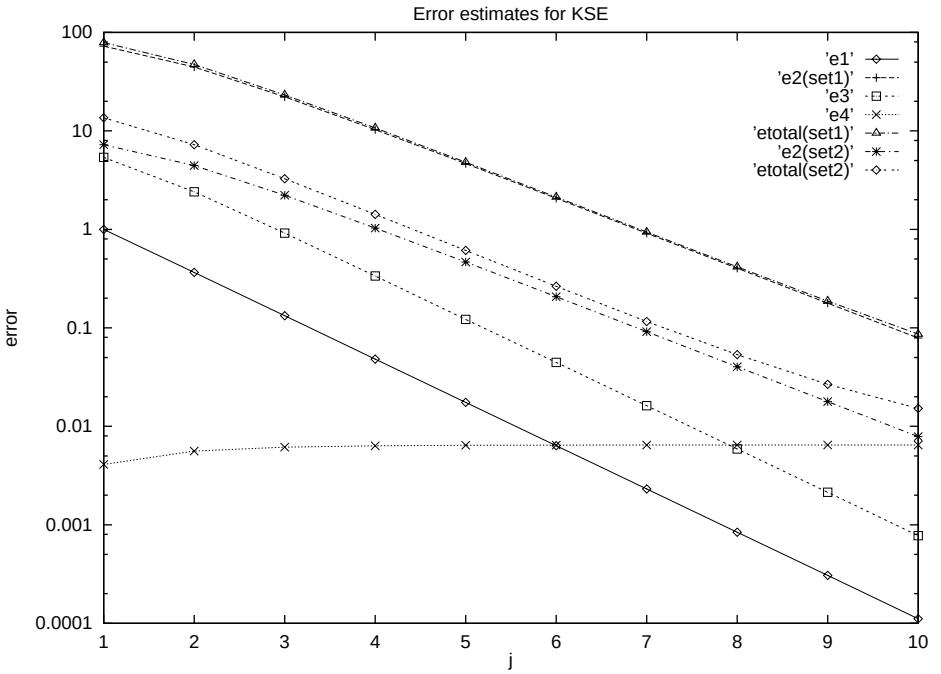


FIG. 9. *Total error estimate and individual terms in* (5.3) *for set*1*:* $c_1 = .001$*,* $c_2/c_1 = 1$*, and set*2*:*$c_1 = .0001$*,* $c_2/c_1 = 10$*.*

for some $c_1 > 0$ *and some* $0 < \delta < 1$*. Suppose that the sequences* $\{\tau_j\}_j$ *and* $\{N_j\}_j$ *are chosen so that*

$$0 < \tau_j \le c_1 \gamma^j \qquad \text{and} \qquad N_j \tau_j \ge c_2 j \quad \forall j \in \mathbb{N}, \tag{5.5}$$

for some $c_1, c_2 > 0$ *and some* $0 < \gamma < 1$*. Let now* σ *and* σ_0 *satisfy* (2.10) *with* $\sigma < \sigma_0$*. Let*

$$\epsilon = \min\{\kappa_{n,\sigma}, \max\{\gamma, e^{-c_2(\sigma-\sigma_0)}, \delta\}\},$$

which is less than 1*. Then, for any* η *with* $\epsilon < \eta < 1$*, there exists a* $c_\eta \ge 0$ *such that*

$$d(\Phi, \tilde{\Phi}_j) \le c_\eta \eta^j \qquad \forall j \in \mathbb{N}. \tag{5.6}$$

5.2. Application to the KSE. We saw in 4.1 that the KSE prepared at a radius $r = 3.2$ in $|\cdot|_1$; A1–A7 hold for $n \ge 25$. The condition A8 also holds, trivially. Here we fix $n = 40$ and observe that κ is a decreasing function of σ over the interval in (2.10). Thus we select σ and σ_0 to be the right and left endpoints (minus .001 and plus .001, respectively) in the calculation of $\beta_{n,\sigma}$, β_{n,σ_0} from (3.12), and $\tilde{\beta}_{n,\sigma}$ from (5.2). We consider a fixed truncation $m_j = m = 64$ and plot in Figure 9 the total error as estimated in (5.3) along with the individual terms denoted e1 through e4, consecutively from left to right (the last three are in the summation). The plots are for two algorithm settings, set1: $c_1 = .001$ $c_2/c_1 = 1$ (as in the previous runs for the KSE) and set2: $c_1 = .0001$ $c_2/c_1 = 10$. For the first setting, the term e2 dominates

by about a factor of ten suggesting that c_1 should be decreased by just this amount. By simultaneously increasing c_2/c_1 by the same factor in set2, we leave the terms e1, e3, and e4 unchanged and arrive at a total error which is fairly evenly balanced in all the terms. Note that e4 actually increases with the number of iterations since m_j is fixed in this case.

REFERENCES

[1] D. Armbruster, J. Guckenheimer, and P. Holmes, *Kuramoto–Sivashinsky dynamics on the center-unstable manifold*, SIAM J. Appl. Math., 49 (1989), pp. 676–691.

[2] H. W. Broer, H. M. Osinga, and G. Vegter, *Algorithms for computing normally hyperbolic invariant manifolds*, Z. Angew. Math. Phys., 48 (1997), pp. 480–524.

[3] J. Carr, *Applications of Centre Manifold Theory*, Appl. Math. Sci. 35, Springer-Verlag, New York, 1981.

[4] H.-C. Chang, *Traveling waves on fluid interfaces: Normal form analysis of the Kuramoto–Sivashinsky equation*, Phys. Fluids, 29 (1986), pp. 3142–3147.

[5] S.-N. Chow and K. Lu, *Invariant manifolds for flows in Banach spaces*, J. Differential Equations, 74 (1988), pp. 285–317.

[6] P. Collet, J.-P. Eckmann, H. Epstein, and J. Stubbe, *A global attracting set for the Kuramoto–Sivashinsky equation*, Comm. Math. Phys., 152 (1993), pp. 203–214.

[7] P. Constantin, C. Foiaş, B. Nicolaenko, and R. Temam, *Integral Manifolds and Inertial Manifolds for Dissipative Partial Differential Equations*, Appl. Math. Sci. 70, Springer-Verlag, New York, 1989.

[8] S. P. Dawson and A. M. Mancho, *Collections of heteroclinic cycles in the Kuramoto–Sivashinsky equation*, Phys. D, 100 (1997), pp. 231–256.

[9] A. Debussche and M. Marion, *On the construction of families of approximate inertial manifolds*, J. Differential Equations, 100 (1992), pp. 173–201.

[10] A. Debussche and R. Temam, *Convergent families of approximate inertial manifolds*, J. Math. Pures Appl. (9), 73 (1994), pp. 489–522.

[11] F. Demengel and J. M. Ghidaglia, *Inertial manifolds for partial differential evolution equations under time-discretization: Existence, convergence, and applications*, J. Math. Anal. Appl., 155 (1991), pp. 177–225.

[12] L. Dieci and J. Lorenz, *Computation of invariant tori by the method of characteristics*, SIAM J. Numer. Anal., 32 (1995), pp. 1436–1474.

[13] T. Dubois, F. Jauberteau, and R. Temam, *Solution of the incompressible Navier–Stokes equations by the nonlinear Galerkin method*, J. Sci. Comput., 8 (1993), pp. 167–194.

[14] C. Foiaş, O. Manley, and R. Temam, *Modelling of the interaction of small and large eddies in two-dimensional turbulent flows*, RAIRO Modél. Math. Anal. Numér., 22 (1988), pp. 93–118.

[15] C. Foiaş, B. Nicolaenko, G. R. Sell, and R. Temam, *Inertial manifolds for the Kuramoto–Sivashinsky equation and an estimate of their lowest dimension*, J. Math. Pures Appl. (9), 67 (1988), pp. 197–226.

[16] C. Foiaş and J.-C. Saut, *Remarques sur les équations de Navier–Stokes stationnaires*, Ann. Scuola Norm. Sup. Pisa Cl. Sci. (4), 10 (1983), pp. 169–177.

[17] C. Foiaş, G. R. Sell, and R. Temam, *Inertial manifolds for nonlinear evolutionary equations*, J. Differential Equations, 73 (1988), pp. 309–353.

[18] C. Foiaş, G. R. Sell, and E. S. Titi, *Exponential tracking and approximation of inertial manifolds for dissipative nonlinear equations*, J. Dynam. Differential Equations, 1 (1989), pp. 199–244.

[19] C. Foiaş and R. Temam, *Remarques sur les équations de Navier–Stokes stationnaires et les phénomènes successifs de bifurcation*, Ann. Scuola Norm. Sup. Pisa Cl. Sci. (4), 5 (1978), pp. 28–63.

[20] B. García-Archilla and J. de Frutos, *Time integration of the non-linear Galerkin method*, IMA J. Numer. Anal., 15 (1995), pp. 221–244.

[21] B. García-Archilla, J. Novo, and E. S. Titi, *Postprocessing the Galerkin method: A novel approach to approximate inertial manifolds*, SIAM J. Numer. Anal., 35 (1998), pp. 941–972.

[22] J. Goodman, *Stability of Kuramoto–Sivashinsky and related systems*, Comm. Pure Appl. Math., 47 (1994), pp. 293–306.

[23] M. Graham, P. Steen, and E. S. Titi, *Computational efficiency and approximate inertial manifolds for a Bénard convection system*, J. Nonlinear Sci., 3 (1993), pp. 153–167.

[24] J. Guckenheimer and P. Worfolk, *Dynamical systems: Some computational problems*, in Bifurcations and Periodic Orbits of Vector Fields (Montreal, 1992), Vol. 408 of NATO Adv. Sci. Inst. Ser. C Math. Phys. Sci. 408, Kluwer, Dordrecht, The Netherlands, 1993, pp. 241–277.

[25] J. K. Hale, *Asymptotic Behavior of Dissipative Systems*, Math. Surveys Monogr. 25, AMS, Providence, RI, 1988.

[26] J. K. Hale and C. Perelló, *The neighborhood of a singular point of functional differential equations*, Contributions to Differential Equations, 3 (1964), pp. 351–375.

[27] D. Henry, *Geometric Theory of Semilinear Parabolic Equations*, Lecture Notes in Math. 840, Springer-Verlag, Berlin, 1981.

[28] J. M. Hyman, B. Nicolaenko, and S. Zaleski, *Order and complexity in the Kuramoto–Sivashinsky model of weakly turbulent interfaces*, Phys. D, 23 (1986), pp. 265–292.

[29] Ju. S. Ilyashenko, *Global analysis of the phase portrait for the Kuramoto–Sivashinsky equation*, J. Dynam. Differential Equations, 4 (1992), pp. 585–615.

[30] V. I. Istratescu, *Fixed Point Theory. An Introduction. With a Preface by Michiel Hazewinkel*, D. Reidel, Dordrecht, The Netherlands, 1981.

[31] M. E. Johnson, M. S. Jolly, and I. G. Kevrekidis, *Two-dimensional invariant manifolds and global bifurcations: Some approximation and visualization studies*, Numer. Algorithms, 14 (1997), pp. 125–140.

[32] M. E. Johnson, M. S. Jolly, and I. G. Kevrekidis, *The Oseberg transition: Visualization of global bifurcations for the Kuramoto–Sivashinsky equation*, Int. J. Bifur. Chaos Appl. Sci. Engrg., 11 (2001), pp. 1–18.

[33] M. S. Jolly, I. G. Kevrekidis, and E. S. Titi, *Approximate inertial manifolds for the Kuramoto–Sivashinsky equation: Analysis and computations*, Phys. D, 44 (1990), pp. 38–60.

[34] M. S. Jolly, I. G. Kevrekidis, and E. S. Titi, *Preserving dissipation in approximate inertial forms for the Kuramoto–Sivashinsky equation*, J. Dynam. Differential Equations, 3 (1991), pp. 179–197.

[35] M. S. Jolly and R. Rosa, *Computations on Center Manifolds*, in preparation, 1998.

[36] M. S. Jolly, R. Rosa, and R. Temam, *Accurate Computations on Inertial Manifolds*, IMA preprint 1602, IMA, University of Minnesota, Minneapolis, MN, 1999.

[37] M. S. Jolly, R. Rosa, and R. Temam, *Evaluating the dimension of an inertial manifold for the Kuramoto–Sivashinsky equation*, Adv. Differential Equations, 5 (2000), pp. 31–66.

[38] I. G. Kevrekidis, B. Nicolaenko, and J. C. Scovel, *Back in the saddle again: A computer assisted study of the Kuramoto–Sivashinsky equation*, SIAM J. Appl. Math., 50 (1990), pp. 760–790.

[39] X. Liu, *Gevrey class regularity and approximate inertial manifolds for the Kuramoto–Sivashinsky equation*, Phys. D, 50 (1991), pp. 135–151.

[40] R. Mañé, *Reduction of semilinear parabolic equations to finite dimensional C^1 flows*, in Geometry and Topology (Proc. III Latin Amer. School of Math., Inst. Mat. Pura Aplicada CNPq, Rio de Janeiro, 1976), Lecture Notes in Math. 597, Springer-Verlag, Berlin, 1977, pp. 361–378.

[41] J. Mallet-Paret and G. R. Sell, *Inertial manifolds for reaction diffusion equations in higher space dimensions*, J. Amer. Math. Soc., 1 (1988), pp. 805–866.

[42] D. Michelson, *Stability of the Bunsen flame profiles in the Kuramoto–Sivashinsky equation*, SIAM J. Math. Anal., 27 (1996), pp. 765–781.

[43] M. Miklavčič, *A sharp condition for existence of an inertial manifold*, J. Dynam. Differential Equations, 3 (1991), pp. 437–456.

[44] B. Nicolaenko, B. Scheurer, and R. Temam, *Some global dynamical properties of the Kuramoto–Sivashinsky equations: Nonlinear stability and attractors*, Phys. D, 16 (1985), pp. 155–183.

[45] F. Pinto, *Nonlinear stability and dynamical properties for a Kuramoto–Sivashinsky equation in space dimension two*, Discrete Contin. Dynam. Systems, 5 (1999), pp. 117–136.

[46] J. Robinson, *Computing Inertial Manifolds*, manuscript.

[47] R. Rosa, *Approximate inertial manifolds of exponential order*, Discrete Contin. Dynam. Systems, 1 (1995), pp. 421–448.

[48] R. Rosa and R. Temam, *Inertial manifolds and normal hyperbolicity*, Acta Appl. Math., 45 (1996), pp. 1–50.

[49] R. D. Russell, D. M. Sloan, and M. R. Trummer, *Some numerical aspects of computing inertial manifolds*, SIAM J. Sci. Comput., 14 (1993), pp. 19–43.

[50] R. Temam, *Induced trajectories and approximate inertial manifolds*, RAIRO Modél. Math. Anal. Numér., 23 (1989), pp. 541–561.

[51] R. Temam, *Infinite-Dimensional Dynamical Systems in Mechanics and Physics*, 2nd ed., Appl. Math. Sci. 68, Springer-Verlag, New York, 1997.

[52] R. Temam and X. Wang, *Estimates on the lowest dimension of inertial manifolds for the Kuramoto–Sivashinsky equation in the general case*, Differential Integral Equations, 7 (1994), pp. 1095–1108.

[53] E. S. Titi, *On approximate inertial manifolds to the Navier–Stokes equations*, J. Math. Anal. Appl., 149 (1990), pp. 540–557.

SIAM J. SCI. COMPUT.
Vol. 22, No. 6, pp. 2239–2255
© 2001 Society for Industrial and Applied Mathematics

INITIAL VALUES FOR THE INVERSE TOEPLITZ EIGENVALUE PROBLEM*

DIRK LAURIE†

Abstract. We show how to find a symmetric Toeplitz matrix whose eigenvalues are close to a specified target set of n real numbers.

Key words. inverse eigenvalue problem, Toeplitz, initial values

AMS subject classifications. 65F18, 47B35

PII. S106482759935561X

1. Introduction.

A Toeplitz matrix is constant along all diagonals parallel to the main diagonal. A symmetric Toeplitz matrix

$$M_n(\mathbf{t}) = \begin{bmatrix} t_1 & t_2 & \dots & t_{n-1} & t_n \\ t_2 & t_1 & t_2 & \dots & t_{n-1} \\ \ddots & \ddots & \ddots & \ddots & \ddots \\ t_{n-1} & \dots & t_2 & t_1 & t_2 \\ t_n & t_{n-1} & \dots & t_2 & t_1 \end{bmatrix}$$

is therefore fully determined by its first row $\mathbf{t} = (t_1, t_2, \dots, t_n)$. In this paper "Toeplitz matrix" always means "real symmetric Toeplitz matrix."

The inverse Toeplitz eigenvalue problem is to find such a matrix whose eigenvalues are a given nondecreasing n-tuple

$$\boldsymbol{\lambda} = (\lambda_1, \lambda_2, \dots, \lambda_n), \quad \text{where} \quad \lambda_1 \le \lambda_2 \le \cdots \le \lambda_n,$$

of real numbers. The first proof of the existence of a solution to the inverse Toeplitz eigenvalue problem was given by Landau [8].

Since the proof is nonconstructive, the actual computation of such a solution still involves numerical methods. Several iterative methods have been suggested. In [9, 10, 3, 12], the iteration involves constructing a sequence of symmetric Toeplitz matrices whose eigenvalues are intended to approach the designated target. These methods start from an initial Toeplitz matrix.

In [4, 13], on the other hand, the iteration involves constructing a sequence of isospectral matrices (i.e., all having the same eigenvalues) which can only become stationary when a Toeplitz matrix is reached. These methods start from an initial matrix isospectral to the target. Such a matrix can be obtained from an initial Toeplitz matrix by finding its eigenvector matrix Q and calculating $Q\,\mathrm{diag}(\lambda_k)Q^{\mathsf{T}}$, provided that one can find a good way to associate eigenvectors with target eigenvalues. We shall say more about this question later.

The construction of good initial values, using some ideas from Landau's proof, is the topic of this paper. The method involves identifying n Toeplitz matrices

*Received by the editors May 7, 1999; accepted for publication (in revised form) February 11, 2000; published electronically April 26, 2001.
http://www.siam.org/journals/sisc/22-6/35561.html

†Potchefstroom University for Christian Higher Education, P.O. Box 1174, Vanderbijlpark 1900, South Africa (dlaurie@na-net.ornl.gov). Current address: Department of Mathematics, Stellenbosch University, Private Bag X1, Matieland 7602, South Africa.

T_k, $k = 1, 2, \ldots, n$, with known eigenvalues, expressing the target eigenvalues as a linear combination of the eigenvalues of the matrices T_k and using as initial Toeplitz matrix the corresponding linear combination of the matrices T_k. It can therefore be viewed as linear interpolation in the space of Toeplitz matrices.

The structure of the paper is as follows: first we give an outline of Landau's proof in order to introduce the main ideas of the interpolation method; then we describe the method itself, its efficient implementation, and an interpretation of the method in terms of successive lower-dimensional interpolations; and finally we give the theoretical basis for the method in formal form. However, for those who are interested in results, not theory, here is a Matlab routine implementing the algorithm:

```
function [t,M,Q]=toepinit(lam)
% TOEPINIT  T=TOEPINIT(LAM) returns the first row of a symmetric Toeplitz
%           matrix T with eigenvalues close to the supplied eigenvalues LAM.
%           [T,M]=TOEPINIT(LAM) returns also a matrix M with the specified
%           eigenvalues, where M is close to Toeplitz.
%           [T,M,Q]=TOEPINIT(LAM) returns also the eigenvector matrix of M.

%  Dirk Laurie  1996-07-31

% Solve the equations
%   db(1)=lam(1), k=2:n, db(k-1)+db(k)=lam(k)
n=length(lam); lam=sort(lam(:))';
db=[0 filter(1,[1 1],lam)];

% C=toeplitz(c) is a circulant, S=toeplitz(s) is a skew-circulant
% T=toeplitz(t) is the required Toeplitz matrix
kc=[n+1:-2:1 rem(n+1,2)+2:2:n]; c=ifft(db(kc));
w=exp(pi*i*(0:n-1)/n);
ks=[n:-2:1 rem(n,2)+2:2:n]; s=w.*ifft(db(ks));
t=real(c+s); if nargout<2, return; end

[Q,D]=eig(toeplitz(t)); [d,k]=sort(diag(D));
Q=Q(:,k);
M=Q*diag(lam)*Q';
```

2. An outline of Landau's proof. The method suggested in this paper was inspired by ideas in Landau's proof. Although the proof itself is nonconstructive in the sense that it does not suggest an algorithm by which the inverse Toeplitz eigenproblem may be solved, it contains powerful and suggestive geometrical ideas which contribute much to the intuitive understanding of the problem. It is therefore not out of place, even in a numerical analysis journal, to devote some space to a qualitative summary of the method of proof.

We say that two Toeplitz matrices A and B are *equivalent* if each can be obtained from the other by an affine transformation with nonzero scale factor, i.e., $A = c_0 I + c_1 B$ with $c_1 \neq 0$. Similarly two nondecreasing n-tuples are equivalent if each can be obtained from the other by such an affine transformation. Equivalent matrices have all their eigenvectors in common, and their eigenvalues are equivalent under the same affine transformation under which the matrices are equivalent.

If a solution

$$M_n(t_1, t_2, \ldots, t_n)$$

to the inverse Toeplitz eigenproblem exists, then

$$M_n(t_1, -t_2, t_3, -t_4, \ldots, (-1)^{n-1}t_n)$$

is also a solution. There is therefore no loss of generality in looking only for solutions such that $t_2 \geq 0$.

A Toeplitz matrix is centrosymmetric, and therefore it is possible to assign parity (the property of being even or odd) to its eigenvalues as follows. For a vector $\mathbf{v} = (v_1, v_2, \ldots, v_k)$, the vector $\mathbf{v}_{\text{rev}}$ is obtained by writing the elements in reverse order, i.e., $\mathbf{v}_{\text{rev}} = (v_k, v_{k-1}, \ldots, v_1)$. A vector $\mathbf{v}$ is called *even* if $\mathbf{v} = \mathbf{v}_{\text{rev}}$ and odd if $\mathbf{v} = -\mathbf{v}_{\text{rev}}$. It is well known [2] that all eigenvectors of a symmetric centrosymmetric matrix with distinct eigenvalues are either even or odd: the parity of an eigenvalue is then that of its eigenvector. In the case of multiple eigenvalues, it is always possible to find a basis consisting only of even and odd eigenvectors.

Delsarte and Genin [6] showed that any solution to the inverse Toeplitz eigenvalue problem which is a continuous function of the eigenvalues must have the property that when the eigenvalues are in increasing order, adjacent eigenvalues are of opposite parity. This property is central to Landau's proof.

We now come to the proof itself.

- The Toeplitz matrices under consideration are restricted to *regular* matrices. A regular Toeplitz matrix has only simple eigenvalues, its largest eigenvalue is even, and adjacent eigenvalues λ_k, λ_{k+1}, $k = 1, 2, \ldots, n-1$, are of opposite parity; also, all leading submatrices of a regular matrix have these properties. Note that regular matrices must have $t_2 > 0$.
- Two degrees of freedom are eliminated by taking only one standard representative from each equivalence class of regular Toeplitz matrices, and from each equivalence class of eigenvalue n-tuples. This allows both sets of equivalence classes to be parametrized by $n-2$ real numbers.
  We denote by $\mathcal{T}_n$ the set of standard regular Toeplitz matrices, by $\mathcal{L}_n$ the set of standard eigenvalue n-tuples, and by $\widehat{\mathcal{T}}$ and $\widehat{\mathcal{L}}$, respectively, their representations in R^{n-2}. To avoid unnecessary circumlocution, we shall refer to a point in $\widehat{\mathcal{T}}$ as a "matrix" and impute properties of a matrix to it, when strictly speaking we mean the matrix associated with that point.
  The function Λ that maps a Toeplitz matrix to its sorted eigenvalues induces a map $\widehat{\Lambda}$ from $\widehat{\mathcal{T}}$ to $\widehat{\mathcal{L}}$ by mapping a standard matrix T to the standard eigenvalue vector equivalent to $\Lambda(T)$.
- The closure of $\widehat{\mathcal{T}}$ in R^{n-2} is compact, and its boundary is made up entirely of matrices with multiple eigenvalues—in other words, matrices that map to points on the boundary of $\widehat{\mathcal{L}}$. A consequence of this fact is that in order to check whether the matrices along a continuous path starting at a point in $\widehat{\mathcal{T}}$ are regular, it is not necessary to check the regularity of submatrices: it is sufficient to check that each matrix in the path has only simple eigenvalues.
- The topological degree of the map $\widehat{\Lambda}$ from $\widehat{\mathcal{T}}$ to the interior of $\widehat{\mathcal{L}}$ is constant. It is therefore sufficient to exhibit one point in the interior of $\widehat{\mathcal{L}}$ that is covered exactly once: then every point in $\widehat{\mathcal{L}}$ must be covered an odd number of times—therefore, at least once.
- An obvious parametrization of $\mathcal{L}_n$ is to use the "middle" eigenvalues ($\lambda_2 < \lambda_3 < \cdots < \lambda_{n-1}$), so that $\widehat{\mathcal{L}}$ is a simplex in R^{n-2}. $\widehat{\mathcal{T}}$ also has some properties reminiscent of those of a simplex. The boundary of $\widehat{\mathcal{T}}$ is described in terms of *vertices*, which have only two distinct eigenvalues each; *k-dimensional edges*

for $k = 1, 2, \ldots, n-4$, which have $k+2$ distinct eigenvalues at each point; and *faces*, which have $n-1$ distinct eigenvalues. Landau proves that each of the $n-1$ faces with $\lambda_k = \lambda_{k+1}$ has a nonempty intersection with the boundary of $\widehat{\mathcal{T}}$. In intuitive terms, the set $\widehat{\mathcal{T}}$ is rather like a simplex with curved faces. Figures 1 and 2 show a two-dimensional representation of $\mathcal{T}_4$ and a projection of a view of a three-dimensional representation of $\mathcal{T}_5$.

- There exists a path starting at the vertex where the largest eigenvalue is simple (this matrix has rank 1 and is easily seen to be unique) and moving along k-dimensional edges for increasing k, until finally a point in the interior of a face is reached, such that the map Λ is one-to-one and its Jacobian nonzero everywhere along the path. By continuity, there must then be some point in the interior of $\widehat{\mathcal{L}}$ at which these properties still hold. Its image is covered exactly once, which completes the proof.

Landau's paper concludes with the following conjecture, followed by the statement: "Of course, it would also be very interesting to find an algorithm for carrying it out."

CONJECTURE 1 (Landau). *The map from $\mathcal{L}_n$ to $\mathcal{T}_n$ is one-to-one.*

3. Initial values by linear interpolation.

3.1. The basic idea. The precise way of standardizing the eigenvalues and Toeplitz matrices is not crucial to Landau's proof. Landau uses $t_0 = 0,\ t_1 = 1$ for the matrices and $\lambda_1 = -1,\ \sum \lambda_i = 0$ for the eigenvalues. The map $\widehat{\Lambda}$ is in that case well defined but does not in general map a matrix to its eigenvalues.

For our purposes the most convenient standardization in both cases maps the smallest eigenvalue to 0 and the largest to 1. Let $T_k,\ k = 1, 2, \ldots, n-1$, be the Toeplitz matrices that correspond to the vertices of $\widehat{\mathcal{T}}$. Their eigenvalue n-tuples $L_k = \Lambda(T_k)$, given by

$$\begin{aligned} L_1 &= (0, 0, \ldots, 0, 1), \\ L_2 &= (0, \ldots, 0, 1, 1), \\ &\cdots\cdots \\ L_{n-1} &= (0, 1, 1, \ldots, 1), \end{aligned}$$

correspond to vertices of $\widehat{\mathcal{L}}$. In section 4 we give an explicit formula (5) for the matrices T_k.

Since the topological structure of $\widehat{\mathcal{T}}$ resembles that of $\widehat{\mathcal{L}}$, it is reasonable to suppose that simple linear interpolation might provide a good approximation to $\Lambda^{-1}(\boldsymbol{\lambda})$.

1. Express $\boldsymbol{\lambda}$ as a linear combination of the vertices of $\mathcal{L}_n$:

$$\boldsymbol{\lambda} = \sum_{k=1}^{n-1} a_k L_k.$$

From the eigenvalue n-tuples for L_k it is obvious that

$$a_{n-k} = \lambda_{k+1} - \lambda_k,\ k = 1, \ldots, n-1.$$

Since $\lambda_1 = 0$ and $\lambda_n = 1$, the coefficients a_k are nonnegative and sum to 1.

2. Approximate $\Lambda^{-1}(\boldsymbol{\lambda})$ by

$$\Gamma(\boldsymbol{\lambda}) = \sum_{k=1}^{n-1} a_k T_k.$$

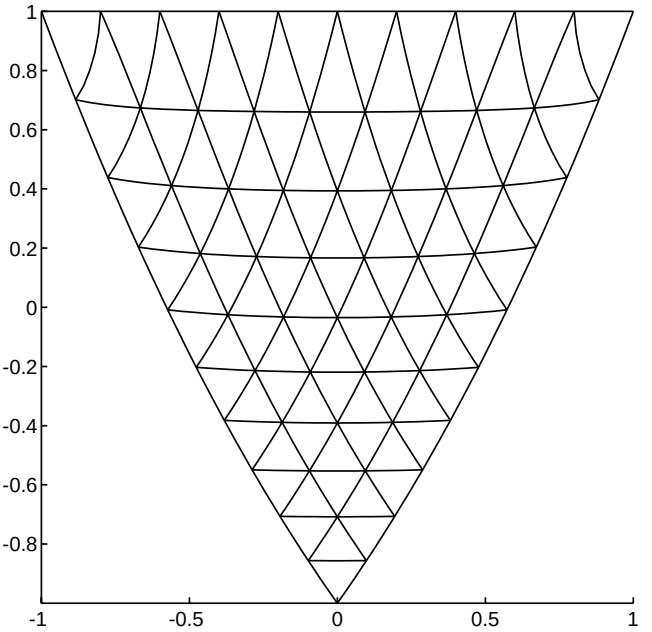


FIG. 1. *Toeplitz matrices with first row* $(0, 1, x, y)$ *are regular for* (x, y) *in the interior of the region shown. When the corresponding eigenvalues are affinely transformed to* $(0, \lambda, \mu, 1)$ *and displayed in the* (λ, μ) *plane, the lines form an equispaced triangular grid.*

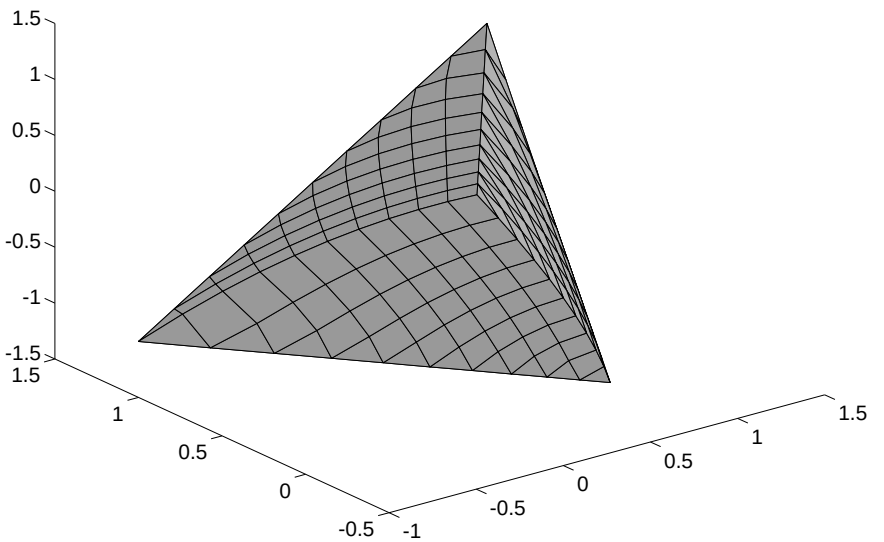


FIG. 2. *Toeplitz matrices with first row* $(0, 1, x, y, z)$ *are regular for* (x, y, z) *in the interior of the region shown. When the corresponding eigenvalues are affinely transformed to* $(0, \lambda, \mu, \nu, 1)$ *and displayed in the* (λ, μ, ν) *plane, the lines on each face form two of the three sets of lines that make up an equispaced triangular grid. The vertices* T_1, T_2, T_3, *and* T_4 *are, respectively, top, right, center, and left in the picture. The initial values are exact along the straight lines* T_1T_3 *and* T_2T4 *but not along the straight line* T_1T_4. *The lines* T_1T_2 *and* T_4T_3 *are actually congruent, although this view does not show the curvature of the former clearly.*

In general $\Gamma(\boldsymbol{\lambda})$ will not belong to $\mathcal{T}_n$ and may need to be standardized unless only the eigenvectors of the initial approximation are needed. (This is in fact true for most of the known iterative methods.)

This theoretical drawback could be avoided by using Landau's standardization ($t_1 = 0,\ t_2 = 1$) instead of ours: any convex combination of standardized matrices in that case is also standardized. This involves calculating

$$\sum_{k=1}^{n-1} \frac{a_k}{(T_k)_{1,2}} T_k$$

instead of $\Gamma(\boldsymbol{\lambda})$, which may not seem to be much less convenient. We shall see in the next section, however, that our standardization allows a fast way of calculating $\Gamma(\boldsymbol{\lambda})$ without explicitly finding the interpolation constants or even the matrices T_k; this expedient does not carry over to Landau's standardization.

The whole process of interpolation would be on shaky ground if it were possible to produce an initial matrix that is not regular. We therefore propose the following.

CONJECTURE 2. *Any linear combination with positive coefficients of regular Toeplitz matrices is itself a regular Toeplitz matrix.*

This conjecture is equivalent to saying that the set $\widehat{\mathcal{T}}$ with Landau's normalization $t_0 = 0, t_1 = 1$ is convex. The conjecture seems to be quite hard to prove but is certainly true for the Trench class of regular Toeplitz matrices [11]. These matrices have the integral representation

$$t_r = \frac{1}{\pi} \int_0^\pi f(\theta) \cos(r-1)\theta \, d\theta, \quad r = 1, \ldots, n, \tag{1}$$

where f is nonincreasing.

After transforming by an affine transformation the initial values thus found, one has in effect approximated the Toeplitz matrix having the original set of eigenvalues by a linear combination of $T_1, T_2, \ldots, T_n$, since $T_n = I$. All the coefficients in the linear combination except possibly the one involving T_n are of the same sign.

3.2. Implementation via circulant theory. A circulant is a Toeplitz matrix in which

$$t_{n-k+2} = t_k, \quad k = 2, 3, \ldots, n,$$

and a skew-circulant is a Toeplitz matrix in which

$$t_{n-k+2} = -t_k, \quad k = 2, 3, \ldots, n.$$

(Remember that we are dealing only with real symmetric Toeplitz matrices and therefore only with symmetric circulants and skew-circulants.) Any Toeplitz matrix $T = M_n(\mathbf{t})$ can be expressed as the sum of a circulant C and a skew-circulant S as follows:

$$C = M_n\left(t_1 - s_1, \tfrac{1}{2}(t_2 + t_n), \tfrac{1}{2}(t_3 + t_{n-1}), \ldots, \tfrac{1}{2}(t_n + t_2)\right),$$
$$S = M_n\left(s_1, \tfrac{1}{2}(t_2 - t_n), \tfrac{1}{2}(t_3 - t_{n-1}), \ldots, \tfrac{1}{2}(t_n - t_2)\right),$$

where s_1 is arbitrary.

It is a well-known fact [5] that any circulant is diagonalized by the Fourier matrix $F = [f_{jk}]$ with $f_{jk} = \omega^{(j-1)(k-1)}$, where $\omega = e^{-2\pi i/n}$: i.e., $F^{-1}CF = D = \operatorname{diag}(d_k)$.

We show in the proof of Lemma 2 that $d_k = d_{n-k+2}$ when k and $n-k+2$ are valid indices. In other words, all eigenvalues of C are double, except d_1 and (if $n = 2m$) d_{m+1}. The matrix C has at most $n_d = \lceil (n+1)/2 \rceil$ distinct eigenvalues. We say that a sequence $\mathbf{d} = (d_1, d_2, \ldots, d_n)$ such that $d_k = d_{n-k+2}$ is *circulant compatible.*

Similarly, any skew-circulant is diagonalized by the matrix WF with

$$W = \mathrm{diag}(1, \zeta, \zeta^2, \ldots, \zeta^{n-1}),$$

where $\zeta = e^{-\pi i/n}$: i.e., $(WF)^{-1} S W F = B = \mathrm{diag}(b_k)$. We show in the proof of Lemma 2 that $b_k = b_{n-k+1}$ when k and $n-k+1$ are valid indices. In other words, all eigenvalues of S are double, except (if $n = 2m-1$) b_m. The matrix S has at most $n_b = \lfloor (n+1)/2 \rfloor$ distinct eigenvalues. We say that a sequence $\mathbf{b} = (b_1, b_2, \ldots, b_n)$ such that $b_k = b_{n-k+1}$ is *skew-circulant compatible.*

Let $L = \mathrm{diag}(\boldsymbol{\lambda}_p)$, where $\boldsymbol{\lambda}_p$ is some permutation (to be specified later) of $\boldsymbol{\lambda}$. We wish to express L as $L = D + B$, where the diagonals of D and B are, respectively, circulant and skew-circulant compatible. The matrix equation $L = D + B$ amounts to a system of n linear equations for the $n+1$ unknowns $d_k,\ k = 1, 2, \ldots, n_d$, and $b_k,\ k = 1, 2, \ldots, n_b$. The extra degree of freedom could be used for making $\sum_{k=1}^{n} b_k = 0$, but in our Matlab code we found it simpler to make one of the unknowns zero.

Having found $C = FDF^{-1}$ and $S = WFB(WF)^{-1}$, we compute $C + S$ as an approximation to the regular Toeplitz matrix with eigenvalues $\boldsymbol{\lambda}$.

It remains to find the correct permutation of $\boldsymbol{\lambda}$. The vertices of L_n are all special cases of eigenvalue sequences that can be permuted to be circulant or skew-circulant compatible; certainly one would require that the corresponding vertices of $\mathcal{T}_n$ be found correctly. We show in Theorem 2 that this requirement is met when $d_k,\ k = 1, 2, \ldots, n_d$, and $b_k,\ k = 1, 2, \ldots, n_b$, are both arranged in nonincreasing order. We therefore have the equations

$$\begin{aligned} d_1 + b_1 &= \lambda_n, \\ d_2 + b_1 &= \lambda_{n-1}, \\ d_2 + b_2 &= \lambda_{n-2}, \\ &\cdots\cdots \end{aligned} \tag{2}$$

$$d_{n_d} + b_{n_b} = \lambda_1. \tag{3}$$

In the Matlab code, we put $d_{n_d} = 0$ if $n_d = n_b + 1$ (which happens when n is even) and $b_{n_b} = 0$ if $n_d = n_b$.

Since the procedure to find it is linear, and maps the vertices of $\mathcal{T}_n$ to those of L_n, the matrix $C + S$ is exactly the same $\Gamma(\boldsymbol{\lambda})$ that was in the previous section found by linear interpolation. The procedure via circulant theory is, however, computationally much easier because it can be carried out with the aid of the fast Fourier transform, as follows.

We need the first rows of C and S. Let $\mathbf{e}_1$ be the first row of the identity matrix, $\mathbf{1}$ a row vector of all ones, $\mathbf{b} = (b_1, b_2, \ldots, b_n)$, and $\mathbf{d} = (d_1, d_2, \ldots, d_n)$. We have

$$\mathbf{e}_1 C = \mathbf{e}_1 FDF^{-1} = \mathbf{1}DF^{-1} = \mathbf{d}F^{-1}$$

and

$$\mathbf{e}_1 S = \mathbf{e}_1 WFB(WF)^{-1} = \mathbf{1}B(WF)^{-1} = \mathbf{b}F^{-1}W^{-1}.$$

Since $1/\zeta = \overline{\zeta}$, the algorithm (implemented in the Matlab code of section 1) follows:

1. Solve (2)–(3).
2. Put $d_{n-k+2} = d_k,\ k = 2, 3, \ldots, n_b$, and calculate
$$(c_1, c_2, \ldots, c_n) = \texttt{ifft}(d_1, d_2, \ldots, d_n).$$
3. Put $b_{n-k+1} = b_k,\ k = 2, 3, \ldots, n_d$, and calculate
$$(s_1, s_2, \ldots, s_n) = \texttt{ifft}(b_1, b_2, \ldots, b_n).$$
4. The desired initial Toeplitz matrix has first row $(c_1 + s_1, c_2 + \overline{\zeta} s_2, \ldots, c_n + \overline{\zeta}^{n-1} s_n)$.

3.3. Composition of lower-dimensional interpolation. We now discuss yet another way to find $\Gamma(\boldsymbol{\lambda})$. The reason for so doing is not in order to find an alternative computational procedure but to gain extra understanding into exactly where the interpolation is exact and where it is only approximate.

We refer to the set of points on the boundary of $\mathcal{T}_n$ for which the Toeplitz matrix is circulant (resp., skew-circulant) as the *circulant* (resp., *skew-circulant*) *hyperedge.* By the exact formula for T_k, to be exhibited in section 4, we see that all odd-numbered vertices are on the circulant hyperedge, and all even-numbered vertices are on the skew-circulant hyperedge. In fact, the circulant hyperedge is simply the $(n_d - 1)$-dimensional convex hull of the odd-numbered vertices, and the skew-circulant hyperedge is simply the $(n_b - 1)$-dimensional convex hull of the even-numbered vertices.

These two hyperedges map linearly to hyperedges of $\mathcal{L}_n$ that are in their turn the convex hulls of the odd-numbered vertices L_{2k+1} and even-numbered vertices L_{2k}, respectively.

The linear interpolation procedure can therefore be factorized as follows:

1. Find points $\boldsymbol{\lambda}_C$ and $\boldsymbol{\lambda}_S$, respectively, on the circulant and skew-circulant hyperedges of $\mathcal{L}_n$, so that the straight line segment connecting them passes through $\boldsymbol{\lambda}$. This implies that for some constant a, $0 \le a \le 1$, we have $\boldsymbol{\lambda} = a\boldsymbol{\lambda}_C + (1 - a)\boldsymbol{\lambda}_S$, with equality if and only if $\boldsymbol{\lambda}$ itself is on either of these hyperedges.
2. Find points $A_C = \Lambda^{-1}(\boldsymbol{\lambda}_C) = \Gamma(\boldsymbol{\lambda}_C)$ and $A_S = \Lambda^{-1}(\boldsymbol{\lambda}_S) = \Gamma(\boldsymbol{\lambda}_S)$ by the above interpolation procedure, which is exact when restricted to a hyperedge.
3. Approximate $\Lambda^{-1}(\boldsymbol{\lambda})$ by $\mathbf{t}_2 = aA_C + (1 - a)A_S$.

To summarize, if we view the n-dimensional linear interpolation process as a composition of three lower-dimensional interpolations, one in the $(n_d - 1)$-dimensional circulant hyperedge, one in the $(n_b - 1)$-dimensional skew-circulant hyperedge, and one along the curve joining a point in one hyperedge to a point in the other, only this last one-dimensional interpolation is not exact.

4. Theorems and proofs. In this section we give the theoretical basis for the interpolation method described above. The derivation, and indeed our whole procedure, depends on the validity of Conjecture 2. We need to show that the association of eigenvalues with the eigenvectors of the circulants and skew-circulants is the proper one and to exhibit the explicit formula (5) for the vertex matrices T_k.

The first two lemmas allow us to find all Toeplitz matrices that could possibly be vertices of the set $\mathcal{T}_n$. In the cases with multiplicity of the nonzero eigenvalue from 2 to $n - 2$, there is more than one Toeplitz matrix with the desired eigenvalues and we need to determine which is regular.

LEMMA 1. *If A is symmetric Toeplitz, then A^2 is Toeplitz if and only if A is circulant or skew-circulant.*

Proof. It is convenient to write A as unsymmetric and use the symmetry later. Let $B = A^2$, where

$$A = \begin{bmatrix} a_0 & a_1 & \dots & a_{n-2} & a_{n-1} \\ a_{-1} & a_0 & a_1 & \dots & a_{n-2} \\ \ddots & \ddots & \ddots & \ddots & \ddots \\ a_{-n+2} & \dots & a_{-1} & a_0 & a_1 \\ a_{-n+1} & a_{-n+2} & \dots & a_{-1} & a_0 \end{bmatrix}.$$

Number the rows and columns of B from 0 to $n-1$. Then

$$b_{j,k} = a_{-j}a_k + a_{-j+1}a_{k-1} + \cdots + a_{n-j-1}a_{k-n+1}.$$

Since $b_{j,k} = b_{j+1,k+1}$ we obtain

$$a_{-j}a_k = a_{n-j}a_{k-n}, \quad j = 1, 2, \dots, n-1, \ k = 1, 2, \dots, n-1. \tag{4}$$

Put $k = j$ in (4) and recall that A is symmetric to obtain

$$a_j^2 = a_{n-j}^2 \implies a_j = \pm a_{n-j}, \quad j = 1, 2, \dots, n-1.$$

A is circulant (resp., skew-circulant) if the positive (resp., negative) sign applies for every j. If A is neither circulant nor skew-circulant, then both signs occur; i.e., there exist indices j, k such that $a_j = a_{n-j} \neq 0$ and $a_k = -a_{n-k} \neq 0$. This gives $a_j a_k = -a_{n-j}a_{n-k} \neq 0$, which contradicts (4). ∎

LEMMA 2. *Any symmetric Toeplitz matrix A with k eigenvalues equal to 1 and $n-k$ eigenvalues equal to 0 can be represented as $A = F^{-1}DF$ or as $A = WFB(WF)^{-1}$, where the diagonals of D and B are, respectively, circulant compatible and skew-circulant compatible sequences containing k ones and $n-k$ zeros, and F is the Fourier matrix $F = [f_{jk}]$ with $f_{jk} = \omega^{(j-1)(k-1)}$, where $\omega = e^{-2\pi i/n}$.*

Proof. Since A is symmetric and has all eigenvalues either 0 or 1, it is a projection matrix and satisfies $A^2 = A$. Since A is Toeplitz, by the previous lemma it must be either circulant or skew-circulant. In the circulant case ($A = C$), let $D = F^{-1}CF$. Since columns k and $n-k+2$ of F are conjugates, d_k and d_{n-k+2} are also conjugates. C is symmetric, and therefore all eigenvalues are real; so $d_k = d_{n-k+2}$, which is the required condition.

Similarly, in the skew-circulant case ($A = S$), columns k and $n-k+1$ of WF are conjugates. Let $B = (WF)^{-1}SWF$; then b_k and b_{n-k+1} are also conjugates, implying $b_k = b_{n-k+1}$. ∎

The following theorem separates out the even and odd parts of the two eigenvectors belonging to a double eigenvalue. This will allow us to work with real eigenvectors, which is more convenient for the theory, although less convenient in computation. As usual in eigenvalue work, "unique" is used in the sense that a nonzero multiple of an eigenvector is not considered different, and that ordering of eigenvectors does not matter.

THEOREM 1. *The unique real eigenvector matrix U with only even and odd*

columns such that $D = U^{\mathsf{T}}CU$ is diagonal for all symmetric circulants C is given by

$$\begin{aligned}
u_{j,1} &= \frac{1}{\sqrt{n}},\\
u_{j,2k} &= \sqrt{\frac{2}{n}}\sin\frac{(2j-1)k\pi}{n}, \qquad k = 1, 2, \ldots, \lfloor \tfrac{n-1}{2} \rfloor,\\
u_{j,2k+1} &= \sqrt{\frac{2}{n}}\cos\frac{(2j-1)k\pi}{n}, \qquad k = 1, \ldots, \lfloor \tfrac{n-1}{2} \rfloor,\\
u_{j,n} &= \frac{(-1)^{j-1}}{\sqrt{n}} \qquad \textit{if } n \textit{ is even.}
\end{aligned}$$

The unique real eigenvector matrix V with only even and odd columns such that $B = V^{\mathsf{T}}SV$ is diagonal for all symmetric skew-circulants S is given by

$$\begin{aligned}
v_{j,2k-1} &= \sqrt{\frac{2}{n}}\cos\frac{(2j-1)(k-\frac{1}{2})\pi}{n}, \qquad k = 1, 2, \ldots, \lfloor \tfrac{n}{2} \rfloor,\\
v_{j,2k} &= \sqrt{\frac{2}{n}}\sin\frac{(2j-1)(k-\frac{1}{2})\pi}{n}, \qquad k = 1, 2, \ldots, \lfloor \tfrac{n}{2} \rfloor,\\
v_{j,n} &= \frac{(-1)^{j-1}}{\sqrt{n}} \qquad \textit{if } n \textit{ is odd.}
\end{aligned}$$

Proof. Let $\mathbf{f}_k$ denote the kth column of F. $\mathbf{f}_k$ and $\mathbf{f}_{n-k+2}$ are orthogonal and conjugate and span a two-dimensional subspace belonging to a double eigenvalue. The real and imaginary parts of $\mathbf{f}_k$ form an orthogonal basis for this subspace but do not form an even-odd pair of vectors. To achieve this, we divide $\mathbf{f}_k$ by ζ^{k-1}, where $\zeta = e^{-\pi i/n}$. Note that $f_{jk} = \zeta^{2(j-1)(k-1)}$. The jth component of $\mathbf{z} = \zeta^{k-1}\mathbf{f}_k$ is $z_j = \zeta^{(2j-1)(k-1)}$. Then $z_j z_{n-j+1} = \zeta^{2n(k-1)} = 1$. Since z_j has modulus 1, this relation means that $z_{n-j+1} = \overline{z}_j$. Therefore $\Re(\mathbf{z})$ is even and $\Im(\mathbf{z})$ is odd, giving two columns that go into U.

The proof for V is similar but easier because the relevant columns of WF require no further scaling: we can immediately take real and imaginary parts. ∎

The way we have chosen the columns of U and V requires that the eigenvalues of the circulant (resp., skew-circulant) be ordered as $D = \mathrm{diag}(d_1, d_2, d_2, d_3, d_3, \ldots, d_{n_d})$ (resp., $B = \mathrm{diag}(b_1, b_1, b_2, b_2, \ldots, b_{n_b})$) instead of the previous ordering used when diagonalizing by F (resp., WF). We get all possibilities for the matrices T_k in this way, e.g., to find T_4 when $n = 8$ requires one of the following twelve cases:

Diagonal of D		Diagonal of B
$(1;1,1;0,0;0,0;1)$		$(1,1;1,1;0,0;0,0)$
$(1;0,0;1,1;0,0;1)$		$(1,1;0,0;1,1;0,0)$
$(1;0,0;0,0;1,1;1)$	or	$(1,1;0,0;0,0;1,1)$
$(0;1,1;1,1;0,0;0)$		$(0,0;1,1;1,1;0,0)$
$(0;1,1;0,0;1,1;0)$		$(0,0;1,1;0,0;1,1)$
$(0;0,0;1,1;1,1;0)$		$(0,0;0,0;1,1;1,1)$

These represent all solutions to the inverse Toeplitz eigenvalue problem with four ones and four zeros as target eigenvalues, but we are not yet able to identify which of these matrices could serve as the required vertex of a set of regular matrices. To do that, we need to assume the validity of Conjecture 2.

THEOREM 2. *If Conjecture* 2 *is true, then for points near the circulant hyperedge the eigenvector matrix is near* U, *and for points near the skew-circulant hyperedge the eigenvector matrix is near* V, *when the eigenvalues are arranged in decreasing order.*

Proof. Let $C = M_n(c_1, c_2, \ldots, c_n)$ be a point on the circulant hyperedge with double eigenvalues only, so that $d_1 > d_3 > d_5 > \cdots$ but $d_2 = d_3$, $d_4 = d_5$, etc. Select an even-odd pair of vectors as basis for the invariant subspace associated with each eigenvalue. The eigenvalue matrix M such that $M^{\mathsf{T}}CM = \operatorname{diag}(d_j)$ must then be some permutation U_P of U in which odd-even pairs of columns stay together, i.e.,

$$U_P = \begin{bmatrix} \mathbf{u}_{p_1} & \mathbf{u}_{p_2} & \mathbf{u}_{p_3} & \ldots & \mathbf{u}_{p_n} \end{bmatrix}$$

with $p_3 = p_2+1$, $p_5 = p_4+1$, etc. The odd-numbered columns of U_P are a permutation of the odd-numbered columns of U and are therefore even vectors. This observation remains true even when the eigenvalue gaps are arbitrarily small.

The matrix $H = M_n(0, 1, 0, 0, \ldots, 0)$ is easily seen to be a regular Toeplitz matrix: just take $f(\theta) = 2\cos\theta$ in (1). By Conjecture 2, the perturbed matrix

$$C_\delta = C + \delta H \text{ with } \delta > 0$$

should also be regular. For small values of δ, we can obtain excellent approximations to the eigenvalues of C_δ with the aid of Rayleigh quotients. Let λ be the eigenvalue of C having eigenvector $\mathbf{u}_k$. Then an approximation accurate to $\mathcal{O}(\delta^2)$ of the eigenvalue of C_δ whose eigenvector is close to $\mathbf{u}_k$ is

$$\begin{aligned} \mathbf{u}^{\mathsf{T}}(T + \delta H)\mathbf{u} &= \mathbf{u}^{\mathsf{T}}T\mathbf{u} + \delta\mathbf{u}^{\mathsf{T}}H\mathbf{u} \\ &= \lambda + 2\delta(u_{1,k}u_{2,k} + u_{2,k}u_{3,k} + \cdots + u_{n-1,k}u_{n,k}). \end{aligned}$$

We now look at the eigenvalues corresponding to odd-numbered columns of U_P. To an accuracy of $\mathcal{O}(\delta^2)$, these are

$$d_1 + 2\delta\mu_{p_1}, d_2 + 2\delta\mu_{p_3}, \ldots, d_k + 2\delta\mu_{p_{2k-1}},$$

where

$$\mu_k = u_{1,k}u_{2,k} + u_{2,k}u_{3,k} + \cdots + u_{n-1,k}u_{n,k}.$$

Since $d_k - d_{k+1}$ may be arbitrarily small, and the ordering of the eigenvalues needs to be preserved if the matrix is to remain regular, the columns of U_P need to be arranged so that the quantities $\mu_{p_1}, \mu_{p_3}, \mu_{p_5}, \ldots$ form a decreasing sequence. Straightforward if tedious trigonometric manipulation yields

$$\begin{aligned} \mu_1 &= \tfrac{n-1}{n}, \\ \mu_{2k+1} &= \tfrac{n-1}{n}\cos\tfrac{2k\pi}{n} - \tfrac{1}{n}, \quad k = 1, 2, \ldots, \lfloor \tfrac{n-1}{2} \rfloor. \end{aligned}$$

From the monotonicity of the cosine function over the interval $(0, \pi)$ it follows that $\mu_1 > \mu_3 > \mu_5 > \cdots$, which shows that the columns of U are already in the correct sequence.

A similar analysis showing that the columns of V are already in the correct sequence is left to the reader. ∎

With the sequence of eigenvalues determined, it is now a simple exercise to calculate the vertex matrices T_k.

THEOREM 3. *The vertex matrices T_k are given by*

$$T_k = \frac{1}{n} M_n \left(k, \frac{\sin k\theta}{\sin \theta}, \frac{\sin 2k\theta}{\sin 2\theta}, \ldots, \frac{\sin (n-1)k\theta}{\sin (n-1)\theta} \right), \tag{5}$$

where $\theta = \frac{\pi}{n}$.

Proof. The diagonal matrices D and B must start with k ones and end with $n-k$ zeros. When k is odd (resp., even), this pattern is compatible with the circulant (resp., skew-circulant) hyperedge. We obtain

$$T_m = \sum_{j=1}^{m} \mathbf{u}_j \mathbf{u}_j^{\mathsf{T}} \text{ when } m \text{ is odd;}$$
$$T_m = \sum_{j=1}^{m} \mathbf{v}_j \mathbf{v}_j^{\mathsf{T}} \text{ when } m \text{ is even.}$$

This gives

$$\begin{aligned} T_1 &= \tfrac{1}{n} M_n(1, 1, \ldots, 1), \\ T_{2k+1} &= T_{2k-1} + \tfrac{2}{n} M_n \left(\sin \frac{k\pi}{n} \sin \frac{(2j-1)k\pi}{n} + \cos \frac{k\pi}{n} \cos \frac{(2j-1)k\pi}{n},\ j = 1, 2, \ldots, n \right) \\ &= T_{2k-1} + \tfrac{2}{n} M_n \left(\cos \frac{2(j-1)k\pi}{n},\ j = 1, 2, \ldots, n \right). \end{aligned}$$

Similarly,

$$\begin{aligned} T_0 &= M_n(0, 0, \ldots, 0), \\ T_{2k} &= T_{2k-2} + \tfrac{2}{n} M_n \left(\cos \frac{(j-1)(2k-1)\pi}{n},\ j = 1, 2, \ldots, n \right). \end{aligned}$$

The recursions in both cases are seen to be special cases of

$$T_{k+1} = T_{k-1} + \tfrac{2}{n} M_n \left(\cos(j-1)k\theta,\ j = 1, 2, \ldots, n \right), \quad k = 1, 2, \ldots, n-2. \tag{6}$$

It is readily seen that the expression (5) satisfies this recursion. ∎

Remark. The entries of T_k are scaled values of Chebyshev polynomials of the second kind, and the recursion (6) is a well known relation ((22.5.8) in [1]) between Chebyshev polynomials.

The matrices U and V share a conspicuous property which is easy to verify: for all n, the kth columns of U and V show $k-1$ sign changes ($k = 1, 2, \ldots, n$). This leads us to the following.

CONJECTURE 3. *The sequence of elements of the eigenvector corresponding to the kth largest eigenvalue of a regular Toeplitz matrix shows $k-1$ sign changes, $k = 1, 2, \ldots, n$.*

Our final result exhibits a case where the initial values are not exact but show the correct qualitative behavior.

THEOREM 4. *If the sequence $\boldsymbol{\lambda}$ contains only three distinct eigenvalues $\alpha < \beta < \gamma$ of multiplicity m_α, m_β, and m_γ, respectively, then the matrix $\Gamma(\boldsymbol{\lambda})$ has the eigenvalue β with multiplicity m_β.*

Proof. Without loss of generality, assume $\boldsymbol{\lambda}$ is standardized so that $\alpha = 0$ and $\gamma = 1$. Let $\boldsymbol{\lambda}_1$ and $\boldsymbol{\lambda}_2$ be of the form $(0, 0, \ldots, 0, 1, 1, \ldots, 1)$ with m_α and $m_\alpha + m_\beta$ zeros, respectively. Then

$$\boldsymbol{\lambda} = \beta\boldsymbol{\lambda}_1 + (1-\beta)\boldsymbol{\lambda}_2.$$

Since $\boldsymbol{\lambda}_1$ and $\boldsymbol{\lambda}_2$ are vertices,

$$\Gamma(\boldsymbol{\lambda}) = \beta\Gamma(\boldsymbol{\lambda}_1) + (1-\beta)\Gamma(\boldsymbol{\lambda}_2).$$

Assume $\boldsymbol{\lambda}_1$ and $\boldsymbol{\lambda}_2$ belong to different hyperedges; otherwise the interpolation is exact and there is nothing to prove. Now consider

$$\Gamma(\boldsymbol{\lambda}) - \beta I = \beta(\Gamma(\boldsymbol{\lambda}_1) - I) + (1-\beta)\Gamma(\boldsymbol{\lambda}_2).$$

The interpolation is exact on hyperedges; therefore the matrices $\Gamma(\boldsymbol{\lambda}_1) - I$ and $\Gamma(\boldsymbol{\lambda}_2)$ have the eigenvalue 1 to the multiplicity m_α and m_γ, respectively. Since the two hyperedges are disjoint from each other and from the space of multiples of I, the rank of $\Gamma(\boldsymbol{\lambda}) - \beta I$ is $m_\alpha + m_\gamma$. This implies that the nullspace of $\Gamma(\boldsymbol{\lambda}) - \beta I$ has dimension m_β, which was to be proven. □

The theorem says nothing about the other two eigenvalues, and in fact in general they do not even stay multiple. It does imply, though, that the edge parametrized by $(0, \beta, \beta, \ldots, \beta, 1)$ corresponds to a straight line in the Landau standardization, even in the case of odd n when the interpolation is not exact.

5. Numerical examples. Following a suggestion of Higham [7], we tried finding extremely unfavorable examples by a method of direct search. The quantity to be maximized over standardized eigenvalue sequences $\boldsymbol{\lambda}$ was chosen as

$$f(\boldsymbol{\lambda}) = \|\boldsymbol{\lambda} - \Lambda(\Gamma(\boldsymbol{\lambda}))\|_1,$$

in other words, the sum of the absolute differences between the original sorted eigenvalues and those of the approximating Toeplitz matrix. The worst case was persistently found when there were two well-separated eigenvalues near 0.29 and 0.71, and all other eigenvalues very close to 0 and 1, with the two clusters of as equal size as possible. We model this behavior by

$$\lambda_k = \tfrac{1}{2}(1 + \tanh h(k - k_0)), \quad k = 1, 2, \ldots, n,$$

where $k_0 = \lfloor \frac{n}{2} \rfloor + \frac{1}{2}$ and $h = 2\tanh^{-1}(\sqrt{2} - 1)$. Eigenvalues distributed as above will be called "difficult" here. For example, for $n = 7$, the formula gives

$$\boldsymbol{\lambda} = (0.0120, 0.0664, 0.2929, 0.7071, 0.9336, 0.9880, 0.9979).$$

The eigenvalues of $\Gamma(\boldsymbol{\lambda})$ are

$$(0.0216, 0.1052, 0.3252, 0.6815, 0.8942, 0.9747, 0.9955).$$

From the point of view of iterative methods for the inverse Toeplitz eigenvalue problem, however, the interesting question is not so much how good the initial values are as how well they perform in an iterative setting. A typical relevant question is,

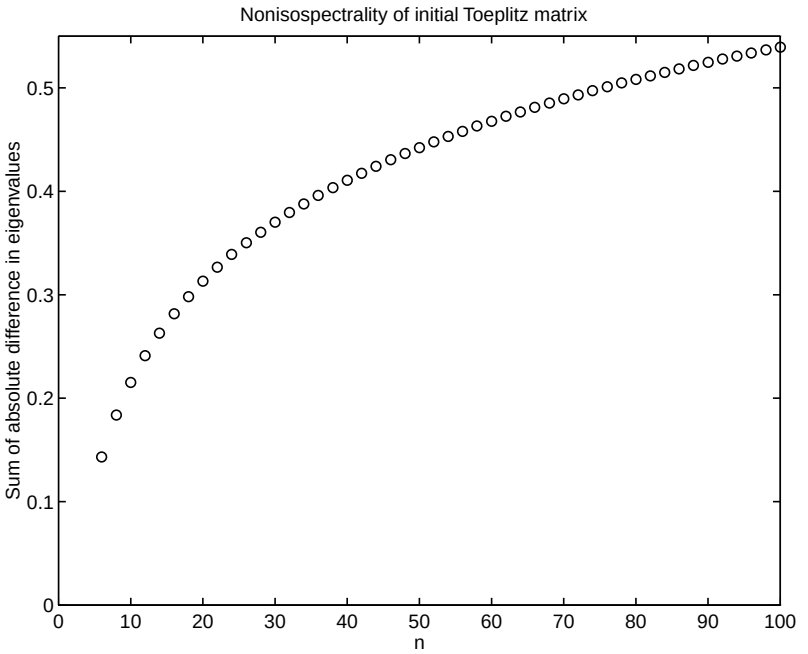


FIG. 3. *Sum of absolute deviation of eigenvalues of initial matrix* $\Gamma(\boldsymbol{\lambda})$ *for "difficult" target* $\boldsymbol{\lambda}$ *for even values of* n.

When Q are the eigenvectors of $\Gamma(\boldsymbol{\lambda})$ for "difficult" $\boldsymbol{\lambda}$, how close is $Q\,\mathrm{diag}(\lambda_k)Q^{\mathsf{T}}$ to a Toeplitz matrix? In the above case, we find

$$Q\,\mathrm{diag}(\lambda_k)Q^{\mathsf{T}} = \begin{bmatrix} 0.5743 & 0.3099 & -0.0613 & -0.0608 & 0.0358 & 0.0144 & -0.0278 \\ 0.3099 & 0.5677 & 0.2946 & -0.0563 & -0.0575 & 0.0351 & 0.0144 \\ -0.0613 & 0.2946 & 0.5711 & 0.2947 & -0.0556 & -0.0575 & 0.0358 \\ -0.0608 & -0.0563 & 0.2947 & 0.5718 & 0.2947 & -0.0563 & -0.0608 \\ 0.0358 & -0.0575 & -0.0556 & 0.2947 & 0.5711 & 0.2946 & -0.0613 \\ 0.0144 & 0.0351 & -0.0575 & -0.0563 & 0.2946 & 0.5677 & 0.3099 \\ -0.0278 & 0.0144 & 0.0358 & -0.0608 & -0.0613 & 0.3099 & 0.5743 \end{bmatrix}.$$

In Figures 3 and 4 we show $f(\boldsymbol{\lambda})$ and the deviation of $Q\,\mathrm{diag}(\lambda_k)Q^{\mathsf{T}}$ from Toeplitz form as a function of n.

One could also ask, When applied in practice in conjunction with some iterative method, how many iterations do the new starting values require in comparison to others that have been suggested? We applied Algorithm 1 from [10] with the difficult eigenvalues. The algorithm in question starts from an initial eigenvector matrix Q and proceeds as follows:

1. Calculate an approximate Toeplitz matrix T from the current Q, and form $A = Q^{\mathsf{T}}TQ$. This is obviously an $O(n^3)$ operation. The algorithm has two possible states: in the "normal" state the approximation is performed by Newton's method, and in the "emergency" state by projecting $Q\,\mathrm{diag}(\lambda_k)Q^{\mathsf{T}}$ to the closest Toeplitz matrix in the Frobenius norm.
2. Perform Jacobi rotations that successively annihilate the largest off-diagonal

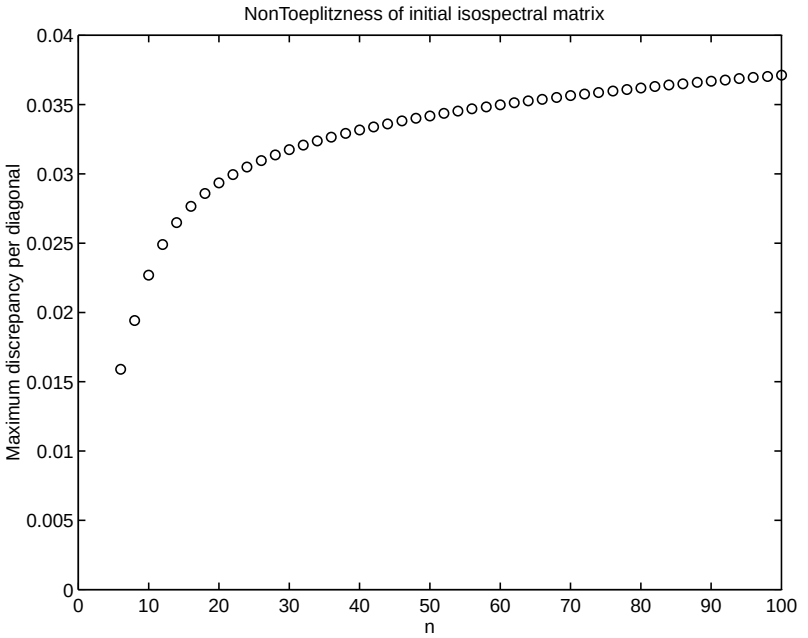


FIG. 4. *Largest difference between elements on the same diagonal in the initial isospectral matrix* $Q\,\mathrm{diag}(\lambda_k)Q^{\mathsf{T}}$ *with "difficult" eigenvalues* λ_k *for even values of* n.

elements of A, and apply the corresponding inverse rotations to Q. The process is not carried to completion (as Newton's method would require): instead, a rotation is made only as long as a certain criterion $\phi(Q)$ of closeness to the solution keeps on decreasing. Each rotation involves a search for the largest element and is followed by a rank 1 update of a certain $n \times n$ matrix, both of which are $O(n^2)$ operations.

3. If $\phi(Q)$ is small enough, exit with "success." If no rotations whatsoever were made in step 2, change to emergency state (if emergency state was already in effect, exit with "failure"); later the algorithm will revert to normal state when $\phi(Q)$ is small enough. Return to step 1.

In Table 1 we show the performance of the above algorithm from the initial values suggested here, compared to those in [9] and [12]. For small problems ($n \leq 10$) there is little difference between the three cases. For larger n, the use of the new initial values leads to fewer outer iterations and fewer rotations in total than the other two cases, and in no case leads to a situation where the emergency state of the algorithm is entered.

Also in a method based on isospectral flows, these initial values have been reported to be efficacious [13, pp. 115–117].

6. Conclusion. The motivation for much of the paper depends on the unproved Conjecture 2. It would be very interesting if this conjecture could be proved; in fact, the same can be said of progress on any of the conjectures cited or proposed in this paper. Nevertheless, as the numerical examples show, the initial values are practically useful when solving the inverse Toeplitz eigenvalue problem by iterative methods.

The present state of the art is that no constructive proof for the mere existence of a solution to the problem has been found yet: in particular, no iterative method

TABLE 1
Performance of Algorithm 1 *of* [10] *with initial values suggested by* [9], [12] *and the present paper* [0]. *In this algorithm, each rotation requires* $O(n^2)$ *and each iteration* $O(n^3)$ *multiplications.*

n	Iterations			Emergencies			Rotations		
	[9]	[12]	[0]	[9]	[12]	[0]	[9]	[12]	[0]
5	3	3	2	0	0	0	13	17	9
6	3	3	2	0	0	0	22	19	17
7	3	3	3	0	0	0	32	32	32
8	3	3	3	0	0	0	44	44	53
9	4	4	3	0	0	0	62	61	57
10	4	4	3	0	0	0	74	84	70
11	5	4	3	0	0	0	107	101	69
12	5	5	3	0	0	0	138	137	74
13	5	5	3	0	0	0	151	151	86
14	5	4	3	0	0	0	184	156	106
15	5	6	4	0	0	0	190	195	138
16	5	5	4	0	0	0	226	224	166
17	6	6	4	0	0	0	264	267	163
18	7	7	4	0	0	0	294	315	186
19	8	8	4	1	0	0	326	345	195
20	8	6	5	1	0	0	364	356	261
21	7	9	5	0	1	0	400	429	278
22	7	8	5	0	0	0	426	472	314
23	8	9	5	0	0	0	460	476	298
24	8	10	5	0	1	0	512	490	299
25	10	9	5	1	0	0	519	516	326
26	10	10	5	1	0	0	630	596	322
27	11	11	5	1	1	0	579	655	289
28	9	8	6	0	0	0	700	628	388
29	11	12	6	1	1	0	648	706	386
30	8	11	6	0	1	0	702	714	354

has been proved to be convergent. The initial values suggested in this paper may aid in finding such a method. Moreover, Conjecture 3 suggests the investigation of numerical methods that take care to respect the sign change pattern of the eigenvector matrix. This could be an intriguing area for future research.

Acknowledgment. An anonymous referee made two useful suggestions that improved the presentation and impact of this paper.

REFERENCES

[1] M. ABRAMOWITZ AND I. A. STEGUN, EDS., *Handbook of Mathematical Functions*, National Bureau of Standards, Washington, D.C., 1964.
[2] A. CANTONI AND P. BUTLER, *Eigenvalues and eigenvectors of symmetric centrosymmetric matrices*, Linear Algebra Appl., 13 (1976), pp. 275–288.
[3] M. T. CHU, *On a Newton Method for the Inverse Toeplitz Eigenvalue Problem*, manuscript; available from http://www4.ncsu.edu/~mtchu/Research/Papers/itep.ps.
[4] M. T. CHU AND K. R. DRIESSEL, *Can Real Symmetric Toeplitz Matrices Have Arbitrary Real Spectra?*, Technical Report 88-1, Department of Mathematics, Idaho State University, Pocatello, ID, 1989.
[5] P. J. DAVIS, *Circulant Matrices*, Wiley, New York, 1979.
[6] P. DELSARTE AND Y. GENIN, *Spectral properties of finite Toeplitz matrices*, in Mathematical Theory of Networks and Systems, P.A. Fuhlmann, ed., Springer, Berlin, 1984, pp. 194–213.
[7] N. J. HIGHAM, *Optimization by direct search in matrix computations*, SIAM J. Matrix Anal. Appl., 14 (1993), pp. 317–333.
[8] H. J. LANDAU, *The inverse eigenvalue problem for real symmetric Toeplitz matrices*, J. Amer. Math. Soc., 7 (1994), pp. 749–767.

[9] D. P. Laurie, *A numerical approach to the inverse Toeplitz eigenproblem*, SIAM J. Sci. Statist. Comput., 9 (1988), pp. 401–405.
[10] D. P. Laurie, *Solving the inverse eigenvalue problem via the eigenvector matrix*, J. Comput. Appl. Math., 35 (1991), pp. 277–289.
[11] W. F. Trench, *Interlacement of the even and odd spectra of real symmetric Toeplitz matrices*, Linear Algebra Appl., 195 (1993), pp. 59–68.
[12] W. F. Trench, *Numerical solution of the inverse eigenvalue problem for real symmetric Toeplitz matrices*, SIAM J. Sci. Comput., 18 (1997), pp. 1722–1736.
[13] A. Zanna, *On the Numerical Solution of Isospectral Flows*, Ph.D. thesis, University of Cambridge, Cambridge, UK, 1998.

SIAM J. SCI. COMPUT.
Vol. 22, No. 6, pp. 2256–2281
© 2001 Society for Industrial and Applied Mathematics

HIGH RESOLUTION SCHEMES FOR CONSERVATION LAWS WITH LOCALLY VARYING TIME STEPS*

CLINT DAWSON† AND ROBERT KIRBY†

Abstract. We develop upwind methods which use limited high resolution corrections in the spatial discretization and local time stepping for forward Euler and second order time discretizations. L^∞ stability is proven for both time stepping schemes for problems in one space dimension. These methods are restricted by a local CFL condition rather than the traditional global CFL condition, allowing local time refinement to be coupled with local spatial refinement. Numerical evidence demonstrates the stability and accuracy of the methods for problems in both one and two space dimensions.

Key words. spatially varying time steps, upwinding, conservation laws

AMS subject classifications. 35L65, 65M12, 65M30

PII. S1064827500367737

1. Introduction.

Hyperbolic conservation laws model a wide range of physically interesting phenomena such as gas dynamics, shallow water flow, and advection of contaminants. Conservative high resolution methods with explicit time discretization have proven effective in capturing the sharp, moving fronts common in these applications. It is well known that such methods require the time step to satisfy a CFL condition in order to guarantee stability.

Local spatial refinement is often introduced in order to resolve these fronts more efficiently. However, this local refinement reduces the allowable time step for the explicit time discretizations typically employed. Rather than considering a fully implicit approach, in which the step size is often constrained by the nonlinear convergence anyway, we consider a method which allows the time step to vary spatially and satisfy a local CFL condition. In this way, we can increase the efficiency of the time stepping significantly in certain situations.

Another situation where local time stepping is useful is when modeling transport of some quantity by a highly varying velocity field. The advection of a species is described by an equation of the form

$$c_t + \nabla \cdot (\mathbf{u}c) = q, \tag{1}$$

where c is the concentration of the species, $\mathbf{u}$ is the velocity of the fluid transporting the species, and q represents source/sink terms. High resolution schemes have become popular for these problems, primarily because they are conservative and satisfy a maximum principle. The CFL constraint on the time step is determined by the ratio of the mesh spacing and the magnitude of the velocity. In certain cases, specifically in the presence of injection or production wells, the magnitude of $\mathbf{u}$ can vary substantially throughout the domain. In these situations, even if the mesh spacing is uniform, using a global CFL time step can be very restrictive, and local time stepping can result in substantial savings in computation time. It is important that the

*Received by the editors February 14, 2000; accepted for publication (in revised form) October 30, 2000; published electronically April 26, 2001. This work was supported by NSF grant DMS-9805491.
http://www.siam.org/journals/sisc/22-6/36773.html

†Center for Subsurface Modeling–C0200, Texas Institute for Computational and Applied Mathematics, University of Texas at Austin, Austin, TX 78712 (clint@ticam.utexas.edu, rob@ticam.utexas.edu).

local time stepping preserve the conservation and maximum principle properties of the underlying method. This situation was explored in some detail from a practical viewpoint in [3], and we will explore it from a more theoretical viewpoint here.

Local time stepping for one-dimensional scalar conservation laws was first proposed in Osher and Sanders [11]. They gave a thorough analysis of a first order spatial discretization with a local forward Euler time stepping scheme. This scheme allows each element to take either an entire time step or some fixed M smaller steps. In [3], the first author examined local time stepping for advection equations of the form (1), using high resolution schemes with slope limiters. Numerical results in two space dimensions were presented.

Another approach to local time stepping involves automatically taking smaller time steps where the mesh is refined. Berger and Oliger developed such an approach in [1], in which refined grids are laid over regions of the coarse mesh. These fine grids can have different orientation than the coarse one, and need not be nested. When the problem is integrated in time, small time steps are taken on the refined mesh and large time steps on the coarse mesh. Information is then passed between the grids by means of injection and interpolation. This approach allows higher order time integration, as the computation for each time step is done independently once the information is passed at the beginning of the time step.

In other work, Kallinderis and Baron [8] coupled the time refinement to the spatial refinement but used a single nonuniform mesh rather than multiple meshes. This work presented several methods (all first order) of handling the interface between regions with different time steps.

Flaherty et al. [5] have developed a parallel, adaptive discontinuous Galerkin method with a local forward Euler scheme which relies on interpolating values in time at interfaces between time steps of different sizes. This scheme, however, does not appear to conserve flux along these interfaces. Also, only first order in time methods are discussed.

In this paper, we seek to extend the work in [11] and [3] in two ways. First, we will show a maximum principle for a local forward Euler method when limited slopes are included. Second, we will show that the main ideas of local forward Euler discretizations may be extended to second order in time by way of the TVD Runge–Kutta methods of Gottlieb and Shu [6] and Shu [13]. A stability result for a constant coefficient case will be derived for this higher order method. We will also indicate how to extend this result to nonlinear problems. This analysis shows that we only need to satisfy a local CFL condition on each element.

The paper is outlined as follows. First, in section 2, we formulate a high resolution local forward Euler time discretization that employs some limited correction to the piecewise constant solution. In section 2.1, a maximum principle is derived for this approach. Then, a local time stepping procedure based on a second order time discretization is described in section 3, and a maximum principle for a simple case is given in section 3.1. Some extensions and implementation details are also discussed in this section. Finally, numerical results validating the theory are presented in section 4.

2. A high resolution method. We first consider the scalar conservation law

$$c_t + f(c)_x = 0, \tag{2}$$

together with the initial condition

$$c(x,0) = c_0(x). \tag{3}$$

We will partition the real line into intervals $I_j = \{x : x_{j-\frac{1}{2}} \le x < x_{j+\frac{1}{2}}\}$. We use the difference operator Δ_+ to denote a forward difference. That is, $\Delta_+ c_j \equiv c_{j+1} - c_j$ for some quantity c. A similar definition holds for Δ_- with $\Delta_- c_j \equiv c_j - c_{j-1}$. By integrating (2) over each I_j and using some consistent, Lipschitz numerical flux h at the cell edges, we obtain the semidiscrete scheme

$$\frac{dc_j}{dt} = -\frac{1}{\Delta x_j}\Delta_+ h(c_{j-1}, c_j), \tag{4}$$

where c_j is an approximation to the integral average of c over I_j. Examples of h include the Godunov flux and the Lax–Friedrichs flux. It is important that h is nondecreasing in the first variable and nonincreasing in the second variable. The initial condition is defined by simple projection. A standard forward Euler discretization can also be obtained in the obvious way.

In order to improve the accuracy of the above approach, we incorporate higher order "corrections" into the method. There are several ways of constructing these corrections. These include the piecewise linear reconstruction used in the monotone upwind scheme for conservation laws (MUSCL) of van Leer [14], the ENO reconstruction schemes of Harten and Osher [7], and the Runge–Kutta discontinuous Galerkin methods of Cockburn and Shu [2]. We shall not deal exclusively with any specific method but require certain properties of whichever is used. Since we are interested in second order accuracy, all of our examples assume linear approximations in space in each interval I_j.

After constructing the corrections and limiting them, we add them to the means to construct more accurate left and right states for the Riemann solutions. The corrections are denoted by tildes. Left states at an interface $x_{j+1/2}$ shall be denoted as $c_j + \tilde{c}_j \equiv c_{j+\frac{1}{2},L}$ and right states as $c_{j+1} - \tilde{\tilde{c}}_{j+1} \equiv c_{j+\frac{1}{2},R}$. In this way, the numerical flux at the edge is given by $h(c_{j+\frac{1}{2},L}, c_{j+\frac{1}{2},R})$.

Vital to the analysis is the assumption that the corrections introduce no new extrema into the solution. This allows corrections derived from the *minmod* slope limiter (described below) but rules out those derived from the modified *minmod* limiter of Shu [12] and the ENO schemes as introduced by Harten and Osher [7]. We can quantify this restriction as

$$-\theta \le \frac{\Delta_+ \tilde{\tilde{c}}_j}{\Delta_+ c_j} \le 1 \tag{5}$$

and

$$-\theta \le -\frac{\Delta_+ \tilde{c}_j}{\Delta_+ c_j} \le 1, \tag{6}$$

where $\theta \ge 0$.

The number θ is a chosen parameter which appears in the CFL condition. For example, see [2]. Geometrically, θ is related to the largest allowable ratio of differences between left and right states at consecutive edges relative to the means. This ratio can be allowed to be larger than 1, and we will see that such a situation will necessitate a restriction of the time step in order to guarantee a maximum principle. Consider the cells j and $j+1$ in Figure 1. The value $c_{j+\frac{1}{2},L} - c_{j-\frac{1}{2},L} \ge c_{j+1} - c_j$; however, the differences have the same sign. Thus

$$0 \le \frac{c_{j+\frac{1}{2},L} - c_{j-\frac{1}{2},L}}{c_{j+1} - c_j} = 1 + \frac{\Delta_+ \tilde{c}_j}{\Delta_+ c_j}, \tag{7}$$

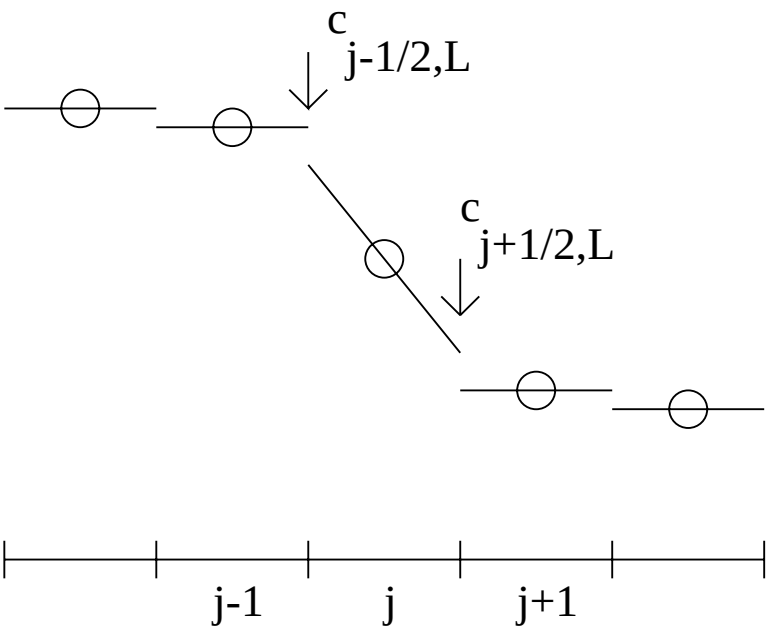


FIG. 1. *Slopes increase ratio of left states to means.*

and if θ satisfies (6), then

$$0 \le \frac{c_{j+\frac{1}{2},L} - c_{j-\frac{1}{2},L}}{c_{j+1} - c_j} \le 1 + \theta. \tag{8}$$

Specifically, suppose that we compute a slope, δc_j, using the *minmod* limiter. Thus,

$$\delta c_j = \begin{cases} \min\left(\frac{|\Delta_+ c_j|}{\Delta_+ x_j}, \frac{|\Delta_- c_j|}{\Delta_- x_j}\right) * \operatorname{sgn}(\Delta_+ c_j) & \text{if } \operatorname{sgn}(\Delta_+ c_j) = \operatorname{sgn}(\Delta_- c_j), \\ 0 & \text{otherwise.} \end{cases} \tag{9}$$

Set $\tilde{c}_j^u = \frac{\Delta x_j}{2} \delta c_j$. Then set

$$\tilde{c}_j = \begin{cases} \tilde{c}_j^u & \text{if } |\tilde{c}_j^u| \le |\Delta_\pm c_j|, \\ 0 & \text{otherwise.} \end{cases} \tag{10}$$

This last limiting step is necessary only in the case of nonuniform mesh. Finally, $\tilde{\tilde{c}}_j = \tilde{c}_j$. The corrections then have the property that

$$\operatorname{sgn}\left(\frac{\tilde{c}_j}{\Delta_\pm c_j}\right) \ge 0. \tag{11}$$

We can thus make the bound

$$1 + \frac{\Delta_+ \tilde{c}_j}{\Delta_+ c_j} \le 1 + \frac{\tilde{c}_{j+1}}{\Delta_+ c_j} \le 2. \tag{12}$$

We also have

$$1 + \frac{\Delta_+ \tilde{c}_j}{\Delta_+ c_j} \ge 1 - \frac{\tilde{c}_j}{\Delta_+ c_j} \ge 0 \tag{13}$$

and

$$1 - \frac{\Delta_+ \tilde{\tilde{c}}_j}{\Delta_+ c_j} \geq 1 - \frac{\tilde{c}_{j+1}}{\Delta_+ c_j} \geq 0. \tag{14}$$

Hence, (5) and (6) hold automatically with the choice $\theta = 1$.

More generally, suppose corrections $\tilde{c}_j^u$ and $\tilde{\tilde{c}}_j^u$ are computed by some means, and θ is chosen such that $0 \leq \theta \leq 1$. Define

$$\tilde{c}_j = \begin{cases} \tilde{c}_j^u & \text{if } 2|\tilde{c}_j^u| \leq \theta |\Delta_\pm c_j|, \\ 0 & \text{otherwise,} \end{cases} \tag{15}$$

with an analogous definition for $\tilde{\tilde{c}}_j$. Then

$$1 + \frac{\Delta_+ \tilde{c}_j}{\Delta_+ c_j} \geq 1 - \frac{|\tilde{c}_{j+1}|}{|\Delta_+ c_j|} - \frac{|\tilde{c}_j|}{|\Delta_+ c_j|} \geq 1 - \theta \geq 0 \tag{16}$$

and

$$1 + \frac{\Delta_+ \tilde{c}_j}{\Delta_+ c_j} \leq 1 + \theta. \tag{17}$$

We now present the first order local time discretization. It is a straightforward extension of the method in [11]. The only difference is that the flux terms are computed using the corrected quantities $c_{j+1/2,L}$, $c_{j+1/2,R}$ rather than piecewise constants. We allow each element to take either a whole time step or some fixed M substeps per main time step.

We begin by noting that the intervals I_j form a partition of the real line. In addition to the spatial partition, we also introduce the temporal partition of $[0, T)$ into time intervals $[t^n, t^{n+1}), n = 0, \ldots, N-1$, with $t^0 = 0$ and $t^N = T$. We denote $\Delta t^n \equiv t^{n+1} - t^n$ and define $\lambda_j^n \equiv \frac{\Delta t^n}{\Delta x_j}$. In order to describe the local time stepping scheme, we will need to further partition the time steps on certain elements. We denote by $\mathcal{C}^n$ the set of all indices j such that a single time step is taken from t^n to t^{n+1} on I_j. On the rest of the elements, we partition the time step $[t^n, t^{n+1})$ into the union of substeps $[t^{n+\eta_l}, t^{n+\eta_{l+1}})$, $l = 0, \ldots, M-1$. Further, $\{\sigma_k\}_{k=1}^M$ is a sequence of positive numbers summing to unity. The numbers η_l are given as the cumulative sum of the σ_k, that is, $\eta_l = \sum_{k=1}^l \sigma_k$, and $\eta_0 = 0$. Correspondingly, the substeps in the time interval are given by $t^{n+\eta_{l+1}} = t^{n+\eta_l} + \sigma_{l+1} \Delta t^n$. Notice that the elements on which the local steps are taken may change over time.

With this notation established, we can modify the predictor-corrector scheme of Osher and Sanders to a so-called high resolution scheme in space.

Following [11], for each $k = 1, \ldots, M-1$, the "predictor" is defined by

$$c_j^{n+\eta_k} = \begin{cases} c_j^n, & j \in \mathcal{C}^n, \\ c_j^n - \lambda_j^n \sum_{l=0}^{k-1} \sigma_{l+1} \Delta_+ h(c_{j-1/2,L}^{n+\eta_l}, c_{j-1/2,R}^{n+\eta_l}), & j \notin \mathcal{C}^n, \end{cases} \tag{18}$$

and the "corrector" is

$$c_j^{n+1} = c_j^n - \lambda_j^n \sum_{l=0}^{M-1} \sigma_{l+1} \Delta_+ h(c_{j-\frac{1}{2},L}^{n+\eta_l}, c_{j-\frac{1}{2},R}^{n+\eta_l}). \tag{19}$$

Notice that if $j-1, j, j+1 \in \mathcal{C}^n$, then the method on I_j reduces to forward Euler with a step size of Δt^n.

2.1. A maximum principle. In this section, we present a maximum principle for the method (18)–(19). The maximum principle will verify that L^∞ stability is attainable with only local assumptions regarding the mesh, time step, velocity, and corrections. It is vital to this analysis that we use a "strict" limiter (i.e., one that does not introduce new extrema).

We introduce some notation for the scheme as follows. We define terms involving the differences of the corrections relative to the means:

$$\kappa_{j,1} \equiv 1 - \frac{\Delta_- \tilde{\tilde{c}}_j}{\Delta_- c_j} \tag{20}$$

and

$$\kappa_{j,2} \equiv 1 + \frac{\Delta_- \tilde{c}_j}{\Delta_- c_j}, \tag{21}$$

where we may have a superscript of n,$n+\eta_l$, etc. Note that (5) and (6) give us the bounds $0 \le \kappa_{j,1}, \kappa_{j,2} \le 1+\theta$. We also define the coefficients $\gamma_{j,1}$ and $\gamma_{j,2}$ as the local Lipschitz coefficients of h. That is,

$$\gamma_{j,1}^{n+\eta_l} \equiv -\Lambda_j^{n+\eta_l} \left[\frac{h\left(c_{j+\frac{1}{2},L}^{n+\eta_l}, c_{j+\frac{1}{2},R}^{n+\eta_l}\right) - h\left(c_{j+\frac{1}{2},L}^{n+\eta_l}, c_{j-\frac{1}{2},R}^{n+\eta_l}\right)}{c_{j+\frac{1}{2},R}^{n+\eta_l} - c_{j-\frac{1}{2},R}^{n+\eta_l}} \right] \tag{22}$$

and

$$\gamma_{j,2}^{n+\eta_l} \equiv -\Lambda_j^{n+\eta_l} \left[\frac{h\left(c_{j+\frac{1}{2},L}^{n+\eta_l}, c_{j-\frac{1}{2},R}^{n+\eta_l}\right) - h\left(c_{j-\frac{1}{2},L}^{n+\eta_l}, c_{j-\frac{1}{2},R}^{n+\eta_l}\right)}{c_{j+\frac{1}{2},L}^{n+\eta_l} - c_{j-\frac{1}{2},L}^{n+\eta_l}} \right], \tag{23}$$

where

$$\Lambda_j^{n+\eta_l} = \begin{cases} \lambda_j^n & \text{if } j-1, j \text{ or } j+1 \in \mathcal{C}^n, \\ \sigma_{l+1}\lambda_j^n & \text{otherwise.} \end{cases} \tag{24}$$

If the flux f is smooth enough, then we can, as in [11], reduce the Lipschitz constants to a local supremum of $f'(c)$. Further, we define the terms

$$\alpha_{j,1} \equiv \gamma_{j,1}\kappa_{j,1} \tag{25}$$

and

$$\alpha_{j,2} \equiv \gamma_{j,2}\kappa_{j,2}. \tag{26}$$

By our assumptions (5) and (6) and the assumed monotonicity properties of h (h nondecreasing in its first argument, nonincreasing in its second), we have $\alpha_{j,1} \ge 0$, $\alpha_{j,2} \le 0$, and thus

$$\alpha_{j,1} - \alpha_{j,2} \ge 0. \tag{27}$$

For each j at each time $t^{n+\eta_l}$, we require the local CFL condition

$$1 - \alpha_{j,1}^{n+\eta_l} + \alpha_{j,2}^{n+\eta_l} \ge 0. \tag{28}$$

Here, we pause to discuss the relation between the terms (5)–(6) and the CFL condition (28). We require that

$$\alpha_{j,1}^{n+\eta_l} - \alpha_{j,2}^{n+\eta_l} \leq 1, \tag{29}$$

or

$$\gamma_{j,1}^{n+\eta_l} \kappa_{j,1}^{n+\eta_l} - \gamma_{j,2}^{n+\eta_l} \kappa_{j,2}^{n+\eta_l} \leq 1. \tag{30}$$

If the corrections are all zero, then each κ term is simply 1. However, as in Figure 1, our κ terms are only bounded by $1+\theta$. Consequently, in order to satisfy (28), we must possibly take a smaller time step than we would if we were not using the corrections. Equivalently, we could write our CFL condition as do Cockburn and Shu [2]:

$$\gamma_{j,1}^{n+\eta_l} - \gamma_{j,2}^{n+\eta_l} \leq \frac{1}{1+\theta}. \tag{31}$$

The L^∞ stability result of [11] depends upon showing that the operator which advances the solution forward is monotone. However, with the addition of the correction terms, the operator ceases to be monotone, and we must take a different approach. We begin by examining the basic global time stepping method (or $j \in \mathcal{C}^n \ \forall \ j$). The method is

$$c_j^{n+1} = c_j^n - \lambda_j^n \left[h\left(c_{j+\frac{1}{2},L}^n, c_{j+\frac{1}{2},R}^n \right) - h\left(c_{j-\frac{1}{2},L}^n, c_{j-\frac{1}{2},R}^n \right) \right]. \tag{32}$$

We can obtain L^∞ stability as follows. Add and subtract $h(c_{j+\frac{1}{2},L}^n, c_{j-\frac{1}{2},R}^n)$ in (32) and multiply and divide by the proper differences, and we have

$$c_j^{n+1} = c_j^n + \alpha_{j,1}^n \left(c_{j+1}^n - c_j^n \right) + \alpha_{j,2}^n \left(c_j^n - c_{j-1}^n \right), \tag{33}$$

or

$$c_j^{n+1} = \left(1 - \alpha_{j,1}^n + \alpha_{j,2}^n \right) c_j^n + \alpha_{j,1}^n c_{j+1}^n - \alpha_{j,2}^n c_{j-1}^n. \tag{34}$$

Then, by (27) and (28), we see that

$$\begin{aligned} |c_j^{n+1}| &\leq \left(1 - \alpha_{j,1}^n + \alpha_{j,2}^n \right) |c_j^n| + \alpha_{j,1}^n |c_{j+1}^n| - \alpha_{j,2}^n |c_{j-1}^n| \\ &\leq \sup_j |c_j^n|. \end{aligned} \tag{35}$$

By making similar arguments in the context of the local time stepping method, we have the following stability result.

PROPOSITION 2.1. *Assume* (5), (6), *and* (28) *hold, and the numerical flux* $h(u,v)$ *is nondecreasing in* u *and nonincreasing in* v*; then for* $n > 0$,

$$\sup_j |c_j^n| \leq \sup_j |c_j^0|,$$

where c_j^n *is defined by* (18)–(19).

Proof. The proof follows from looking separately at the cases in which $j \in \mathcal{C}^n$ and $j \notin \mathcal{C}^n$.

We begin with the case of $j \notin \mathcal{C}^n$. In this case, we may use the same technique as in the global time stepping case recursively to bound the solution at time $n+1$ in terms of the solution at time n.

$$
\begin{aligned}
|c_j^{n+\eta_l}| &= \left| c_j^{n+\eta_{l-1}} - \sigma_{l+1}\lambda_j^n \Delta_+ h\left(c_{j-\frac{1}{2},L}^{n+\eta_{l-1}}, c_{j-\frac{1}{2},R}^{n+\eta_{l-1}}\right)\right| \\
&= \left| c_j^{n+\eta_{l-1}} + \alpha_{j,1}^{n+\eta_{l-1}}\left(c_{j+1}^{n+\eta_{l-1}} - c_j^{n+\eta_{l-1}}\right)\right. \\
&\quad \left. + \alpha_{j,2}^{n+\eta_{l-1}}\left(c_j^{n+\eta_{l-1}} - c_{j-1}^{n+\eta_{l-1}}\right)\right| \\
&= |c_j^{n+\eta_{l-1}}|\left(1 - \alpha_{j,1}^{n+\eta_{l-1}} + \alpha_{j,2}^{n+\eta_{l-1}}\right) \\
&\quad + \alpha_{j,1}^{n+\eta_{l-1}}|c_{j+1}^{n+\eta_{l-1}}| - \alpha_{j,2}^{n+\eta_{l-1}}|c_{j-1}^{n+\eta_{l-1}}| \\
&\leq \max_{j-1\leq k\leq j+1} |c_k^{n+\eta_{l-1}}|.
\end{aligned} \tag{36}
$$

The α terms satisfy the same sign conditions as their global time stepping counterparts, so

$$
|c_j^{n+\eta_l}| \leq \max_{j-1\leq k\leq j+1} |c_k^{n+\eta_{l-1}}|. \tag{37}
$$

If $j^* \in [j-1, j+1]$ is the element on which the maximum occurs, and if $j^* \in \mathcal{C}^n$, then we are done for this element, as we have the solution bounded in terms of values at time n. If $j^* \notin \mathcal{C}^n$, then we apply this argument recursively.

Now, suppose that $j \in \mathcal{C}^n$. Using (18) in (19),

$$
\begin{aligned}
c_j^{n+1} &= c_j^n - \lambda_j^n \sum_{l=0}^{M-1} \sigma_{l+1}\Delta_+ h\left(c_{j-\frac{1}{2},L}^{n+\eta_l}, c_{j-\frac{1}{2},R}^{n+\eta_l}\right) \\
&= \sum_{l=0}^{M-1} \sigma_{l+1}\left(c_j^{n+\eta_{l-1}} - \lambda_j^n \Delta_+ h\left(c_{j-\frac{1}{2},L}^{n+\eta_l}, c_{j-\frac{1}{2},R}^{n+\eta_l}\right)\right) \\
&= \sum_{l=0}^{M-1} \sigma_{l+1}\left[c_j^{n+\eta_{l-1}} + \alpha_{j,1}^{n+\eta_{l-1}}\left(c_{j+1}^{n+\eta_{l-1}} - c_j^{n+\eta_{l-1}}\right)\right. \\
&\quad \left. + \alpha_{j,2}^{n+\eta_{l-1}}\left(c_j^{n+\eta_{l-1}} - c_{j-1}^{n+\eta_{l-1}}\right)\right] \\
&= \sum_{l=0}^{M-1} \sigma_{l+1}\left[c_j^{n+\eta_{l-1}}\left(1 - \alpha_{j,1}^{n+\eta_{l-1}} + \alpha_{j,2}^{n+\eta_{l-1}}\right)\right. \\
&\quad \left. + \alpha_{j,1}^{n+\eta_{l-1}} c_{j+1}^{n+\eta_{l-1}} - \alpha_{j,2}^{n+\eta_{l-1}} c_{j-1}^{n+\eta_{l-1}}\right].
\end{aligned} \tag{38}
$$

Again, using the monotonicity of h and (28), we have the sign conditions for the α terms, and

$$
\begin{aligned}
|c_j^{n+1}| &\leq \sum_{l=0}^{M-1} \sigma_{l+1}\left[|c_j^{n+\eta_{l-1}}|\left(1 - \alpha_{j,1}^{n+\eta_{l-1}} + \alpha_{j,2}^{n+\eta_{l-1}}\right)\right. \\
&\quad \left. + \alpha_{j,1}^{n+\eta_{l-1}}|c_{j+1}^{n+\eta_{l-1}}| - \alpha_{j,2}^{n+\eta_{l-1}}|c_{j-1}^{n+\eta_{l-1}}|\right] \\
&\leq \sum_{l=0}^{M-1} \sigma_{l+1} \max_{j-1\leq k\leq j+1} |c_k^{n+\eta_{l-1}}|.
\end{aligned} \tag{39}
$$

Now, each term $|c_k^{n+\eta_{l-1}}|$ may be bounded by prior information. If $k \in \mathcal{C}^n$, then $c_k^{n+\eta_{l-1}} = c_k^n$ and the bound is obvious. Otherwise, the above bound for $j \notin \mathcal{C}^n$ applies, and we are done. □

Note that while c_j^n remains constant in time on I_j, we might have to modify the correction terms on I_j at intermediate times if $j \in \mathcal{C}^n$ and $j-1 \notin \mathcal{C}^n$ or $j+1 \notin \mathcal{C}^n$ to avoid adding extrema to the solution.

Note also that for general problems, the CFL condition (28) for substeps after the first when $j \notin \mathcal{C}^n$ cannot be verified a priori without adjusting the time step at each substep. However, examining (25) and (26), we have uniform bounds on the κ terms by doing appropriate limiting. Moreover, with some knowledge of f, we can bound the γ terms locally depending on $f'(c)$ and the local mesh spacing. Thus we can estimate an upper bound on the α terms over regions of the domain, which can be used to set the local time steps over these regions. We have used this approach successfully in practice.

3. A second order time stepping scheme. We now turn to developing a local time stepping procedure based on a formally second order method. We are again interested in integrating the semidiscrete scheme

$$\frac{dc_j}{dt} = -\frac{1}{\Delta x_j}\Delta_+ h(c_{j-1/2,L}, c_{j-1/2,R}). \tag{40}$$

We first describe the basic method we are using, which is a second order Runge–Kutta scheme (in particular, Heun's method), shown to be TVD in [13, 6]. Then, we formulate the scheme for handling the interface between two regions with different time steps. We will assume the basic method holds for some distance away from the interfaces.

Heun's method for integrating (40) is given by

$$\overline{c}_j^{n+1} = c_j^n - \lambda_j^n \Delta_+ h(c_{j-1/2,L}^n, c_{j-1/2,R}^n), \tag{41}$$

$$w_j^{n+1} = \overline{c}_j^{n+1} - \lambda_j^n \Delta_+ h(\overline{c}_{j-1/2,L}^{n+1}, \overline{c}_{j-1/2,R}^{n+1}), \tag{42}$$

$$c_j^{n+1} = \frac{1}{2}\left(c_j^n + w_j^{n+1}\right). \tag{43}$$

Here $\overline{c}_{j+1/2,L}^{n+1} = \overline{c}_j^{n+1} + \tilde{\overline{c}}_j^{n+1}$, with an analogous definition for $\overline{c}_{j+1/2,R}^{n+1}$, where the corrections are computed as in (5)–(6). This method is nothing more than a convex combination of forward Euler steps and the initial value. Correspondingly, using the properties of the forward Euler method analyzed in the previous section, one easily obtains stability for this scheme.

PROPOSITION 3.1. *For the scheme* (41)–(43), *with corrections limited as in* (5)–(6) *and a CFL time-step constraint as in* (28), *we have*

$$\sup_j |c_j^n| \le \sup_j |c_j^0|.$$

For now, we are interested in computing at the interface on the space-time mesh shown in Figure 2. Thus, the time step in interval I_{j+1} is Δt and in I_j, $\Delta t/2$. For

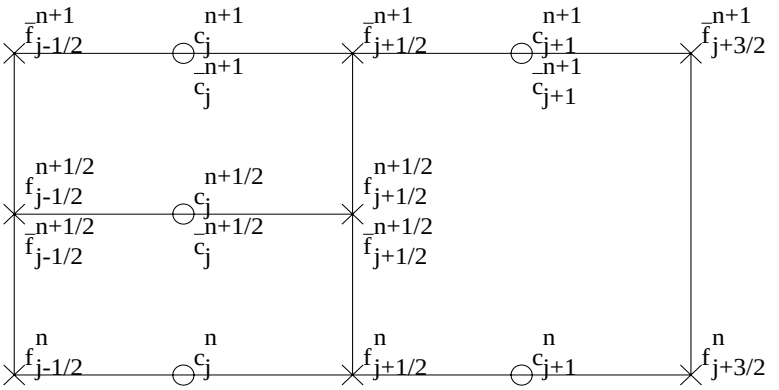


FIG. 2. *Local time stepping mesh.*

simplicity, we assume the time step is Δt for intervals to the right of $x_{j+1/2}$ and $\Delta t/2$ for intervals to the left of $x_{j+1/2}$. We will also assume for simplicity that $f(c) = uc$, where $u > 0$. We will describe the method and state conditions which the corrections $\tilde{c}$ should satisfy to ensure a maximum principle. We will then describe in more detail how the corrections may be constructed so as to satisfy these conditions. Finally, we will describe the method when the situation in Figure 2 holds and $u < 0$, and then discuss generalizations to a nonlinear flux function and to the case with M steps.

Thus, assuming $f(c) = uc$, where $u > 0$, to compute the solution in elements I_j and I_{j+1}, we use the following procedure.

Scheme I ($u > 0$).

(i) Compute corrections $\tilde{c}_{j-1}^n$, $\tilde{c}_j^n$, and $\tilde{c}_{j+1}^{n,1}$ by some means and limit them so that (44) is satisfied.

(ii) $\bar{c}_j^{n+\frac{1}{2}} = c_j^n - \frac{u\lambda_j^n}{2}\left[c_j^n + \tilde{c}_j^n - \left(c_{j-1}^n + \tilde{c}_{j-1}^n\right)\right]$.

(iii) Compute $\tilde{\bar{c}}_{j-1}^{n+1/2}$, $\tilde{\bar{c}}_j^{n+1/2}$, and $\tilde{c}_{j+1}^{n,2}$ so that (45) is satisfied.

(iv) $w_j^{n+\frac{1}{2}} = \bar{c}_j^{n+\frac{1}{2}} - \frac{u\lambda_j^n}{2}[\bar{c}_j^{n+\frac{1}{2}} + \tilde{\bar{c}}_j^{n+\frac{1}{2}} - (\bar{c}_{j-1}^{n+\frac{1}{2}} + \tilde{\bar{c}}_{j-1}^{n+\frac{1}{2}})]$.

(v) $c_j^{n+\frac{1}{2}} = \frac{1}{2}(c_j^n + w_j^{n+\frac{1}{2}})$.

(vi) Define $\tilde{c}_{j+1}^n \equiv \frac{1}{2}(\tilde{c}_{j+1}^{n,1} + \tilde{c}_{j+1}^{n,2})$.

(vii) $\bar{c}_{j+1}^{n+1} = c_{j+1}^{n+1/2} - u\lambda_{j+1}^n[c_{j+1}^n + \tilde{c}_{j+1}^n - \frac{1}{2}(c_j^n + \tilde{c}_j^n + \bar{c}_j^{n+\frac{1}{2}} + \tilde{\bar{c}}_j^{n+\frac{1}{2}})]$.

(viii) Compute $\tilde{c}_{j-1}^{n+\frac{1}{2}}$, $\tilde{c}_j^{n+\frac{1}{2}}$, and $\tilde{\bar{c}}_{j+1}^{n+1,1}$ so that (46) is satisfied.

(iv) $\bar{c}_j^{n+1} = c_j^n - \frac{u\lambda_j^n}{2}[c_j^{n+\frac{1}{2}} + \tilde{c}_j^{n+\frac{1}{2}} - (c_{j-1}^{n+\frac{1}{2}} + \tilde{c}_{j-1}^{n+\frac{1}{2}})]$.

(x) Compute $\tilde{\bar{c}}_{j-1}^{n+1}$, $\tilde{\bar{c}}_j^{n+1}$, and $\tilde{\bar{c}}_{j+1}^{n+1,2}$ so that (47) is satisfied.

(xi) $w_j^{n+1} = \bar{c}_j^{n+1} - \frac{u\lambda_j^n}{2}[\bar{c}_j^{n+1} + \tilde{\bar{c}}_j^{n+1} - (\bar{c}_{j-1}^{n+1} + \tilde{\bar{c}}_{j-1}^{n+1})]$.

(xii) $c_j^{n+1} = \frac{1}{2}(c_j^{n+\frac{1}{2}} + w_j^{n+1})$.

(xiii) Define $\tilde{\bar{c}}_{j+1}^{n+1} \equiv \frac{1}{2}(\tilde{\bar{c}}_{j+1}^{n,1} + \tilde{\bar{c}}_{j+1}^{n,2})$.

(xiv) $w_{j+1}^{n+1} = \bar{c}_{j+1}^n - u\lambda_{j+1}^n[\bar{c}_{j+1}^n + \tilde{\bar{c}}_{j+1}^n - \frac{1}{2}(c_j^{n+\frac{1}{2}} + \tilde{c}_j^{n+\frac{1}{2}} + \bar{c}_j^{n+1} + \tilde{\bar{c}}_j^{n+1})]$.

(xv) $c_{j+1}^{n+1} = \frac{1}{2}\left(c_{j+1}^n + w_{j+1}^{n+1}\right)$.

The corrections should satisfy the following conditions to guarantee a maximum principle:

$$-\theta \le -\frac{\tilde{c}_j^n - \tilde{c}_{j-1}^n}{c_j^n - c_{j-1}^n},\ -\frac{\tilde{c}_{j+1}^{n,1} - \tilde{c}_j^n}{c_{j+1}^n - c_j^n} \le 1, \tag{44}$$

$$-\theta \le -\frac{\tilde{\bar{c}}_j^{n+\frac{1}{2}} - \tilde{\bar{c}}_{j-1}^{n+\frac{1}{2}}}{\bar{c}_j^{n+\frac{1}{2}} - \bar{c}_{j-1}^{n+\frac{1}{2}}},\ -\frac{\tilde{c}_{j+1}^{n,2} - \tilde{\bar{c}}_j^{n+\frac{1}{2}}}{c_{j+1}^n - \bar{c}_j^{n+\frac{1}{2}}} \le 1, \tag{45}$$

$$-\theta \le -\frac{\tilde{c}_j^{n+\frac{1}{2}} - \tilde{c}_{j-1}^{n+\frac{1}{2}}}{c_j^{n+\frac{1}{2}} - c_{j-1}^{n+\frac{1}{2}}},\ -\frac{\tilde{\bar{c}}_{j+1}^{n+1,1} - \tilde{\bar{c}}_j^{n+\frac{1}{2}}}{\bar{c}_{j+1}^{n+1} - c_j^{n+1/2}} \le 1, \tag{46}$$

$$-\theta \le -\frac{\tilde{\bar{c}}_j^{n+1} - \tilde{\bar{c}}_{j-1}^{n+1}}{\bar{c}_j^{n+1} - \bar{c}_{j-1}^{n+1}},\ -\frac{\tilde{\bar{c}}_{j+1}^{n+1,2} - \tilde{\bar{c}}_j^{n+1}}{\bar{c}_{j+1}^{n+1} - \bar{c}_j^{n+1}} \le 1. \tag{47}$$

3.1. A maximum principle for the second order method. In this section, we derive a maximum principle for Scheme I above. The maximum principle argument allows for the method to have a time step on one cell that would violate the CFL condition on the adjacent cell. In particular, we assume

$$\frac{u\Delta t^n}{2\Delta x_j} \le \frac{1}{1+\theta} \tag{48}$$

and

$$\frac{u\Delta t^n}{\Delta x_{j+1}} \le \frac{1}{1+\theta}, \tag{49}$$

which is the usual CFL condition for grid blocks I_j and I_{j+1}.

Away from the interface, where the time step among adjacent cells is the same, the stability of the method is easily demonstrated. We summarize the result in the following proposition.

PROPOSITION 3.2. *Assume a time step of size Δt^n for all elements to the right of $x_{j+1/2}$ and size $\Delta t^n/2$ for all elements to the left of $x_{j+1/2}$, where Δt^n satisfies* (48) *and* (49). *Then, for Scheme* I *above with $f(c) = uc$ where $u > 0$, with corrections satisfying* (44)–(47) *on I_j and I_{j+1} and* (5)–(6) *elsewhere, we have*

$$\sup_j |c_j^n| \le \sup_j |c_j^0|. \tag{50}$$

Proof. For all blocks except I_j and I_{j+1}, the bound $|c_i^{n+1}| \le \sup_j |c_j^n|$ follows from the arguments used to prove Propositions 2.1 and 3.1. Furthermore, the only difference between the computation on I_j and the standard method is that there are somewhat different restrictions on the corrections, namely, they should satisfy (44)–(47). Thus, one can easily show that $|\bar{c}_j^{n+\frac{1}{2}}|$, $|w_j^{n+\frac{1}{2}}|$, $|c_j^{n+\frac{1}{2}}|$, $|\bar{c}_j^{n+1}|$, $|w_j^{n+1}|$ and hence $|c_j^{n+1}|$ are all bounded above by $\sup_j |c_j^n|$.

The solution on I_{j+1} must be more carefully analyzed. We will first show a maximum principle for $|\overline{c}_{j+1}^{n+1}|$. Then, this will be used to show maximum principles for $|w_{j+1}^{n+1}|$ and $|c_{j+1}^{n+1}|$.

Consider $\overline{c}_{j+1}^{n+1}$. The argument will proceed by the standard technique of choosing the CFL condition and using the restrictions on the corrections to guarantee that $\overline{c}_{j+1}^{n+1}$ is a convex combination of previous values. We define

$$\kappa_1 = 1 + \frac{\tilde{c}_{j+1}^{n,1} - \tilde{c}_j^n}{c_{j+1}^n - c_j^n} \tag{51}$$

and

$$\kappa_2 = 1 + \frac{\tilde{c}_{j+1}^{n,2} - \tilde{\overline{c}}_j^{n+\frac{1}{2}}}{c_{j+1}^n - \overline{c}_j^{n+\frac{1}{2}}}. \tag{52}$$

We can write the computation of $\overline{c}_{j+1}^n$ as

(53)

$$\begin{aligned}
\overline{c}_{j+1}^{n+1} &= c_{j+1}^n - \frac{u\Delta t^n}{\Delta x_{j+1}} \left[\left(c_{j+1}^n + \tilde{c}_{j+1}^n \right) - \frac{1}{2} \left(\left(c_j^n + \tilde{c}_j^n \right) - \left(\overline{c}_j^{n+\frac{1}{2}} + \tilde{\overline{c}}_j^{n+\frac{1}{2}} \right) \right) \right] \\
&= c_{j+1}^n - \frac{u\Delta t^n}{2\Delta x_{j+1}} \left[c_{j+1}^n + \tilde{c}_{j+1}^{n,1} - \left(c_j^n + \tilde{c}_j^n \right) + c_{j+1}^n + \tilde{c}_{j+1}^{n,2} - \left(\overline{c}_j^{n+\frac{1}{2}} + \tilde{\overline{c}}_j^{n+\frac{1}{2}} \right) \right] \\
&= c_{j+1}^n - \frac{u\Delta t^n}{2\Delta x_{j+1}} \left[\kappa_1 \left(c_{j+1}^n - c_j^n \right) + \kappa_2 \left(c_{j+1}^n - \overline{c}_j^{n+\frac{1}{2}} \right) \right] \\
&= c_{j+1}^n \left[1 - \frac{u\Delta t^n}{2\Delta x_{j+1}} \left(\kappa_1 + \kappa_2 \right) \right] + \frac{u\Delta t^n}{2\Delta x_{j+1}} \kappa_1 c_j^n + \frac{u\Delta t^n}{2\Delta x_{j+1}} \kappa_2 \overline{c}_j^{n+\frac{1}{2}}.
\end{aligned}$$

We note that the coefficients of the terms sum to 1. By (44) and (45), $\kappa_i \geq 0$, $i = 1, 2$, and

$$1 - \frac{u\Delta t^n}{2\Delta x_{j+1}} \left(\kappa_1 + \kappa_2 \right) \geq 0 \tag{54}$$

if the time step is chosen such that (49) is satisfied.

By the above-stated bound on $|\overline{c}_j^{n+\frac{1}{2}}|$, we can take the absolute value of each side of (53) and obtain

$$\begin{aligned}
|\overline{c}_{j+1}^{n+1}| &= |c_{j+1}^n| \left[1 - \frac{u\Delta t^n}{2\Delta x_{j+1}} \left(\kappa_1 + \kappa_2 \right) \right] + \frac{u\Delta t^n}{2\Delta x_{j+1}} \kappa_1 |c_j^n| + \frac{u\Delta t^n}{2\Delta x_{j+1}} \kappa_2 |\overline{c}_j^{n+\frac{1}{2}}| \\
&\leq \sup_j |c_j^n|.
\end{aligned} \tag{55}$$

Next, consider w_{j+1}^{n+1}. We define two new terms:

$$\kappa_3 = 1 + \frac{\tilde{\overline{c}}_{j+1}^{n+1,2} - \tilde{c}_j^{n+\frac{1}{2}}}{\overline{c}_{j+1}^{n+1} - c_j^{n+\frac{1}{2}}} \tag{56}$$

and

$$\kappa_4 = 1 + \frac{\tilde{\overline{c}}_{j+1}^{n+1,1} - \tilde{\overline{c}}_j^{n+1}}{\overline{c}_{j+1}^{n+1} - \overline{c}_j^{n+1}}. \tag{57}$$

The method can be written as

$$
\begin{aligned}
(58)\qquad w_{j+1}^{n+1} &= \bar{c}_{j+1}^{n+1} - \frac{u\Delta t^n}{\Delta x_{j+1}}\left[\bar{c}_{j+1}^{n+1} + \tilde{\bar{c}}_{j+1}^{n+1} - \frac{1}{2}\left(c_j^{n+\frac{1}{2}} + \tilde{c}_j^{n+\frac{1}{2}} + \bar{c}_j^{n+1} + \tilde{\bar{c}}_j^{n+1}\right)\right] \\
&= \bar{c}_{j+1}^{n+1} - \frac{u\Delta t^n}{2\Delta x_{j+1}}\left[\bar{c}_{j+1}^{n+1} + \tilde{\bar{c}}_{j+1}^{n+1,1}\right. \\
&\qquad \left. - \left(c_j^{n+\frac{1}{2}} + \tilde{c}_j^{n+\frac{1}{2}}\right) + \bar{c}_{j+1}^{n+1} + \tilde{\bar{c}}_{j+1}^{n+1,2} - \left(\bar{c}_j^{n+1} + \tilde{\bar{c}}_j^{n+1}\right)\right] \\
&= \bar{c}_{j+1}^{n+1} - \frac{u\Delta t}{2\Delta x_{j+1}}\left[\kappa_3\left(\bar{c}_{j+1}^{n+1} - c_j^{n+\frac{1}{2}}\right) + \kappa_4\left(\bar{c}_{j+1}^{n+1} - \bar{c}_j^{n+1}\right)\right] \\
&= \bar{c}_{j+1}^{n+1}\left[1 - \frac{u\Delta t^n}{2\Delta x_{j+1}}\left(\kappa_3 + \kappa_4\right)\right] + \frac{u\Delta t^n}{2\Delta x_{j+1}}\kappa_3 c_j^{n+\frac{1}{2}} + \frac{u\Delta t^n}{2\Delta x_{j+1}}\kappa_4 \bar{c}_j^{n+1}.
\end{aligned}
$$

As before, the coefficients sum to 1. By a similar argument, (46) and (47) give the nonnegativity of each κ term, and we have the same CFL restriction (49) as above. Taking the absolute value of each side of (58), we see that

$$
|w_{j+1}^{n+1}| \le \max\left(|\bar{c}_{j+1}^{n+1}|, |c_j^{n+\frac{1}{2}}|, |\bar{c}_j^{n+1}|\right),
$$

and each of these terms is bounded by the solution at time n. Finally

$$
\begin{aligned}
|c_{j+1}^{n+1}| &= \left|\frac{1}{2}\left(c_{j+1}^n + w_{j+1}^{n+1}\right)\right| \\
&\le \frac{1}{2}|c_{j+1}^n| + \frac{1}{2}|w_{j+1}^{n+1}| \\
&\le \sup_j |c_j^n|. \qquad \square
\end{aligned}
$$

To summarize, our method is conservative (the flux used at the right edge of element I_j is equal to that used at the left edge of I_{j+1}). We have demonstrated stability subject only to local CFL restrictions. Next we discuss the limiting of the corrections so that the conditions (44)–(47) are satisfied. Later, we will present numerical evidence demonstrating that local time stepping does not appear to degrade the accuracy of the method.

3.2. Computing and limiting the corrections. We discuss in slightly more detail the computation of the $\tilde{c}$ terms. We assume as before that θ is chosen so that $0 \le \theta \le 1$.

First, assume $\tilde{c}_j^n$, $\tilde{c}_{j-1}^n$, and $\tilde{c}_{j+1}^{n,1}$ are computed so that (44) is satisfied, say, using (15). We then set

$$
(59)\qquad \tilde{c}_{j+1}^{n,2} = \begin{cases} \tilde{c}_{j+1}^{n,1} & \text{if } 2|\tilde{c}_{j+1}^{n,1}| \le \theta|c_{j+1}^n - \bar{c}_j^{n+1/2}|, \\ 0 & \text{otherwise.} \end{cases}
$$

In a similar way, corrections $\tilde{\bar{c}}_j^{n+1/2}$ and $\tilde{\bar{c}}_{j-1}^{n+1/2}$ are computed so that the first inequality in (45) is satisfied, and $\tilde{\bar{c}}_j^{n+1/2}$ is limited so that

$$
(60)\qquad 2|\tilde{\bar{c}}_j^{n+1/2}| \le \theta|c_{j+1}^n - \bar{c}_j^{n+1/2}|.
$$

Hence, one can show

$$
(61)\qquad 0 \le \kappa_1, \kappa_2 \le 1 + \theta.
$$

Proceeding, $\tilde{c}_j^{n+1/2}$ and $\tilde{c}_{j-1}^{n+1/2}$ are computed so that the first inequality in (46) is satisfied, and so that

$$2|\tilde{c}_j^{n+1/2}| \le \theta|\overline{c}_{j+1}^{n+1} - c_j^{n+1/2}|; \tag{62}$$

$\tilde{\overline{c}}_{j+1}^{n+1,1}$ is computed so that it also satisfies

$$2|\tilde{c}_{j+1}^{n+1,1}| \le \theta|\overline{c}_{j+1}^{n+1} - c_j^{n+1/2}|. \tag{63}$$

Similarly, $\tilde{\overline{c}}_{j-1}^{n+1}$, $\tilde{\overline{c}}_j^{n+1}$, and $\tilde{\overline{c}}_{j+1}^{n+1,2}$ are computed so that (47) is satisfied. Thus

$$0 \le \kappa_3, \kappa_4 \le 1 + \theta. \tag{64}$$

3.3. The case $u < 0$. Consider Figure 2 again, now with $f(c) = uc$ and $u < 0$. This case is analogous to the situation in which $u > 0$ and the picture in Figure 2 is reversed, namely, the local time steps are to the right of the interface. In this case, to compute the solution in elements I_j and I_{j+1}, we use the following procedure.

Scheme II ($u < 0$).

(i) Compute corrections $\tilde{c}_j^n$, $\tilde{c}_{j+1}^{n,1}$, and $\tilde{c}_{j+2}^n$ so that (65) is satisfied.

(ii) $\overline{c}_j^{n+\frac{1}{2}} = c_j^n - \frac{u\lambda_j^n}{2}[c_{j+1}^n - \tilde{c}_{j+1}^{n,1} - (c_j^n - \tilde{c}_j^n)]$.

(iii) Compute $\tilde{\overline{c}}_j^{n+1/2}$ and $\tilde{c}_{j+1}^{n,2}$ so that (66) is satisfied.

(iv) $w_j^{n+\frac{1}{2}} = \overline{c}_j^{n+\frac{1}{2}} - \frac{u\lambda_j^n}{2}[c_{j+1}^n - \tilde{c}_{j+1}^{n,2} - (\overline{c}_j^{n+\frac{1}{2}} - \tilde{\overline{c}}_j^{n+\frac{1}{2}})]$.

(v) $c_j^{n+\frac{1}{2}} = \frac{1}{2}(c_j^n + w_j^{n+\frac{1}{2}})$.

(vi) Define $\tilde{c}_{j+1}^n \equiv \frac{1}{2}(\tilde{c}_{j+1}^{n,1} + \tilde{c}_{j+1}^{n,2})$.

(vii) $\overline{c}_{j+1}^{n+1} = c_{j+1}^n - u\lambda_{j+1}^n\left[c_{j+2}^n - \tilde{c}_{j+2}^n - \left(c_{j+1}^n - \tilde{c}_{j+1}^n\right)\right]$.

(viii) Compute $\tilde{c}_j^{n+\frac{1}{2}}$, $\tilde{\overline{c}}_{j+1}^{n+1,1}$, and $\tilde{\overline{c}}_{j+2}^{n+1}$ so that (67) is satisfied.

(ix) $\overline{c}_j^{n+1} = c_j^{n+1/2} - \frac{u\lambda_j^n}{2}[\overline{c}_{j+1}^{n+1} - \tilde{\overline{c}}_{j+1}^{n+1,1} - (c_j^{n+\frac{1}{2}} - \tilde{c}_j^{n+\frac{1}{2}})]$.

(x) Compute $\tilde{\overline{c}}_j^{n+1}$ and $\tilde{\overline{c}}_{j+1}^{n+1,2}$ to satisfy (68).

(xi) $w_j^{n+1} = \overline{c}_j^{n+1} - \frac{u\lambda_j^n}{2}[\overline{c}_{j+1}^{n+1} - \tilde{\overline{c}}_{j+1}^{n+1,2} - (\overline{c}_j^{n+1} - \tilde{\overline{c}}_j^{n+1})]$.

(xii) $c_j^{n+1} = \frac{1}{2}(c_j^{n+\frac{1}{2}} + w_j^{n+1})$.

(xiii) Define $\tilde{\overline{c}}_{j+1}^{n+1} \equiv \frac{1}{2}(\tilde{\overline{c}}_{j+1}^{n+1,1} + \tilde{\overline{c}}_{j+1}^{n+1,2})$.

(xiv) $w_{j+1}^{n+1} = \overline{c}_{j+1}^n - u\lambda_{j+1}^n[\overline{c}_{j+2}^{n+1} - \tilde{\overline{c}}_{j+2}^{n+1} - (\overline{c}_{j+1}^{n+1} - \tilde{\overline{c}}_j^{n+1})]$.

(xv) $c_{j+1}^{n+1} = \frac{1}{2}\left(c_{j+1}^n + w_{j+1}^{n+1}\right)$.

The conditions on the corrections in this case are

$$-\theta \le \frac{\tilde{c}_{j+1}^{n,1} - \tilde{c}_j^n}{c_{j+1}^n - c_j^n}, \frac{\tilde{c}_{j+2}^n - \tilde{c}_{j+1}^{n,1}}{c_{j+2}^n - c_{j+1}^n} \le 1, \tag{65}$$

$$-\theta \le \frac{\tilde{c}_{j+1}^{n,2} - \tilde{\overline{c}}_j^{n+1/2}}{c_{j+1}^n - \overline{c}_j^{n+1/2}}, \frac{\tilde{c}_{j+2}^n - \tilde{c}_{j+1}^{n,2}}{c_{j+2}^n - c_{j+1}^n} \le 1, \tag{66}$$

$$-\theta \le \frac{\tilde{\bar{c}}_{j+1}^{n+1,1} - \tilde{c}_j^{n+1/2}}{\bar{c}_{j+1}^{n+1} - c_j^{n+1/2}},\ \frac{\tilde{\bar{c}}_{j+2}^{n+1} - \tilde{\bar{c}}_{j+1}^{n+1,1}}{\bar{c}_{j+2}^{n+1} - \bar{c}_{j+1}^{n+1}} \le 1, \tag{67}$$

$$-\theta \le \frac{\tilde{\bar{c}}_{j+2}^{n+1} - \tilde{\bar{c}}_{j+1}^{n+1,2}}{\bar{c}_{j+2}^{n+1} - \bar{c}_{j+1}^{n+1}},\ \frac{\tilde{\bar{c}}_{j+1}^{n+1,2} - \tilde{\bar{c}}_j^{n+1}}{\bar{c}_{j+1}^{n+1} - \bar{c}_j^{n+1}} \le 1. \tag{68}$$

These conditions lead to the following result. The proof is left to the reader.

PROPOSITION 3.3. *Assume a time step of size Δt^n for all elements to the right of $x_{j+1/2}$ and size $\Delta t^n/2$ for all elements to the left of $x_{j+1/2}$, where Δt^n satisfies* (48) *and* (49) *(with u replaced by $-u$). Then, for Scheme* II *above with $f(c) = uc$ where $u < 0$, with corrections satisfying* (65)–(68) *on I_j and I_{j+1} and* (5)–(6) *elsewhere, we have*

$$\sup_j |c_j^n| \le \sup_j |c_j^0|. \tag{69}$$

3.4. Extension to a nonlinear flux function. The extension of the methods above to a more general numerical flux h essentially combines the ideas of Schemes I and II above. In particular, the method is as follows.

Scheme III (general flux function).

(i) Compute corrections $\tilde{c}_{j-1}^n$, $\tilde{c}_j^n$, $\tilde{c}_{j+1}^{n,1}$, and $\tilde{c}_{j+2}^n$ so that (44) and (65) are satisfied.

(ii) $\bar{c}_j^{n+\frac{1}{2}} = c_j^n - \frac{\lambda_j^n}{2}[h(c_j^n + \tilde{c}_j^n, c_{j+1} - \tilde{c}_{j+1}^{n,1}) - h(c_{j-1}^n + \tilde{c}_{j-1}^n, c_j^n - \tilde{c}_j^n)]$.

(iii) Compute $\tilde{\bar{c}}_{j-1}^{n+1/2}$, $\tilde{\bar{c}}_j^{n+1/2}$, and $\tilde{c}_{j+1}^{n,2}$ so that (45) and (66) are satisfied.

(iv) Compute $w_j^{n+\frac{1}{2}}$ by

$$w_j^{n+\frac{1}{2}} = \bar{c}_j^{n+\frac{1}{2}} - \frac{\lambda_j^n}{2}\left[h(\bar{c}_j^{n+\frac{1}{2}} + \tilde{\bar{c}}_j^{n+\frac{1}{2}}, c_{j+1}^n - \tilde{c}_{j+1}^{n,2}) - h(\bar{c}_{j-1}^{n+\frac{1}{2}} + \tilde{\bar{c}}_{j-1}^{n+\frac{1}{2}}, \bar{c}_j^{n+\frac{1}{2}} - \tilde{\bar{c}}_j^{n+\frac{1}{2}})\right].$$

(v) $c_j^{n+\frac{1}{2}} = \frac{1}{2}(c_j^n + w_j^{n+\frac{1}{2}})$.

(vi) Define $\tilde{c}_{j+1}^n \equiv \frac{1}{2}(\tilde{c}_{j+1}^{n,1} + \tilde{c}_{j+1}^{n,2})$.

(vii) Compute $\bar{c}_{j+1}^{n+1}$ by

$$\bar{c}_{j+1}^{n+1} = c_{j+1}^n - \lambda_{j+1}^n \Bigg[h(c_{j+1}^n + \tilde{c}_{j+1}^n, c_{j+2}^n - \tilde{c}_{j+2}^n) - \frac{1}{2}(h(c_j^n + \tilde{c}_j^n, c_{j+1}^n - \tilde{c}_{j+1}^{n,1}) + h(\bar{c}_j^{n+\frac{1}{2}} + \tilde{\bar{c}}_j^{n+\frac{1}{2}}, c_{j+1}^n - \tilde{c}_{j+1}^{n,2}))\Bigg].$$

(viii) Compute $\tilde{c}_{j-1}^{n+\frac{1}{2}}$, $\tilde{c}_j^{n+\frac{1}{2}}$, $\tilde{\bar{c}}_{j+1}^{n+1,1}$, and $\tilde{\bar{c}}_{j+2}^{n+1}$ so that (46) and (67) are satisfied.

(ix) Compute $\bar{c}_j^{n+1}$ by

$$\bar{c}_j^{n+1} = c_j^{n+\frac{1}{2}} - \frac{\lambda_j^n}{2}\left[h(c_j^{n+\frac{1}{2}} + \tilde{c}_j^{n+\frac{1}{2}}, \bar{c}_{j+1}^{n+1} - \tilde{\bar{c}}_{j+1}^{n+1,1}) - h(c_{j-1}^{n+\frac{1}{2}} + \tilde{c}_{j-1}^{n+\frac{1}{2}}, c_j^{n+\frac{1}{2}} - \tilde{c}_j^{n+\frac{1}{2}})\right].$$

(x) Compute $\tilde{\bar{c}}_{j-1}^{n+1}$, $\tilde{\bar{c}}_j^{n+1}$, and $\tilde{\bar{c}}_{j+1}^{n+1,2}$ so that (47) and (68) are satisfied.

(xi) Compute w_j^{n+1} by

$$w_j^{n+1} = \bar{c}_j^{n+1} - \frac{\lambda_j^n}{2}\left[h(\bar{c}_j^{n+1} + \tilde{\bar{c}}_j^{n+1}, \bar{c}_{j+1}^{n+1} - \tilde{\bar{c}}_{j+1}^{n+1,2}) - h(\bar{c}_{j-1}^{n+1} + \tilde{\bar{c}}_{j-1}^{n+1}, \bar{c}_j^{n+1} - \tilde{\bar{c}}_j^{n+1})\right].$$

(xii) $c_j^{n+1} = \frac{1}{2}(c_j^{n+\frac{1}{2}} + w_j^{n+1})$.

(xiii) Define $\tilde{\bar{c}}_{j+1}^{n+1} \equiv \frac{1}{2}(\tilde{\bar{c}}_{j+1}^{n,1} + \tilde{\bar{c}}_{j+1}^{n,2})$.
(xiv)

$$w_{j+1}^{n+1} = \bar{c}_{j+1}^{n+1} - \frac{\Delta t^n}{\Delta x_{j+1}} \left[h(\bar{c}_{j+1}^{n+1} + \tilde{\bar{c}}_{j+1}^{n+1}, \bar{c}_{j+2}^{n+1} - \tilde{\bar{c}}_{j+2}^{n+1}) \right.$$
$$\left. -\frac{1}{2}\left(h(c_j^{n+\frac{1}{2}} + \tilde{c}_j^{n+\frac{1}{2}}, \bar{c}_{j+1}^{n+1} - \tilde{\bar{c}}_{j+1}^{n+1,1}) + h(\bar{c}_j^{n+1} + \tilde{\bar{c}}_j^{n+1}, \bar{c}_{j+1}^{n+1} - \tilde{\bar{c}}_{j+1}^{n+1,2}) \right) \right].$$

(xv) $c_{j+1}^{n+1} = \frac{1}{2}\left(c_{j+1}^n + w_{j+1}^{n+1}\right)$.

3.5. Extension to M steps. Now suppose we take M steps in grid block I_j to one step in block I_{j+1}. The easier extension of the method described above is for the case where M is even, although an extension to M odd can also be made.

Assume M is even. When $M = 2$, note that to compute $\bar{c}_{j+1}^{n+1}$ we used the values c_j^n up through $\bar{c}_j^{n+1/2}$. The same is true in general. We compute $\bar{c}_j^{n+1/2}$ by taking $M/2$ steps in block I_j. The average of the fluxes along edge $j + 1/2$ over these steps is used to compute $\bar{c}_{j+1}^{n+1}$, just as above. The computation of w_{j+1}^{n+1} is analogous. We take $M/2$ steps to compute $\bar{c}_{j+1}^{n+1}$, starting with $c_j^{n+1/2}$, and the average of the fluxes along edge $j + 1/2$ is used to compute w_{j+1}^{n+1}.

The maximum principle argument can be carried through with appropriate limiting of the corrections. Similar to the case $M = 2$, at each substep i, $i = 1, \ldots, M/2$, we compute a correction $\tilde{c}_{j+1}^{n,i}$ in block I_{j+1} so that the analogues of (44), (45), (65), and (66) are satisfied. The final correction $\tilde{c}_{j+1}^n$ is the average of all of these substep corrections. Similarly, once $\bar{c}_{j+1}^{n+1}$ is computed, corrections $\tilde{\bar{c}}_{j+1}^{n+1,i}$ are computed at each substep so that the analogues of (46), (47), (67), and (68) are satisfied, and $\tilde{\bar{c}}_{j+1}^{n+1}$ is computed by averaging these corrections.

4. Numerical results. Here we present results examining the accuracy and stability of the method in section 3.

4.1. Linear example: Smooth problem. We will first show how local time stepping affects the errors and stability in the case of a smooth linear problem. Consider initial and boundary conditions such that the true solution is $\sin(\pi(x-t))$. The space-time domain is $[0,1]^2$. We have the following cases:

1. Uniform mesh of width Δx, global time step $\Delta t = \frac{2\Delta x}{3}$.
2. Region $[0, 0.5]$ refined to $\frac{\Delta x}{2}$, time step globally refined to $\Delta t = \frac{\Delta x}{3}$.
3. Region $[0, 0.5]$ refined to $\frac{\Delta x}{2}$, time step on $[0, 0.5]$ reduced (locally) by a factor of 2.

Examining Tables 1, 2, and 3, we see that the rate of convergence in L^2 in all three cases is around 1.5, which is to be expected for a *minmod*-limited method. Conceivably, the local time stepping could incur an $\mathcal{O}(h^{\frac{1}{2}})$ error which a constant time step would not incur. Thus, we turned off the limiter on the slopes and saw that the local stepping does not degenerate the order of accuracy, as we see second order convergence in Tables 4 and 5.

4.2. Linear example: Rough problem. Now that we have seen that the local time stepping scheme does not degenerate the accuracy of the approximation, we will show that the method is indeed stable with a local CFL condition. Consider the linear model case of $f(c) = c$ with $c(x, 0) = 1$ for $x < 0$ and $c(x, 0) = 0$ for $x > 0$. At $t = 0.5$, the true solution is a front at $x = 0.5$. Consider the following situations:

TABLE 1
Case 1: *Uniform mesh, global time step.*

Δx	L^2 error	L^1 error
.2500D+00	.1447D+00	.1205D+00
.1250D+00	.4995D-01	.3827D-01
.6250D-01	.1821D-01	.1422D-01
.3125D-01	.6033D-02	.4248D-02
.1562D-01	.1955D-02	.1287D-02
.7812D-02	.6291D-03	.3714D-03
rate:	1.56	1.66

TABLE 2
Case 2: *Local refinement, global time step.*

Δx	L^2 error	L^1 error
.2500D+00	.1145D+00	.9424D-01
.1250D+00	.4116D-01	.2646D-01
.6250D-01	.1461D-01	.8774D-02
.3125D-01	.4869D-02	.2550D-02
.1562D-01	.1586D-02	.7236D-03
.7812D-02	.5112D-03	.1965D-03
rate:	1.56	1.77

TABLE 3
Case 3: *Local refinement, local time step.*

Δx	L^2 error	L^1 error
.2500D+00	.1306D+00	.1030D+00
.1250D+00	.4304D-01	.2641D-01
.6250D-01	.1620D-01	.1018D-01
.3125D-01	.5981D-02	.3328D-02
.1562D-01	.2128D-02	.1052D-02
.7812D-02	.7441D-03	.3196D-03
rate:	1.48	1.63

TABLE 4
Uniform mesh and time step on $[0, 1]$, *unlimited slopes.*

Δx	L^2 error	L^1 error
.2500D+00	.1380D+00	.1317D+00
.1250D+00	.4331D-01	.3371D-01
.6250D-01	.1147D-01	.9483D-02
.3125D-01	.2869D-02	.2417D-02
.1562D-01	.7079D-03	.6061D-03
.7812D-02	.1746D-03	.1509D-03
rate:	1.94	1.95

TABLE 5
Uniform mesh and local time step on $[0, 1]$, *unlimited slopes.*

Δx	L^2 error	L^1 error
.2500D+00	.1290D+00	.1113D+00
.1250D+00	.3908D-01	.2951D-01
.6250D-01	.9640D-02	.7284D-02
.3125D-01	.2235D-02	.1785D-02
.1562D-01	.5294D-03	.4355D-03
.7812D-02	.1281D-03	.1078D-03
rate:	2.02	2.01

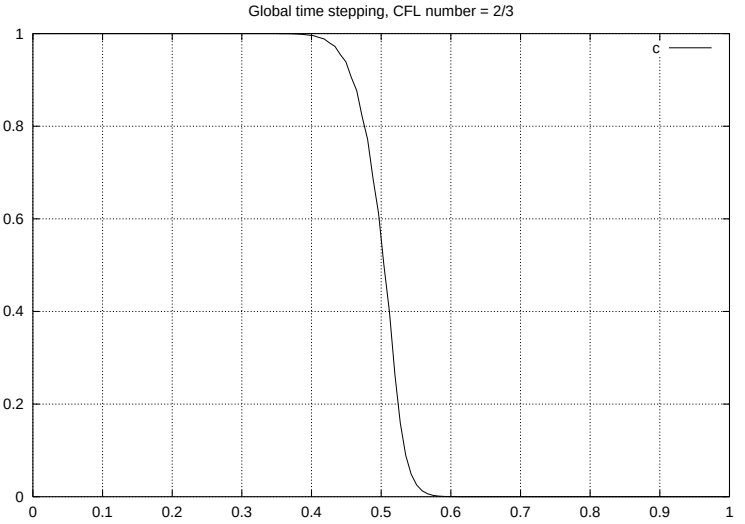


FIG. 3. *Global time stepping, model advective front.*

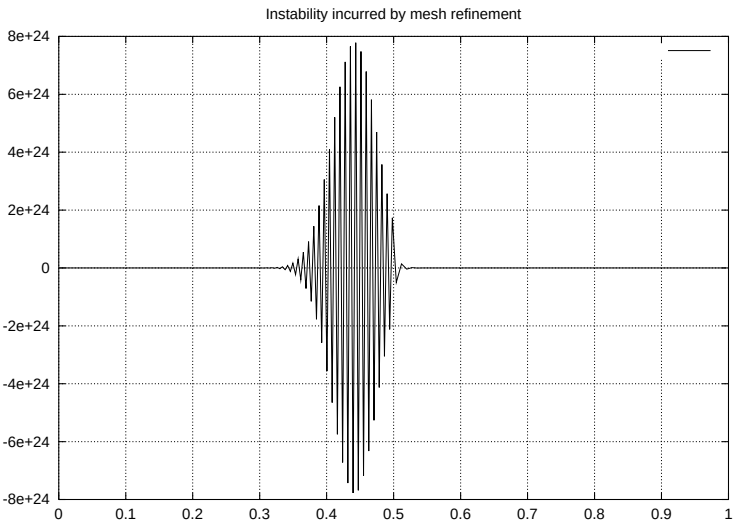


FIG. 4. *Local mesh refinement creates instability.*

1. Global uniform mesh, $\Delta x = \frac{1}{128}$, global CFL number of $\frac{2}{3}$.
2. Refine the mesh on $[0, 0.5]$ by a factor of 2, keep the same global time step. (CFL number in first half of the domain is $\frac{4}{3}$.)
3. Refine the time step on $[0, 0.5]$ by a factor of 2, giving a *local* CFL number of $\frac{2}{3}$ everywhere.

Observe that the front is propagated stably in the first case in Figure 3 but when the CFL condition is violated in the second case, we incur massive instability (Figure 4). However, the local time stepping method, with its local CFL condition, gives a stable and accurate approximation to the front in Figure 5.

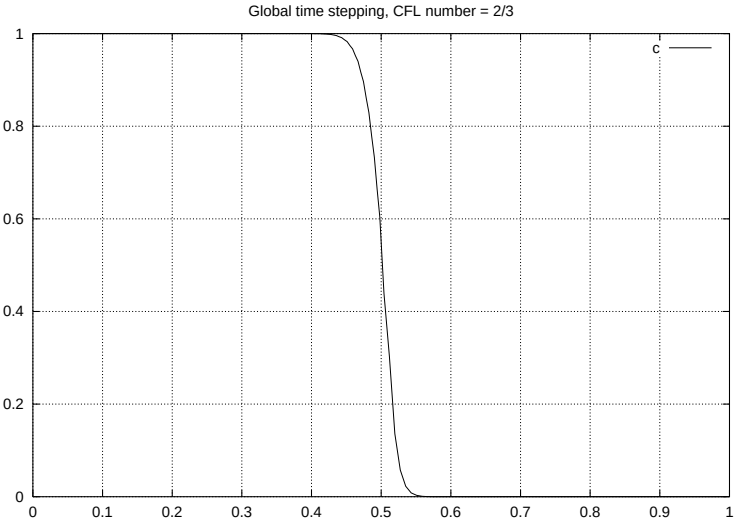


FIG. 5. *Local time stepping restores stability.*

4.3. Nonlinear example: Buckley–Leverett. Next, we examine a case where the flux is nonlinear. The Buckley–Leverett problem, given by

$$f(c) = \frac{c^2}{c^2 + a(1-c)^2}, \tag{70}$$

is a standard test problem arising in two-phase flow in porous media. This flux function is Lipschitz but nonconvex. In the case of $a = 0.25$, it is known from the Rankine–Hugoniot condition that the front is a rarefaction down to the point where $c = 0.44$, and then the solution jumps to $c = 0$. See, for example, [10] for a discussion. The Rankine–Hugoniot condition also gives that this front propagates at a velocity of 1.62. We performed a series of numerical experiments on a uniform mesh. First, we used global time stepping to verify that the code put the shock in the right location with the correct jump. Then, we refined the time step in the first half of the domain repeatedly in order to verify that the local time stepping did not alter the method's shock-capturing abilities. Each of these cases was run with a mesh spacing of $\Delta x = \frac{1}{128}$ and a main time step of $\Delta t = \frac{1}{160}$ to time $t = 0.5$. The true front should be at $x = 0.81$. These fronts were all near, though diffused, and further spatial refinements gave sharper fronts at the right spot. Figure 6 shows these results.

Further, we show that we have stability only subject to a local CFL constraint. Again, we consider three cases, with $\Delta x_{\max} = \frac{1}{128}$ and $\Delta t_{\max} = \frac{1}{160}$. In each case, the mesh width is $\frac{1}{2}\Delta x_{\max}$ on $[0, 0.5]$ and $\Delta x_{\max}$ elsewhere.

1. Global time step $\Delta t_{\max}$.
2. Global time step $\frac{1}{2}\Delta t_{\max}$.
3. Local time stepping.

As seen in Figure 7, the large time step in the presence of the local refinement has caused some mild instability with over- and undershoot. Due to the large velocity, this mild oscillation has propagated out of the first part of the domain and into the second. However, when the time step is refined either globally or locally in the presence of the mesh refinement, the front is well approximated.

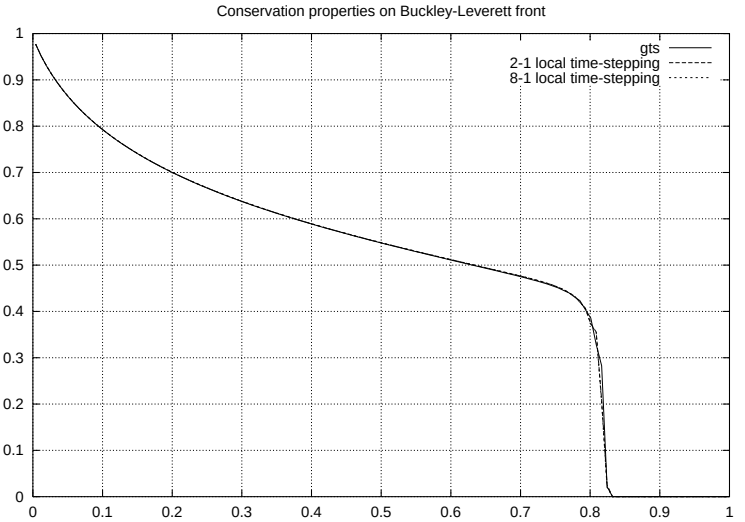


FIG. 6. *Local time stepping approximations to Buckley–Leverett front at* $x = 0.81$.

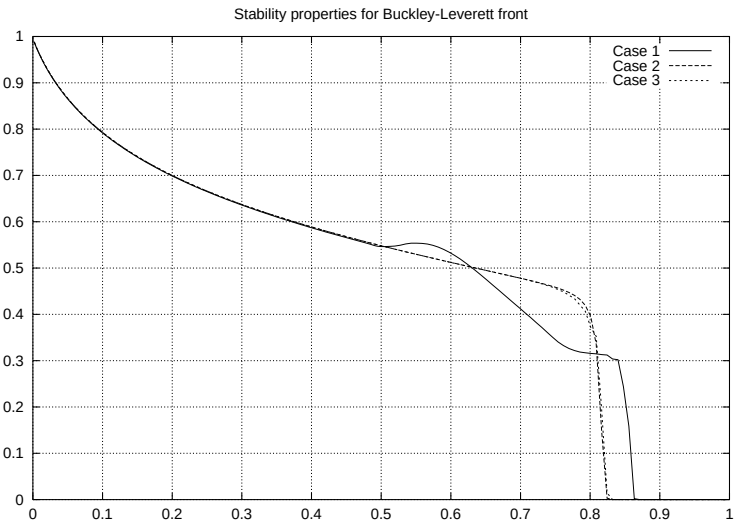


FIG. 7. *Local time stepping cures instability caused by local mesh refinement.*

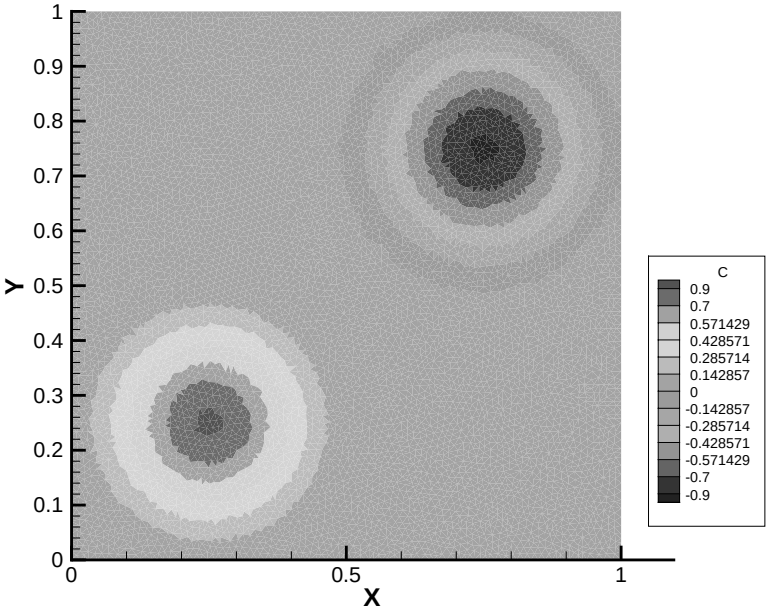


FIG. 8. *Initial condition for Burgers' equation, consisting of a cone of height* 1 *in the lower left corner, and a cone of height* -1 *in the upper right corner.*

5. Two-dimensional Burgers' equation. The local time stepping schemes described above have been extended to two space dimensions. The spatial discretization we have employed is based on a high resolution method developed by Durlofsky, Engquist, and Osher [4] for unstructured, triangular grids. The difference between what we have implemented and the method in [4] is in the slope-limiting step. Three different slopes are constructed using linear interpolation with the element and its neighbors. Then the steepest slope which does not introduce overshoot/undershoot at the edge boundaries is selected. The two methods differ in the case where no slopes satisfy this condition. In our implementation, the slope is set to zero in this case, whereas Durlofsky, Engquist, and Osher simply choose the interpolant with the smallest gradient.

Some results for local time stepping applied to the transport equation (1), with highly varying velocity fields, can be found in [9]. Here, we apply local time stepping to the two-dimensional inviscid Burgers' equation

$$c_t + f(c)_x + f(c)_y = 0, \tag{71}$$

where $f(c) = \frac{c^2}{2}$. We take as an initial condition a function consisting of two cone shapes, one with height 1 and the other with height -1. The initial condition is displayed in Figure 8.

In Burgers' equation, larger concentrations give rise to larger velocities. Consequently, the centers of the cones are advected along at a higher rate than the edges. Moreover, the positive cone has positive velocity and the negative cone negative velocity; thus, the cones approach each other and "collide" in the center of the domain. This general behavior is given in Figure 9, where we have shown the solution at time

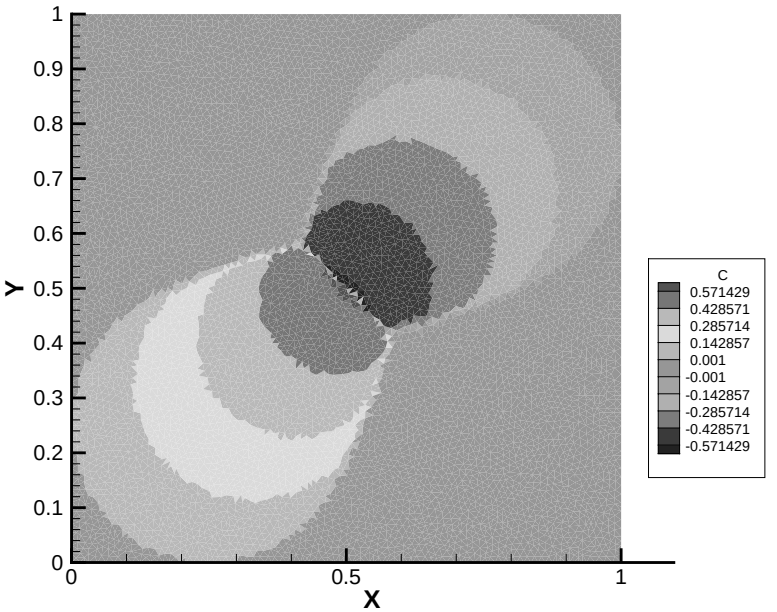


FIG. 9. *Global time stepping solution at* $T = 1.1$.

$T = 1.1$. This solution was obtained using the method in [4], modified as mentioned above, with Heun's method used for the temporal integration. The finite element mesh for this case, shown in Figure 10, consists of 15489 triangles and is unstructured.

An extension of the second order local time stepping scheme described previously was implemented for this problem. In our implementation, the elements take either the global CFL time step or a time step M times larger, assuming that this larger time step does not violate the local CFL constraint. The time steps are redistributed throughout the domain after each large time step. Numerical results for $M = 2$, 5, and 10 were generated. Relative L^1 and L^2 errors between the local and global time stepping solutions were computed at time $T = 1.1$. The relative L^1 error is

$$Error_{L^1} = \frac{\sum_E |c_{global}(x_E) - c_{local}(x_E)| m(E)}{\sum_E |c_{global}(x_E)| m(E)}, \tag{72}$$

where the sum on E is over all elements, x_E is the barycenter of element E, c_{global} is the global time stepping solution, c_{local} is the local time stepping solution, and $m(E)$ is the area of triangle E. A similar definition holds for the relative L^2 error. These errors are given in Table 6. Note that they are on the order of 1%.

Table 7 shows the CPU run times in seconds for each of these cases. Notice a speedup of about 1.7–2.4 for the M:1 time stepping scheme over the global time stepping scheme. For this problem, most elements near the peaks of the cones take the smaller (global CFL) time step, as these elements have larger velocity. As the simulation proceeds, the solution spreads and eventually more elements are forced to take the smaller time step. This phenomenon is seen in Figures 11 and 12, where the local time step distribution is plotted at times $T = .1$ and $T = 1.1$, respectively, for the case $M = 5$. For the $M = 10$ case, the small time step region is more extensive, because fewer elements meet the criterion necessary to take the larger time step. Thus, any gains in increasing M are offset somewhat, and consequently there

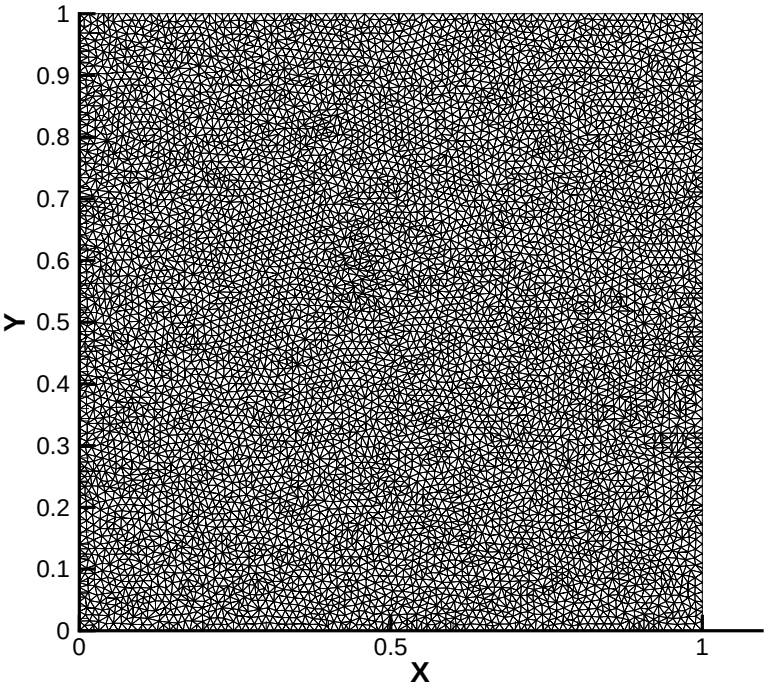


FIG. 10. *Finite element mesh for Burgers' equation test case.*

TABLE 6
Relative L^1 and L^2 errors for M:1 time stepping schemes.

M	Rel. L^1 error	Rel. L^2 error
2	.0041	.0130
5	.0044	.0086
10	.0040	.0074

TABLE 7
Transport run times in seconds for M:1 time stepping schemes.

M	Run time (sec)
1	2533
2	1480
5	1116
10	1075

is not a dramatic decrease in run time as we increase M for this problem. In Figure 13, we have plotted the percentage of elements taking the global CFL time step vs. simulation time for $M = 2$, 5, and 10. Here we see that for $M = 2$, initially about 3.5% of the elements take the global CFL time step, and the percentage grows to about 7% by the final time, $T = 1.1$. For $M = 5$, the range is from about 20% to 27%, and for $M = 10$, the range is about 28% to 37%. For M too large, of course, all elements would be forced to take the global CFL time step.

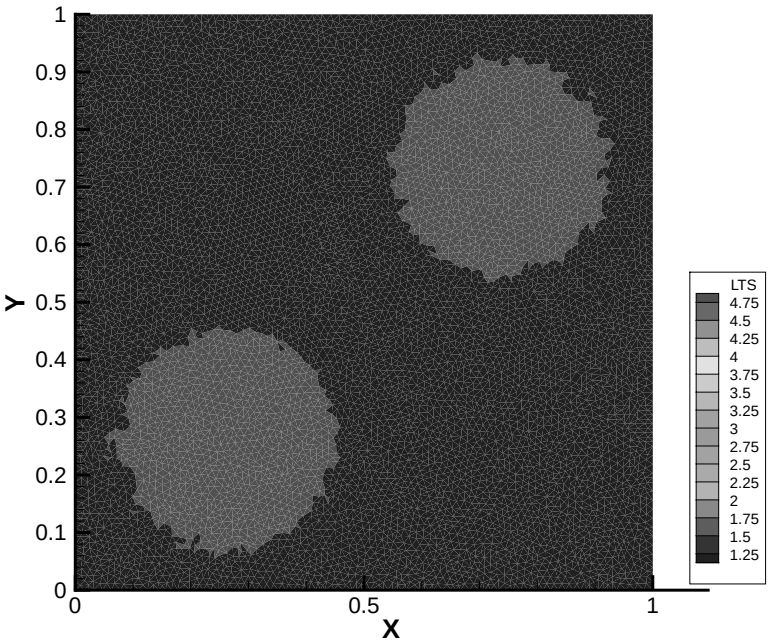


FIG. 11. *Distribution of local time steps for $M = 5$ and $T = .1$. Lighter region indicates where smaller time steps were taken.*

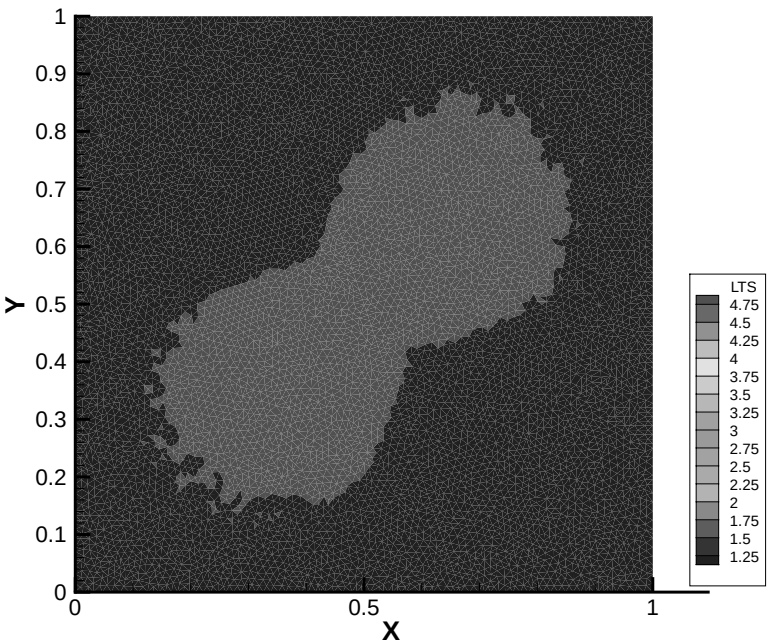


FIG. 12. *Distribution of local time steps for $M = 5$ and $T = 1.1$. Lighter region indicates where smaller time steps were taken.*

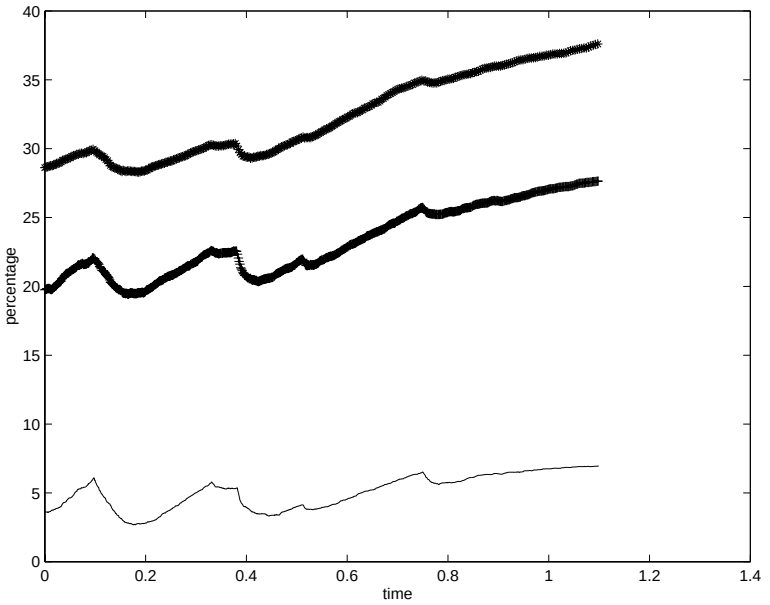


FIG. 13. *Percentage of elements taking the global CFL time step as a function of simulation time for $M = 2$ (solid line), $M = 5$ (+), and $M = 10$ (*).*

6. Conclusions. We have developed and proved maximum principles for local time stepping schemes based on first and second order time discretizations, and piecewise linear spatial discretizations. Numerical results given here and in [9] indicate that these local time stepping schemes exhibit similar accuracy and stability to the global time stepping schemes upon which they are based, at a fraction of the computational cost.

In this paper, we have not addressed the issue of whether our local time stepping schemes satisfy additional properties (TVB, etc.) from which we could conclude that the schemes converge to the entropy solution. However, all of our numerical results to date indicate that the local time stepping solutions are almost identical to those obtained by global time stepping. Thus, our experiences so far lead us to believe that our local time stepping schemes inherit the convergence properties of the related global scheme, though this has not been proven.

REFERENCES

[1] M. BERGER AND J. OLIGER, *Adaptive mesh refinement for hyperbolic partial differential equations*, J. Comput. Phys., 53 (1984), pp. 484–512.

[2] B. COCKBURN AND C.-W. SHU, *TVB Runge-Kutta local projection discontinuous Galerkin finite element method for scalar conservation laws* II: *General framework*, Math. Comp., 52 (1989), pp. 411–435.

[3] C. DAWSON, *High resolution upwind-mixed finite element methods for advection-diffusion equations with variable time-stepping*, Numer. Methods Partial Differential Equations, 11 (1995), pp. 525–538.

[4] L. DURLOFSKY, B. ENGQUIST, AND S. OSHER, *Triangle based adaptive stencils for the solution of hyperbolic conservation laws*, J. Comput. Phys., 98 (1992), pp. 64–73.

[5] J. E. FLAHERTY, R. M. LOY, M. S. SHEPHARD, B. K. SZYMANSKI, J. D. TERESCO, AND L. H. ZIANTZ, *Adaptive local refinement with octree load-balancing for the parallel solution of three-dimensional conservation laws*, Journal of Parallel and Distributed Computing, 47 (1997), pp. 139–152.

[6] S. Gottlieb and C.-W. Shu, *Total variation diminishing Runge-Kutta schemes*, Math. Comp., 67 (1998), pp. 73–85.
[7] A. Harten and S. Osher, *Uniformly high-order accurate nonoscillatory schemes.* I, SIAM J. Numer. Anal., 24 (1987), pp. 279–309.
[8] J. Kallinderis and J. Baron, *Adaptation methods for viscous flows*, in Computational Methods in Viscous Aerodynamics, C. A. Brebbia, ed., Elsevier, Amsterdam, 1990.
[9] R. Kirby, *A Posteriori Error Estimates and Local Time-Stepping for Flow and Transport Problems in Porous Media*, Ph.D. thesis, University of Texas at Austin, Austin, TX, 2000.
[10] R. Leveque, *Numerical Methods for Conservation Laws*, Birkhäuser-Verlag, Boston, 1990.
[11] S. Osher and R. Sanders, *Numerical approximations to nonlinear conservation laws with locally varying time and space grids*, Math. Comp., 41 (1983), pp. 321–336.
[12] C.-W. Shu, *TVB uniformly high-order schemes for conservation laws*, Math. Comp., 49 (1987), pp. 105–121.
[13] C.-W. Shu, *Total-variation-diminishing time discretizations*, SIAM J. Sci. Statist. Comput., 9 (1988), pp. 1073–1084.
[14] B. van Leer, *Towards the ultimate conservative difference scheme,* V. *A second order sequel to Godunov's method*, J. Comput. Phys., 32 (1979), pp. 101–136.